Microsoft SQL Server 2005 (Implementation & Design)

فهرست مطالب

موضوع	صفحه
۱. مقدمه	۱
۲. دیتابیس (Database)	۶
۳. جدول (Table)	۱۰
۴. فهرست (Index)	۱۷
۵. دستور Select	۲۳
۶. توابع تجمیعی (Aggregate Functions)	۲۸
۷. دستورات دستکاری جداول	۴۰
۸. توابع از پیش تعیین شده (Built in Functions)	۴۷
۹. متغیرها و دستورات شرطی	۴۹
۱۰. توابع (Functions)	۵۱
۱۱. روالها (Procedures)	۵۹
۱۲. دستور Triggers	۶۶
۱۳. دستور Transactions	۷۰
۱۴. قفل کردن رکوردها (Locking)	۷۲
۱۵. کنترل خطا (Error Handling)	۷۴
۱۶. دستور Cursor	۷۶

۱. مقدمه :

۱.۱. مقدمه ای بر دیتابیس:

جهت بررسی اینکه آیا SQL Server روی کامپیوتر نصب شده است باید مسیر زیر چک شود:

Control Panel>Administrative Tools>Services>SQL Server (MSSQLServer)

چنانچه سرویس فوق وجود داشت به معنی اینست که S.S نصب شده است.

انواع دیتابیس ها عبارتند از:

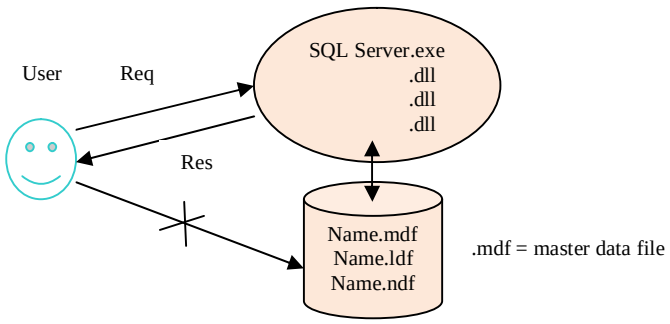
- دیتابیس های Desktop عبارتند از:

.mdb	Access
.dbf	dBaseIV or Clipper

- دیتابیس های Client Server عبارتند از:

DB2
Oracle
MS SQLServer
MySQL

مهمترین ویژگی دیتابیس های Client Server اینست که کاربر بصورت مستقیم به دیتابیس دسترسی ندارد بلکه باید از طریق یک سرویس واسط دستور خود را برای اجرا به سیستم بدهد.



۱.۲. نکات کلی از نصب SQL Server 2005 :

هرچه کامپیوتر با CPU قویتر و RAM بالاتر باشد برای نصب S.S بهتر است. مثلاً برای یک سازمان ۱۵۰ نفره نیاز به یک کامپیوتر با دو CPU و Ram حدود 4G میباشد.

S.S دو نوع ۳۲ و ۶۴ بیتی است ولی در بازار ایران ۶۴ بیتی مرسوم نیست و تنها از نوع ۳۲ بیتی استفاده میشود.

قیمت تقریبی S.S از نوع Enterprise (برای سازمانهای بزرگ) حدود ۲۵۰۰۰ دلار میباشد.

در S.S Enterprise یک تکنیکی بنام Log Shipping داریم. در این روش دو سرور اصلی و فرعی داریم که اطلاعات بصورت منظم و اتوماتیک در فواصل زمانی مشخص از سرور اصلی به سرور فرعی کپی میشود و اگر سرور اصلی خراب شد کاربران اطلاعات را از روی سرور فرعی دریافت می نمایند.

روش دیگر استفاده از تکنیک Clustering است که چند کامپیوتر بهم متصل شده و یک سرور بعنوان سرور اصلی کابران را به سایر سرورها متصل می کند و اگر یکی از سرورها خراب شود سرور اصلی کاربران آن سرور را به سرور دیگر هدایت کرده و کاربر متوجه خرابی سرور قبلی نمی شود.

چنانچه S.S 2005 دارای Service Pack2 باشد میتواند از تکنیک Mirroring استفاده کند. در این روش دیتابیس بر روی دو سرور (نزدیک بهم یا راه دور) قرار داشته و هر دستور SQL بر روی هر دو سرور عمل می کند. بدیهی است اگر سرورها به فاصله دور از هم میباشند (مثلاً در دو شهر مختلف) باید بستر ارتباطی مطمین وجود داشته باشد.

اگر بخواهیم S.S را روی کامپیوتر شخصی خودمان که ویندوز آن مثلاً XP است نصب کنیم بهتر است از نوع Standard یا Developer استفاده شود و نیاز به نصب از نوع Enterprise نیست. ضمناً باید XP از نوع Service Pack2 به بالا باشد.

ولی اگر بخواهیم S.S را برای شرکت یا سازمان نصب کنیم باید از نوع Enterprise بوده و بر روی Windows 2003 با Service Pack1 به بالا نصب گردد چون S.S Enterprise بر روی Windows XP نصب نمی گردد.

نصب S.S نیازمند یک سری امکانات است، مهمترین آن.Net Framework 2.0 میباشد که باید از قبل نصب شده باشد. این برنامه دارای قابلیت بسیار خوب نوشتن برنامه به زبان VB,.NET,C# است.

سرویس دیگر Reporting Service بوده و امکان بسیار خوبی برای گزارشگیری فراهم می آورد. نصب نرم افزار IIS قبل از نصب Reporting Service ضروری می‌باشد. (این نرم‌افزار از طریق Control Panel>Add/Remove… قابل نصب است.)

سرویس S.S Analysis Service برای ساخت گزارشات تحلیلی است. اگر بخواهیم از روی دیتابیسهای خیلی بزرگ گزارشگیری کنیم پروسه ساخت گزارش خیلی کند است. برای این منظور یک مخزن اطلاعات جنبی در نظر گرفته و گزارشات از روی آن تهیه میشود. نتیجه گزارش بر روی یک فضای جداگانه قرار میگیرد و چنانچه اطلاعات تغییر کند تغییرات ایجاد شده بصورت اتوماتیک بر روی گزارشات اعمال میشود. این یکی از سرویسهای بسیار سودمند S.S است.

۱.۳. توضیحاتی در باره Reporting Service:

اگر در زمان نصب S.S و در قسمت Component to Install ردیف Reporting Service فعال شود بخش گزارشگیری خودکار برروی سیستم نصب خواهد شد. این نرم افزار یکی از قویترین و سودمندترین امکانات S.S است که در اینجا به اختصار اشاره ای به آن میشود.

جهت فعال کردن این سرویس باید از طریق زیر وارد شد:

Start>All Program>Microsoft S.S 2005>SQL Server Business Intelligence

پس از آن وارد بخش Microsoft Visual Studio خواهیم شد که جهت تعریف گزارشات حرفه ای در S.S استفاده میگردد.

در این مرحله یک گزارش نمونه از طریق File>New>Project>Report Server Project Wizard ساخته شده و پس از آماده سازی Layout آن، گزارش Preview شده و پس از آن برای مشاهده از طریق اینترانت و اینترنت Deploy خواهیم کرد.

گزارش فوق تحت Web قابل دسترسی و مشاهده است و برای اینکه بتوانیم گزارشی را از طریق Windows مشاهده کنیم باید یک Windows Application ساخته و این گزارش نیز Deploy شود. با اینکار این گزارش نیز در لیست گزارشات قابل استفاده قرار گرفته و همزمان میتوان از یک دیتابیس چند گزارش تحت Web و Windows را مشاهده کرد.

یک مدل دیگر بنام Report Model Project وجود دارد که جهت ساخت گزارش به دلخواه برنامه نویس است ولی قبل از آن باید یک مدل شامل مراحل زیر ساخته شود:

- تعیین Data Source
- ساخت Data Views
- ساخت Data Models

ضمناً میتوان از Microsoft Report Builder جهت ساخت گزارش استفاده کرد.

با توجه به اینکه برخی اطلاعات نسبتاً ثابت میباشند و مرتب تغییر نمی کنند (مانند لیست کالاها) در این حالت میتوان یک Time انقضاء برای گزارش تعیین نمود. اولین نفر که به سیستم متصل میگردد اطلاعات را از SQL میگیرد ولی نفرات بعدی تا پایان زمان انقضاء، گزارش را از روی حافظه Cash دریافت مینمایند.

۱.۴. مراحل نصب:

پس از آماده سازی سخت افزار و نرم افزارهای مورد نیاز مذکور در بخش قبلی، کار نصب را شروع میکنیم. یکی از بخشهای مهم، اجزاء نصب است که در صفحه Components to Install میباشد که اجزاء آن به شرح زیر است:

بخش SQL Server Database Services مهمترین بخش S.S است که باید حتماً انتخاب شود.

همانطور که در بالا اشاره شد بخش دیگر از امکانات S.S سرویس Analysis Services است که امکان تهیه گزارشات تحلیلی را فراهم مینماید.

سرویس بعدی Reporting Services است که به اختصار در بخش قبل به آن اشاره شد.

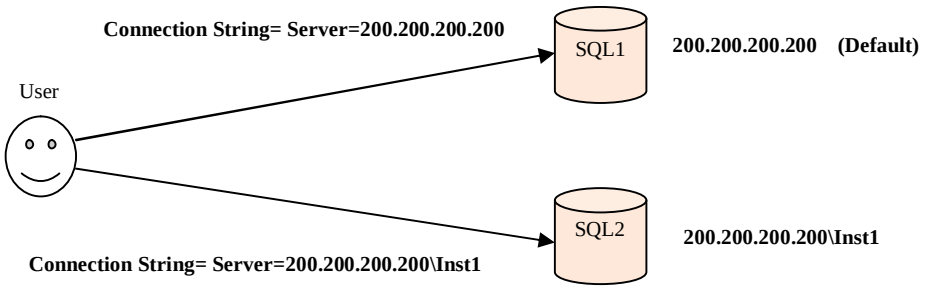
یکی دیگر از بخشهای سیستم Notification Services است که در صورت انتخاب، اخبار سایر سیستمهای متصل به S.S از قبیل خبرگزاریها، سهام، و غیره را اعلام مینماید. بدیهی است نیازی به انتخاب این بخش نمی باشد.

سرویس بعدی Integration Services میباشد که جهت هماهنگ کردن و تهیه گزارشات تجمیعی از فایلها و سیستمهای متفاوت مورد استفاده قرار میگیرد. این سرویس میتواند اطلاعات را دریافت یا ارسال نماید.

آخرین سرویس Workstation com… است که همانند سرویس Notification مورد نیاز نمیباشد.

پس از آن وارد صفحه مشخصات یا Feature Selection می شویم. در این بخش نوع و محل سرویسها را مشخص میکنیم.

صفحه Instance Name برای تعریف و نصب چندین دیتابیس S.S بر روی یک کامپیوتر است. شکل زیر آدرس دهی و نام گذاری چندین S.S را نشان میدهد:



اگرچه می‌توان روی یک کامپیوتر تعداد ۶۴ سرویس یا SQL را نصب نمود، ولی حداکثر ۴ عدد Instant را میتوان نامگذاری و تعریف نمود.

بخش بعدی از نصب S.S سرویس Service Account است. در این بخش نام کاربری که میتواند سیستم SQL را فعال یا خاموش کند مشخص مینماید. ولی اگر Local System انتخاب و فعال شود کامپیوتر به محض روشن شدن کامپیوتر بلافاصله به SQL متصل شده و نیازی به کاربر خاص ندارد.

همزمان با نصب SQL تعدادپنج سرویس زیر نیز قابل نصب شدن میباشد که میتوان هر یک از آنها را برای نصب شدن، فعال و انتخاب کرد.

1. SQL Server
2. SQL Server Agent
3. SQL Browser
4. Reporting Services
5. Analysis Services

اولین سرویس SQL Server میباشد که سرویس اصلی SQL میباشد و باید انتخاب شود.

سرویس S.S Agent وظیفه رییس دفتری SQL را برعهده دارد و مثلاً هر شب سر ساعت ۱۰ از اطلاعات Backup میگیرد.

با انتخاب سرویس SQL Browser در زمان رویت نام SQL ها توسط کاربران مختلف، نام این SQL نیز نشان داده خواهد شد در غیر اینصورت نام آن نشان داده نمیشود.

سرویسهای Reporting و Analysis نیز در بخشهای فوق توضیح داده شد.

بخش بعدی Authentication Mode است که تعیین نکات ایمنی در زمان اتصال به SQL میباشد. چنانچه Windows Authentication Mode انتخاب گردد کلمات رمز و دسترسی کاربران به SQL بر اساس مقادیر تعریف شده در Windows انجام میگردد ولی اگر Mixed Mode انتخاب شود باید برای دسترسی به SQL مجدداً نام کاربر و سطح دسترسی تعیین گردد.

بخش بعدی Collation Setting است. این بخش در مورد زبانهای مختلف بحث می کند. مثلاً برای سیستم عامل DOS از کدهای ASCII استفاده می شود. ولی در Windows توسط برنامه نویسان از Code Page های مختلف برای برنامه نویسی و زبانهای مختلف استفاده میشود.کدپیج شماره 1256 برای زبان فارسی درنظر گرفته شده است. این روش مشکل زبانها در ویندوز را حل کرد ولی در اینترنت هنوز مشکل پابرجا بود چون در اینترنت ممکن است نیاز باشد همزمان چند زبان روی صفحه نمایش داده شود (مانند صفحات وب سایت خبرگزاریها که برخی کلمات به چند زبان نشان داده شده است) در این حالت از تعریف Unicode استفاده میشود و چون برای نمایش هر کاراکتر یا حرف از دو بایت استفاده میشود لذا حروف تمام زبانهای معروف دنیا در این کد پیش بینی شده است.

چنانچه متن به غیر از روش Unicode ذخیره شود یک بایتی بوده و باید نوع زبان آن توسط Collation مشخص شود.

در واقع Collation سرویس و امکانی در S.S است که نحوه Sort حروف الفبای یک زبان خاص و همچنین مقایسه حروف آن زبان را در بر دارد.

معمولاً دو روش Collation در سیستمهای جدید تعریف شده است:

1. SQL Server Collation
2. Windows Collation

در Windows 2000 به بالا برای زبانهای عربی و فارسی یک Collation بنام Arabic_CI_AI تعریف شده است که برای فارسی کاربرد دارد.

در صفحه Collation Settings برای دو حالت Binary و Binary–Code Point مرتب سازی یا Sort بر اساس کد ASCII انجام میگردد ولیکن در چهار حالت دیگر (Case, Accent, Kana, Width) مرتب سازی بر اساس حروف الفبا انجام میگردد.

در حالت Case–Sensitive، حروف بزرگ و کوچک لاتین متفاوت خواهد بود.

اگر حالت Accent–Sensitive انتخاب شود لهجه های متفاوت (مانند U و Ů در برخی زبانها) متفاوت خواهند بود.

حالت width-sensitive ، به طول حساسیت خواهد داشت یعنی اگر حرفی در کد یک بایتی یا دو بایتی ذخیره شده باشد متفاوت نشان داده خواهد شد.

حالت Kana–Sensitive مربوط به زبانهای چینی بوده و در زبان فارسی کاربردی ندارد.

چنانچه قسمت Customize for each Service Account علامت زده شود امکان تعریف Collation های متفاوت برای Windows و SQL وجود خواهد داشت.

ضمناً در زمان تعریف SQL و Table مجدداً باید Collation تعریف شود در غیر اینصورت همین Collation از پیش تعریف شده فرض خواهد شد.

۲. دیتابیس (Database) :

۲.۱. مشخصات دیتابیس:

پس از نصب S.S جهت ساخت یک دیتابیس نمونه به روش زیر وارد SQL میشویم:

Start>All Programs>Microsoft S.S 2005>Configuration Tools>S.S Config Manager

پس از ورود به صفحه اصلی میتوان مشخصات اصلی و سرویسهای SQL و شبکه را مشاهده و اصلاح نمود. مثلاً در قسمت SQL Server 2005 Services عناوین سرویسهای SQL و فعال یا فعال نبودن آنها نشان داده خواهد شد.

و یا در بخش Protocols for MSSQLSERVER انواع پروتکلهای قابل قبول را نشان می دهد. به عبارتی تنظیمات شبکه برای سرور در این بخش و تنظیمات شبکه برای کاربران در بخش بعدی انجام خواهد شد.

مثلاً در همین بخش Protocols for MSSQLSERVER چهار پروتکل را پشتیبانی می کند که معروفترین و پرکاربردترین آن TCP/IP است. چنانچه بر روی TCP/IP کلید سمت راست Mouse کلیک و Properties انتخاب شود و IP Addresses نیز از بالای صفحه انتخاب گردد آدرسهای سرور اصلی را نمایش خواهد داد. قسمت IP1 مربوط به آدرس دستگاه کاربر اصلی و IP2 مربوط به آدرس 127.0.0.1 است که برای Loop کردن با دستگاه خودتان میباشد.

در قسمت IPAll آدرس پورت SQL که همیشه 1433 میباشد در TCP Port قرار داده شده است. توضیح اینکه در زمان ارسال اطلاعات هر بسته شامل چهار بخش به شرح زیر میباشد:

DATA	IP Source	IP Destination	Port

قسمت Port آدرس هر نرم افزار خاص است که مثلاً 80 برای Internet Explorer و 21 برای FTP بوده و همیشه آدرس 1433برای SQL Server میباشد. علت اصلی پیش بینی آدرس Port برای هر نرم افزار اینست که با رسیدن هر بسته اطلاعاتی، تمام کاربران برای دریافت آن هجوم نبرده و وقت خود و سیستم را اشغال ننمایند و تنها زمانی سراغ بسته اطلاعاتی بروند که مربوط به نرم افزار فعال خودشان میباشد.

نکته مهم اینکه کار اصلی Firewall بستن Port های مختلف است و باید دقت شود آدرس 1433 جزو لیست مسدودیها نباشد.

۲.۲. ورود به دیتابیس:

جهت ورود به دیتابیس SQL باید از طریق زیر اقدام کرد:

Start>All Programs>Microsoft S.S 2005>Sql Server Management Studio

پس از آن صفحه ای برای اتصال به سرور بنام Connection to Server ظاهر شده که باید نام سرور مورد نظر را برای اتصال به آن مشخص نماییم. نام سرور میتواند به اشکال مختلف به شرح زیر باشد:

- Server33 or Server33\Inst1 — نام سرور
- 200.200.200.200 or 200.200.200.200\Inst1 — آدرسIP
- 127.0.0.0 or 127.0.0.0\Inst1 — وصل به خودم
- (Local) or (Local)\Inst1 — وصل به سرور پیش فرض
- . or .\Inst1 — وصل به سرور Domain
- x — همان نامی که در Alias برده شده است.

پس از آن باید Authentication انتخاب شود. در صورت انتخاب .Windows Auث، کاربر بدون کلمه رمز به SQL وصل خواهد شد ولی اگر .SQL Auث انتخاب شود باید کلمه رمز داده شود.

۲.۳. ایجاد دیتابیس:

جهت ورود به دیتابیس SQL باید از طریق زیر اقدام کرد:

Start>All Programs>Microsoft S.S 2005>Sql Server Management Studio

پس از ورود به صفحه اصلی SQL میتوان با زدن کلید سمت راست Mouse بر روی نماد Databases یک دیتابیس جدید ایجاد نمود. در این حالت صفحه ای بنام New Database ظاهر شده و باید نام دیتابیس مشخص گردد.

سپس باید نوع فهرست گذاری برای کلمات متن تعیین شود. اگر عنوان Use Full text Index انتخاب و فعال شود برای تمام کلمات ایندکس ساخته شده و در زمان جستجوی کلمات به سرعت متنی که دارای آن کلمه باشد پیدا شده و ارایه میگردد. بدیهی است باید در این حالت برخی دستورات برنامه متناسب با این حالت تغییر یابد. مثلاً اگر در حالت عادی دستور جستجوی کلمه test بدین شرح است:

Where DESC like ‘test’

در حالت فعال بودن Use Fulltext Index شکل دستور بصورت زیر تغییر خواهد یافت:

Where contains (DESC, ‘test’)

هر دیتابیس دارای دو نوع فایل به نام زیر خواهد بود:

Master Data File	.mdf	فایل اطلاعات اصلی دیتابیس
Log Data File	.ldf	فایل تاریخچه دستورات

در زمان تعریف هر یک از فایلها میتوان از طریق Initial Size مقدار اولیه حجم فایل را تخمین زده و در قسمت Autogrowth مقدار افزایش حجم فایل را پس از رسیدن به حداکثر حجم تعیین شده، مشخص نمود. در این قسمت میتوان مقدار مشخص یا درصدی از حجم تعیین شده را ذکر نمود. ضمناً در همین قسمت میتوان حداکثر حجم مجاز فایل را تعیین کرده و یا اینکه حداکثر حجم مجاز را نامحدود تعیین کرد.

پس از آن میتوان محل ذخیره کردن فایل را در قسمت Path تعیین نمود.

برای انجام هر کاری در S.S یک دستور بنام T_sql صادر و اجرا میشود. مثلاً در زمان ایجاد دیتابیس جدید که از طریق منوی New Database انجام میگردد یک سری دستور T_sql ایجاد شده و برای سیستم ارسال میگردد. برای دیدن و اصلاح دستورات تولید شده میتوان کلید Script که در بالای صفحه New Database قراردارد را زده و سپس دستورات را مشاهده و در صورت نیاز اصلاح نمود. جنانچه در این صفحه مشاهده میگردد یکسری دستورات به شکل ...ALTER. وجود دارد. بنابراین میتوان گفت SQL یک زبان Command Processor است.

S.S در هر لحظه فقط یک دستور را میتواند اجرا نماید و پس از اجرای دستور، تاریخچه و زمان اجرای دستور در Logfile ذخیره میشود. این فایل دارای منافع و کاربردهای زیادی است که مختصراً به چند مورد آن اشاره میگردد.

فرض کنید در پایان هر روز از اطلاعات Backup گیری میشود ولی در وسط روز بنا بهر دلیلی فایل اصلی خراب شود و یا هارد دیسک با اشکال مواجه شود. اگر فایل های Master و Log روی دو هارد دیسک جداگانه نگهداری شده باشند میتوان در ابتدا فایل اصلی را از Backup شب گذشته بازخوانی کرده و سپس بکمک دستورات موجود بر روی Logfile از زمان Backup تا کنون نسبت به بازسازی فایل اصلی اقدام کرد.

یکی دیگر از کاربردهای Logfile استفاده در Replication است. مثلاً فرض کنید بنا بهر دلیلی لازم است دو نسخه از فایل اصلی بر روی دو سرور جداگانه (مثلاً در مشهد و تهران) نگهداری شود. بروزرسانی همزمان و Online فایلهای اصلی دو سرو نیاز به پهنای باند وسیع مخابراتی دارد درحالی که میتوان در پایان روز تنها دستورات اجرا شده روزانه از روی Logfile برای سرور مقابل ارسال شده و در آنجا دستورات اجرا شده و فایل اصلی بروزرسانی شود. ضمناً از همین روش نیز میتوان برای ساخت فایل Mirror در دو مکان مختلف استفاده کرد.

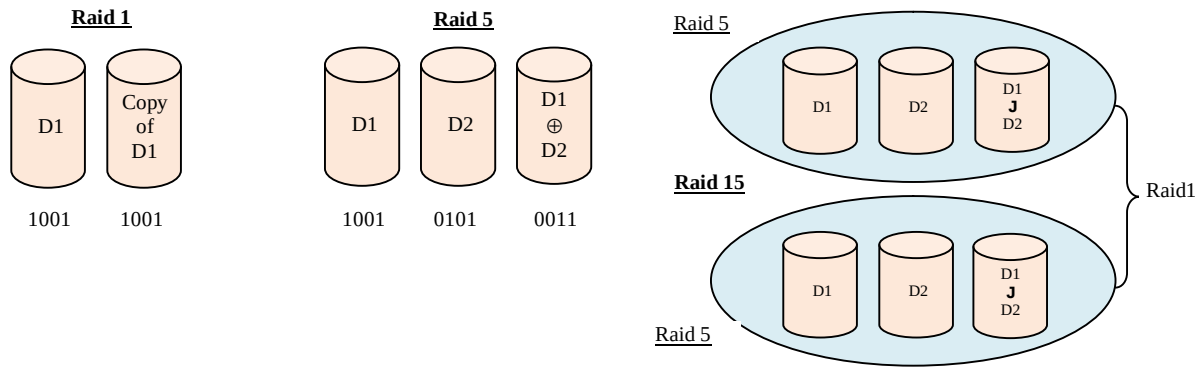
با توجه به نکات فوق توصیه میگردد محل نگهداری فایلهای.mdf و.ldf جداگانه و بر روی دو هارد دیسک جداگانه باشد. مگر اینکه از تکنیک Raid برای نگهداری اطلاعات استفاده شود که به اختصار در مورد این تکنیک اشاره ای میگردد.

روش ساده اولیه Raid 1 است که به این روش Mirror نیز گفته میشود. بر اساس این روش اطلاعات همزمان بر روی دو دیسک ذخیره میگردد. چنانچه اطلاعات هر دیسک خراب شود از دیسک دوم بازیابی میشود. بدیهی است این روش دارای دو ایراد سرعت پایین و حجم ذخیره سازی بالا خواهد بود.

یکی دیگر از روشهای ذخیره سازی اطلاعات Raid 5 است در این روش اطلاعات بر روی دو هارد دیسک نگهداری شده و XOR آنها بر روی هارد دیسک سوم قرار میگیرد. در این حالت اگر اطلاعات هر یک از دو دیسک اصلی خراب شود میتوان با XOR کردن دو دیسک دیگر، اطلاعات دیسک خراب را بازیابی کرد.

روش دیگر Raid 15 است که از ترکیب دو روش قبلی است که برای داده های بسیار با اهمیت SQL این روش توصیه میگردد.

مثالهای زیر نمونه ای از روشهای فوق را نشان می دهد:



ضمناً میتوان فایلها را دسته بندی کرده و در گروههای مختلف قرار داد. برای این منظور باید در زمان تعریف دیتابیس نام گروه فایل را در Filegroup تعریف نمود. بدیهی است نام گروه فایل اولیه Primary است و اگر در صفحه New Database کلید Add زده شود امکان تعریف یک فایل جدید بوجود آمده و در این قسمت میتوان نام یک گروه فایل جدید در Filegroup داده شود. از طرفی دیگر در صفحه Properties که در زمان تعریف Table در سمت راست صفحه اصلی ظاهر میشود (این منو با زدن کلید F4 برروی قسمت Tables نیز قابل رویت است) نیز میتوان در قسمت Failegroup or Partition نسبت به تعیین نام گروه فایل نیز اقدام کرد. یکی از کاربردهای Filegroup تعیین استراتژی Backup گیری از فایلهاست مثلاً در یک سیستم حسابداری باید فایلهای اسناد بصورت روزانه کپی برداری شود ولی کپی‌برداری ماهانه از فایل کدینگ حسابداری کافیست.

فرض کنید قصد طراحی سیستم حسابداری حاوی اسناد سالهای مختلف را داشته باشیم. سه روش زیر وجود دارد:

۱. قراردادن فایلهای هر سال بر روی فایلهای مختلف با نامهای گوناگون و تغییر نام فایلها در Source برنامه در زمان نیاز برای کارکردن با اسناد هر سال مشخص.
۲. قراردادن فایلهای هر سال بر روی فایلهای مختلف در درایوهای مختلف و تغییر آدرس دسترسی به فایلها در زمان نیاز برای کارکردن با اسناد هر سال مشخص.
۳. قراردادن اطلاعات فایلهای تمام سالها بر روی یک فایل و تفکیک سالها بوسیله یک فیلد بر روی فایل اصلی.

بدیهی است راهکار اول روش خوبی نیست ولیکن راهکارهای دوم و سوم هر کدام دارای مزایا و معایبی است. بعنوان مثال راهکار دوم سریعتر میباشد ولیکن امکان تهیه گزارشات تجمیعی از سالهای مختلف را ندارد و در راهکار سوم اگرچه امکان تهیه گزارشات تجمیعی وجود دارد ولی همیشه با یک فایل بسیار بزرگ سر و کار داریم و مثلاً در زمان Backup مجبور به کپی برداری از تمام فایل میباشیم.

در S.S 2005 یک امکان جدیدی اضافه شده است که در نسخه های قبلی S.S وجود نداشته است این امکان پارتیشن بندی یک فایل و قرار دادن محتوای یک فایل در چند پارتیشن جداگانه و در نتیجه هارد دیسکهای جداگانه است. مثلاً در شکل زیر اسناد حسابداری سالهای قبل از ۸۶، اسناد سال ۸۶ و اسناد بعد از سال ۸۶ یک فایل در سه پارتیشن جداگانه نگهداری میشود:

Create Function $x < 1386$, $1386 \leq x < 1387$, $x \geq 1387$

با امکان پارتیشن بندی در S.S 2005 میتوان روش دوم و سوم را برای طراحی سیستم حسابداری انتخاب کرد در نتیجه معایب روشها برطرف شده و محاسن هر دو روش بدست خواهد آمد. بدین ترتیب که بکمک ویژگی پارتیشن بندی، اسناد هر سال را در یک پارتیشن و هارد دیسک جداگانه قرار داد (روش دوم) ولی تمام اطلاعات در داخل یک فایل قرار گیرد (روش سوم).

ضمناً در قسمت Database Properties (به روی نام دیتابیس اگر کلید سمت راست Mouse زده شود نماد Properties قابل انتخاب است) میتوان نام یک Filegroup را بعنوان Default تعیین کرد. با این انتخاب چنانچه نام گروه فایل هر دیتابیس یا جدولی تعیین نشود، نام گروه فایل Default بعنوان پیش فرض برای آن در نظر گرفته خواهد شد.

در همین صفحه Properties دیتابیس اگر ردیف Option انتخاب گردد در بالای صفحه یک قسمت بنام Recovery Model وجود دارد. اگر حالت Simple انتخاب شود اصلاً فایل Log ساخته نمیشود. در این حالت سرعت عملیات افزایش می‌یابد ولیکن در صورت خرابی اطلاعات امکان بازیابی وجود ندارد. دو حالت Full و Bulk_Logged بصورت کامل از عملیات، دستورات و اتفاقات بر روی فایل Log یک اثر گذاشته میشود ولیکن تفاوت مختصری در دستورات Bulk خواهند داشت. (دستورات Bulk مربوط به فایلهایی است که فیلدها و رکوردها با علایم جداکننده مانند ویرگول از هم جدا میشوند.)

یکی دیگر از نکات تنظیمات اولیه S.S مربوط به Compatibility Level است که در همان صفحه Option از Properties مرتبط با دیتابیس میباشد. S.S 2005 بصورت اتوماتیک فایلهای S.S 2000 را به S.S 2005 تبدیل میکند ولی تمام امکانات S.S 2005 بر روی فایلهای تبدیل شده قابل استفاده نمی باشد مگر اینکه در قسمت Compatibility Level حالت SQL Server 2005 (90) انتخاب شود.

۳. جدول (Table):

برای ساخت یک جدول باید ابتدا دیتابیس مربوطه انتخاب شده سپس بر روی Table به کمک کلید سمت راست Mouse ردیف New Table انتخاب گردد.

۳.۱. انواع فیلدها :

برای ساخت جدول باید ابتدا فیلدها یا ستونهای جدول تعریف شود. فیلدها دارای انواع مختلف میباشند که به اختصار توضیح داده خواهند شد.

۳.۱.۱. Binary Data:

این نوع فیلد جهت نگهداری اطلاعات بصورت باینری (اعداد دودویی یا مبنای ۲) از قبیل تصاویر استفاده شده و شامل ۴ نوع به شرح زیر بوده و سریعترین نوع فیلد نیز میباشند:

- Binary — باطول ثابت — از ۱ تا حداکثر ۸۰۰۰ بایت (در این حالت اطلاعات داخل فیلد قرار میگیرد)
- VarBinary — باطول متغیر — از ۱ تا حداکثر ۸۰۰۰ بایت (در این حالت اطلاعات داخل فیلد قرار میگیرد)
- Image — باطول ثابت — از ۱ تا حداکثر ۲ گیگا بایت (در این دو حالت فقط اشاره گر داخل فیلد و اصل
- VarBinay(Max) — باطول متغیر — از ۱ تا حداکثر ۲ گیگا بایت اطلاعات در فایلهای جداگانه ۸ کیلوبایتی قرار میگیرد)

۳.۱.۲. Character Data:

این نوع فیلد برای نگهداری عبارات و حروف Ascii بوده و در ضمن سریع نیز میباشد. در این روش برای نگهداری هر حرف یک بایت اشغال شده و لذا نیاز به Collation برای تعیین زبان اطلاعات میباشد. این فیلد نیز شامل ۴ نوع به شرح زیر است:

- Char — باطول ثابت — از ۱ تا حداکثر ۸۰۰۰ حرف (در این حالت اطلاعات داخل فیلد قرار میگیرد)
- VarChar — باطول متغیر — از ۱ تا حداکثر ۸۰۰۰ حرف (در این حالت اطلاعات داخل فیلد قرار میگیرد)
- Text — باطول ثابت — از ۱ تا حداکثر ۲ مگا حرف (در این دو حالت فقط اشاره گر داخل فیلد و اصل
- VarChar(Max) — باطول متغیر — از ۱ تا حداکثر ۲ مگا حرف اطلاعات در فایلهای جداگانه ۸ کیلوبایتی قرار میگیرد)

نکته: فیلدهای از نوع Image و Text در SQL 10 حذف شده و باید بجای آنها از VarBinary(Max) و VarChar(Max) استفاده نمود.

۳.۱.۳. Unicode Data:

این نوع فیلد برای نگهداری عبارات و حروف Unicode بوده و خیلی سریع نمیباشد. برای نگهداری هر حرف این نوع فیلد دو بایت اشغال شده و نسبت به حالت قبل حافظه بیشتری را بخود اختصاص میدهد. این نوع فیلد نیاز به Collation برای تعیین زبان اطلاعات ندارد. کاربرد این نوع فیلد برای طراحی سیستمهای تحت Web میباشد. این فیلد نیز شامل ۴ نوع به شرح زیر است:

- nChar — باطول ثابت — از ۱ تا حداکثر ۴۰۰۰ حرف (در این حالت اطلاعات داخل فیلد قرار میگیرد)
- nVarChar — باطول متغیر — از ۱ تا حداکثر ۴۰۰۰ حرف (در این حالت اطلاعات داخل فیلد قرار میگیرد)
- nText — باطول ثابت — از ۱ تا حداکثر ۱ مگا حرف (در این دو حالت فقط اشاره گر داخل فیلد و اصل
- nVarChar(Max) — باطول متغیر — از ۱ تا حداکثر ۱ مگا حرف اطلاعات در فایلهای جداگانه ۸ کیلوبایتی قرار میگیرد)

۳.۱.۴. Integer Data:

این نوع فیلد برای نگهداری اعداد صحیح و بدون اعشار استفاده میگردد. این نوع فیلد سریع بوده و دارای ۴ نوع به شرح زیر است:

- TinyInt 1 Byte 0 - 255
- SmallInt 2 Byte -32768 - +32767
- Int 4 Byte ± 2 Milliard
- BigInt 8 Byte ± 4 Milliard

۳.۱.۵. Money Data:

این نوع فیلد برای نگهداری اعداد غیر صحیح با تعداد ارقام اعشار ثابت ۴ رقمی استفاده میگردد. این نوع فیلد سریع نبوده و دارای ۲ نوع به شرح زیر است:

- SmallMoney 4 Byte ۴ رقم اعشار . ۶ رقم صحیح
- Money 8 Byte ۴ رقم اعشار . ۱۵ رقم صحیح

۳.۱.۶. Approximate Data:

این نوع فیلد برای نگهداری اعداد غیر صحیح با تعداد ارقام اعشار متغیر و یا تخمینی استفاده میگردد. این نوع فیلد کند بوده و استفاده از آن توصیه نمیگردد و دارای ۲ نوع به شرح زیر است:

- Real 4 Byte تعداد اعشار مشخص نیست.
- Float 8 Byte اعداد بصورت نمایی (توانهایی از ۱۰) نگهداری میشود.

۳.۱.۷. Decimal(Numeric) Data:

این نوع فیلد برای نگهداری اعداد اعشاری با تعداد اعشار مشخص استفاده میگردد. این نوع فیلد بسیارکند بوده و اصلاً استفاده از آن توصیه نمیگردد و فقط دارای یک نوع با فرمت زیر است.:

- Decimal(Precision, Scale) (تعداد اعشار و طول فیلد)

مثلاً اگر فیلدی بصورت Decimal(5,2) تعریف شود حداکثر مقدار فیلد برابر 999.99 خواهد بود.

۳.۱.۸. Date & Time Data:

این نوع فیلد برای نگهداری تاریخ میلادی و ساعت استفاده میشود و برای تاریخ شمسی کاربرد ندارد و بهتر است برای تاریخ شمسی از Char(8) استفاده شود. مثلاً تاریخ ۸۷/۲/۱۰ بصورت '۱۳۷۸۰۲۱۰' نگهداری میشود.

- Smalldatetime 4 Byte 1900 – 2079 تا هزارم ثانیه
- Datetime 8 Byte 1700 – 9999 تا هزارم ثانیه

۳.۱.۹. Special Data:

برخی از انواع فیلدهای خاص در این بخش به اختصار توضیح داده میشود:

- Bit 1 bit 0, 1, Null

این فیلد یک بیتی بوده و کاربرد آن در زمانی است که دو حالت وجود داشته باشد مانند جنسیت.

- Timestamp 8 Byte

در این فیلد تایم لحظه ای اجرای دستور نگهداری میشود. یکی از کاربردهای آن کنترل بروزرسانی همزمان فایلها توسط چند کاربر است.

- Uniqueidentifier 16 Byte

در زمان اجرای سیستم، یک کد غیر تکراری بین المللی در این فیلد GUID (Global Unique Identifier Code) قرار گیرد. یک کد قرار میگیرد. کنترل عملیات بر روی فایلهای یکسان در سرورهای متفاوت و راه دور، یکی از کاربردهای آن است.

- Sql_variant

این نوع فیلد برای نگهداری انواع داده ها استفاده میشود و نوع آن با توجه به اولین مقداری که در آن قرار گیرد تعیین خواهد شد. چون نوع و حجم فیلد مشخص نیست لذا تنها یک اشاره گر ۱۶ بایتی در آن قرار گرفته و داده اصلی در فایل جداگانه نگهداری میشود بنابراین استفاده از این نوع فیلد بهیچ عنوان توصیه نمیگردد.

- Cursor

این فیلد مربوط به کنترل Cursor است که بعداً توضیح داده خواهد شد.

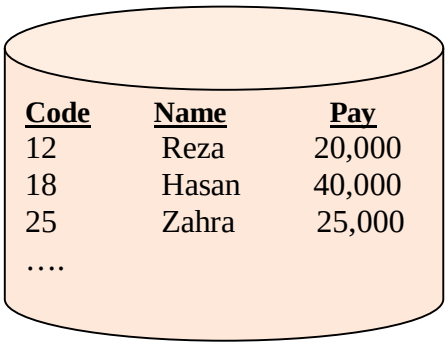
- Table

مقادیر برخی از توابع بصورت جدول برگردانده میشود که میتوان از این نوع فیلد استفاده کرد. این نوع فیلد نیز متعاقباً توضیح داده خواهد شد.

- xml

این فیلد برای انتقال اطلاعات و دستورات تحت web استفاده شده و شامل انواع دستورات با Meta Data های مختلف است.

فرض کنید بخواهیم فایل زیر را در سیستم نگهداری کنیم. یکی از روشهای نگهداری آن، استفاده از یکسری Meta Data است که با استفاده از xml و به شکل زیر ساخته میشود.




```
<Persons>
    <Person><Code>12</Code><Name>Reza</Name><Pay>20,000</Pay>
    <Person><Code>18</Code><Name>Hasan</Name><Pay>40,000</Pay>
    <Person><Code>25</Code><Name>Zahra</Name><Pay>25,000</Pay>
    ……
</Persons>
```

با استفاده از دستورات xmlDom و xmlxpath میتوان داخل ساختار اطلاعاتی فوق حرکت کرده و عملیات جستجو و تغییر را انجام داد.این دستورات در این درس توضیح داده نخواهد شد.

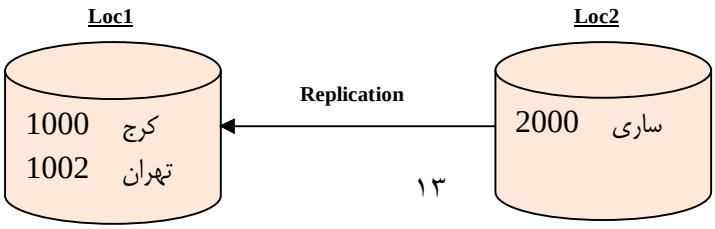
۳.۲. تعریف سایر مشخصات جدول:

در زمان تعریف فیلدها یا ستونهای جدول، یک صفحه بنام Column Properties در پایین صفحه اصلی نمایش داده میشود که میتوان با توجه به نوع فیلد، مشخصات بیشتری از قبیل مقدار اولیه، شرایط و غیره را تعریف کرد. در ادامه بحث به برخی از این مشخصات اشاره میگردد:

یکی از مشخصات، تعریف مقدار اولیه پیش فرض برای فیلد است. باید مقدار اولیه مورد نظر در مقابل Default value or Binding قرار گیرد.

اگر برای فیلدهای عددی غیر اعشاری Identity Specification برابر Yes شود در اینصورت این فیلد بصورت اتوماتیک اضافه خواهد شد. مقدار شروع در قسمت Identity seed و مقدار افزاینده در Identity Increment قرار خواهد گرفت. ضمناً برای استفاده از این امکان باید دقت شود در قسمت Default Value مقدار پیش فرض داده نشود.

ردیف Not for Replication مربوط به شرایط انتقال اطلاعات بین فایلهای مشابه در دو سرور مختلف و یا بعبارتی مربوط به زمان استفاده از Replication است. فرض کنید فایل کد شهرستانها به شرح زیر در دو شهر مختلف بگونه ای تعریف شده است که مقدار شروع کد برای Loc1 از ۱۰۰۰ و با افزاینده ۲ و برای Loc2 از ۲۰۰۰ با افزاینده ۲ میباشد. اگر بخواهیم با تکنیک Replication اطلاعات Loc1 به Loc2 منتقل شود چنانچه مقدار Not for R.. برابر Yes باشد کد ساری از ۲۰۰۰ به ۱۰۰۴ تبدیل شده و سپس دراخل فایل Loc1 قرار میگیرد در غیر اینصورت با همان مقدار ۲۰۰۰ در فایل Loc1 قرار میگیرد.



چنانچه فیلد از نوع Unicode (مانند nchar) باشد باید کد استاندارد زبان آن مشخص شود. با زدن کلید Collation منوی انتخاب زبان ظاهر خواهد شد. در این بخش دو حالت SQL Collation و Windows Collation وجود دارد که بهتر است از Windows Collation استفاده شده و پس از آن زبان Arabic و حالت Dictionary Sort انتخاب گردد.

در این بخش از طریق New Table یک فایل بنام Test1 و با مشخصات فیلدهای زیر ساخته میشود:

- فیلد Code از نوع int با مقدار اولیه 1000 و افزاینده 2
- فیلد Name از نوع varchar
- فیلد Amount از نوع Decimal(5,2)
- فیلد x از نوع uniqueidentifier با مقدار پیش فرض NewID()

پس از اتمام تعاریف فیلدها، مشخصات جدول را با نام Test1 ذخیره کرده و با انتخاب Open Table وارد جدول شده و چندین رکورد داده ثبت خواهد شد.

چنانچه قبلاً نیز گفته شد SQL برای انجام هر کاری یک دستورالعمل را تولید و اجرا میکند. بعنوان مثال برای ساختن جدول فوق میتوان مستقیماً برنامه نویسی کرده و با اجرای آن، جدول را تولید کرد. برنامه ساخت جدول فوق بنام Test2 به شکل زیر است که باید با انتخاب New Query (سمت چپ بالای صفحه SQL) این برنامه تایپ شده و سپس با زدن کلید Execute ! برنامه را اجرا نمود. پس از آن یک جدول بنام Test2 همانند Test1 ساخته خواهد شد:

```
Create Table Test2
    (Code       int Identity (1000, 2),
     Name       varchar (50),
     Amount     Decimal (5, 2),
     X          Uniqueidentifier    default (NewID()))
```

برنامه زیر را نیز همانند برنامه فوق برای ساختن یک جدول جدید بنام Test3 مینویسیم:

```
Create Table Test3
    ( a         int,
      b         int,
      c         as a+b)
```

چنانچه ملاحظه میگردد در این جدول نوع فیلد c تعیین نشده است و نوع آن همان نوع int فیلدهای اصلی است. ضمناً برای این فیلد فضایی در جدول در نظر گرفته نخواهد شد و همیشه مقدار این فیلد بر اساس محاسبه جمع دو فیلد a و b بدست می آید. برای در نظر گرفته فضای جداگانه و پرهیز از محاسبه مکرر این فیلد باید در صفحه مشخصات جدول، مقدار Computed Column Specification برابر Yes شود. ضمناً فرمول a+b در این قسمت قرار میگیرد.

از طرفی دیگر چنانچه بخواهیم Source برنامه یک جدول که از طریق New Table ساخته ایم را داشته باشیم کافیست بر روی نام جدول کلید سمت راست Mouse را زده و مسیر زیر انتخاب شود:

Script Table as > CREAT to > New Query Editor Window

پس از آن کد برنامه ساختن جدول مورد نظر تولید شده و قابل استفاده و ذخیره سازی است.

۳.۳. محدود کننده های فیلد (Constraint):

باید برای برخی از فیلدها بر اساس نیاز تعاریف و محدود کننده های خاص در نظر گرفته شود که در این بخش به چند مورد از آنها اشاره میگردد:

۳.۳.۱. Primary Key Constraint:

یکی از مهمترین محدودیت‌ها، تعریف فیلد کلید است. اینگونه فیلدها باید غیر تکراری بوده و معمولاً برای کدگذاری و ارتباط بین فایلها و یا جستجو مورد استفاده قرار میگیرد. برای تعریف یکی از فیلدها بعنوان کلید، باید در زمان ایجاد جدول و بر روی نام فیلد کلید سمت راست Mouse زده شده و سپس ردیف Set Primary Key انتخاب شود پس از آن علامت کلید در کنار نام فیلد ظاهر خواهد شد.

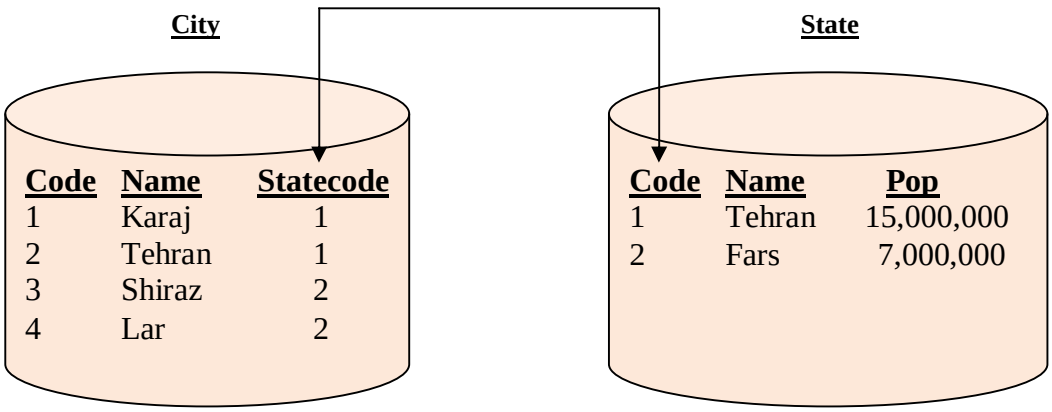
ضمناً این امکان وجود دارد که همزمان دو فیلد بعنوان کلید تعریف شود. برای اینکار باید روی فیلد دوم کلید Ctrl را گرفته و سپس با زدن کلید سمت راست Mouse، ردیف Set Primary Key انتخاب گردد. در اینصورت علامت کلید در کنار فیلد دوم نیز ظاهر میشود.

ضمناً باید توجه داشت که نباید فیلد کلید Null باشد لذا نباید ستون Allow Nulls برای این فیلد علامت زده شود.

۳.۳.۲. Foreign Key Constraint:

در زمانیکه برای برخی از عناوین مانند نام شهرها از کد استفاده میشود جهت ارتباط جدول اصلی با جدول شرح کد از این نوع کلید استفاده میشود.

بعنوان مثال دو جدول به شرح زیر در اختیار داریم. جدول City شامل کد و نام شهرها و کد استان مربوطه بوده و جدول State شامل کد، نام و جمعیت استانهای کشور میباشد. جدول City (بعنوان جدول فرزند)از طریق کد استان (Statecode) با جدول State (بعنوان جدول پدر) ارتباط دارد.



فیلدهای کلید خارجی دارای سه نوع محدودیت میباشد:

- حق حذف رکورد از جدول پدر وجود ندارد در حالیکه از آن کد در جدول فرزند سابقه وجود داشته باشد.
- درصورتی امکان اضافه کردن رکورد در جدول فرزند وجود دارد که کد مربوطه در جدول پدر تعریف شده باشد.
- حق تغییر کد اولیه در جدول پدر وجود ندارد درحالیکه از این کد در جدول فرزند سابقه وجود دارد.

برای تعریف کلید خارجی، در ابتدا باید در زمان ساختن جدول State (جدول پدر)، فیلد Code بعنوان کلید اولیه (Primary Key) تعریف گردد. سپس جدول City (جدول فرزند) بگونه ای ساخته شود که نوع فیلد Statecode دقیقاً مشابه نوع فیلد Code از جدول State باشد. پس از ساخته و ذخیره شدن دو جدول، بر روی جدول City رفته و بر روی ردیف Key کلید سمت راست Mouse زده شده و ردیف New Foreign Key... انتخاب شود. در این حالت صفحه تعریف مشخصات کلید خارجی ظاهر خواهد شد.

در صفحه Foreign Key Relationship باید ردیف Tables and Column Specification انتخاب گردد در اینصورت صفحه Tables and Columns نشان داده خواهد شد. سپس باید در این صفحه نام جدول State (پدر) و فیلد Code از این جدول در ستون Primary Key Table و نام جدول City (فرزند) و فیلد Statecode در ستون Foreign Key Table نوشته شود.

چنانچه بخواهیم کلید خارجی دارای مشخصات و ویژگیهای خاصی باشد باید از سایر ردیفهای جدول Foreign Key Relationship استفاده شود. این مشخصات به اختصار توضیح داده خواهد شد.

اگر Enforce for Replication برابرYes باشدتمام قوانین و ویژگیهای فوق در زمان Replication نیز رعایت خواهد شد. مثلاً در زمان انتقال فایل City از یک سرور A به سرور B، چنانچه یکی از رکوردهای City سرور A دارای کد استانی باشد که در فایل State سرور B تعریف نشده باشد این رکورد در فایل City سرور B اضافه نخواهد شد. ولی اگر این ردیف No باشد از این محدودیت کلید خارجی چشم پوشی شده و رکورد مذکور اضافه میگردد.

چناچه ردیف Enforce Foreign Key Constraint برابر Yes باشد محدودیت جلوگیری از اضافه شدن رکورد به فایل فرزند، در صورتی که سابقه ای از کد مربوطه در فایل پدر نباشد، رعایت میگردد در صورت No بودن از این محدودیت نیز چشم پوشی خواهد شد.

در ردیف Insert & Update Specification میتوان قوانین و محدودیتهای اضافه، حذف و بروزرسانی کردن کلید خارجی را تغییر داد. بعنوان مثال ردیف Delete Rules دارای چهار حالت زیر است:

- اگر No Action انتخاب شود اجازه حذف رکورد پدر را، در صورت وجود سابقه در جدول فرزند نمیدهد.
- اگر Cascade انتخاب شود اجازه حذف رکورد پدر داده میشود ولی تمامی رکوردهای فرزند مرتبط با کد حذف شده پدر نیز حذف خواهند شد.
- اگر Set Null برگزیده شود اجازه حذف رکورد پدر داده میشود ولیکن فیلد کد حذف شده پدر در رکوردهای مرتبط در فایل فرزند برابر Null خواهند شد.(به شرطی که این فیلد دارای شرط Allow Null باشد)
- اگر Set Default انتخاب شود اجازه حذف رکورد پدر داده میشود ولیکن فیلد کد حذف شده پدر در رکوردهای مرتبط در فایل فرزند برابر مقدار پیش فرض (Default) خواهد شد.(به شرطی که برای این فیلد دارای مقدار Default تعیین شده باشد)

۳.۳.۳. Check Constraint:

چنانچه بخواهیم برای یک فیلد محدوده مقداری مشخصی تعریف کنیم میتوان از این محدودیت استفاده کرد. محدوده فیلد میتواند در برنامه فرم ورود اطلاعات تعریف شود ولیکن بهتر است در زمان تعریف جدول و بروش زیر مشخص شود در صورت مهم بودن رعایت محدوده فیلد، و تعریف آن در داخل فرمها، ممکن است تعریف محدوده در برخی فرمها فراموش شود و همین امر باعث ورود اطلاعات غیر مجاز شود. یا تغییر محدوده مستلزم تغییر برنامه در تمامی فرمهاست (البته در زمان طراحی صفحات فرم ورود اطلاعات سیستمهای تحت Web بهتر است محدوده فیلد در فرمها نیز تعریف شود تا از انتقال اطلاعات غلط به دیتابیس و عودت داده شدن آن که باعث کاهش سرعت میگردد جلوگیری شود)

برای تعریف محدوده فیلد باید روی نام جدول کلیک کرده و ردیف Constraints انتخاب شده و سپس بکمک Mouse ردیف New Constraint فعال شود. در اینصورت جدول Check Constraint ظاهر خواهد شد. پس از آن باید دستور محدودیت مورد نظر در قسمت Expression تایپ گردد. مثلاً چنانچه بخواهیم فیلد Per بین 0 تا 100 باشد دستور Per>=0 and Per<=100 نوشته خواهد شد.

۳.۳.۴. Unique Constraint:

برخی فیلدها کلید نیستند ولیکن باید غیر تکراری باشند مثلاً کد ملی یا آدرس پست الکترونیکی از این نوع میباشند. این نوع فیلدها را فیلدهای غیر تکراری یا Unique نامند. از SQL 7 به بعد بجای این نوع فیلد از Unique Index استفاده شده است که در بخشهای بعدی توضیح داده خواهد شد.

۳.۳.۵. Default Constraint:

تعریف مقدار پیش فرض یا Default بعنوان مقدار اولیه نیز یک نوع محدودیت است که در زمان تعریف و ایجاد جدول به آن اشاره شد.

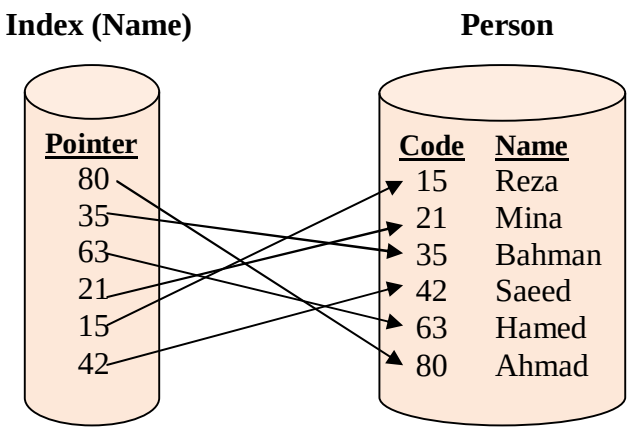
۳.۳.۶. Nullability Constraint:

این محدودیت نیز که اجازه دادن یا ندادن مقدار Null را برای یک فیلد تعیین می کند در زمان تعریف یا ایجاد جدول مورد بحث قرار گرفت.

۴. فهرست (Index):

۴.۱. تعریف فهرست :

با توجه به اینکه جداول را تنها میتوان بر اساس یک فیلد یا ستون مرتب نمود ولیکن معمولاً برای جستجو یا گزارشگیری لازم است لیست مرتب شده جدول را بر اساس فیلدهای مختلف داشته باشیم لذا از روش Indexing استفاده میشود. به مثال زیر توجه نمایید در این مثال فایل اصلی بر اساس Code مرتب شده است ولیکن یک فایل Index نیز جهت نمایش صعودی جدول بر اساس Name ساخته شده است:



در زمان تعریف Index باید به تعاریف زیر توجه شود:

۴.۱.۱. Clustered Index:

چنانچه داده در داخل جدول اصلی مرتب شود آنرا Clustered Index نامند. بنابراین برای هر جدول فقط یک Clustered Index داریم ولیکن تعداد None Clustered Index برای هر جدول برابر ۲۴۹ جدول میباشد.

۴.۱.۲. Ascending / Descending:

این حالت نوع صعودی و نزولی بودن ترتیب را مشخص می کند. اگر نوع ترتیب مشخص نشود حالت صعودی یا Ascending در نظر گرفته خواهد شد.

۴.۱.۳. Unique:

یکی دیگر از حالات فهرست، غیر تکراری بودن فیلد فهرست میباشد.

۴.۱.۴. Singlecolumn / Multicolumn Index:

چنانچه جدول بر اساس یک فیلد مرتب شود فهرست را Singlecolumn Index نامند و اگر بر اساس چندین فیلد مرتب شود فهرست را Multicolumn Index گویند.

۴.۲. ساخت ایندکس (Index):

برای ساخت ایندکس باید بر روی نام جدول مورد نظر، ردیف Index را انتخاب کرده و سپس به‌کمک Mouse New Index فعال شود. در اینصورت یک صفحه جهت ساخت ایندکس ظاهر میشود که شامل بخشهای مختلف است:

۴.۲.۱. General:

باید در این صفحه مشخصات ایندکس ذکر شود. سه نوع ایندکس یا فهرست قابل تعریف کردن است:

- Clustered: این فهرست بر روی فایل اصلی ساخته شده و معمولاً بر اساس Primary Key می‌باشد.
- Nonclustered: معمولاً ایندکسها از این نوع بوده و برای هر ایندکس یک فایل جداگانه ایجاد می‌گردد.
- Primary XML: چنانچه این نوع ایندکس انتخاب شود جستجو در فایلهای xml سریعتر انجام میشود.

پس از آن می‌توان غیرتکراری بودن ایندکس را با فعال کردن کلید Unique تعریف نمود. در ادامه با زدن کلید Add و انتخاب فیلد یا فیلدهای ایندکس، ترتیب ایندکس مشخص میشود. این موارد در بخش General از صفحه New Index است.

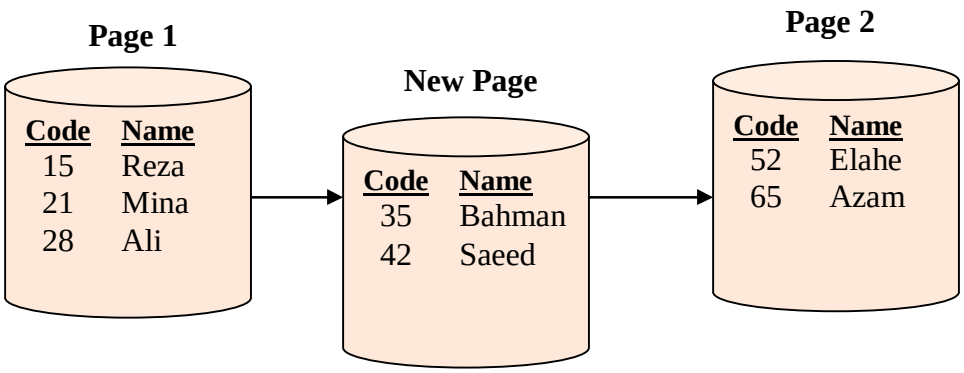
۴.۲.۲. Option:

سایر ویژگیهای ایندکس در قسمت Option تعریف میگردد. در این قسمت موارد زیر وجود دارد:

- Ignore Duplicate Values: چنانچه فایل از نوع Unique باشد میتوان این امکان را فعال کرد. فرض کنید بخواهیم یک‌سری رکورد را به یک فایل از نوع Unique اضافه کنیم سیستم با مشاهده اولین رکورد تکراری عملیات را متوقف و از اضافه کردن الباقی رکوردها خودداری می‌کند. ولی اگر این ویژگی فعال شود از ثبت رکورد تکراری جلوگیری میشود ولیکن سایر رکوردهای غیر تکراری به فایل اضافه خواهد شد.
- Automatically Recomputed Statistics: هر دستور که برایSQL ارسال میشود یک نرم افزار بنام Optimizer روش اجرای دستور را با توجه به آمار و اطلاعات موجود تحلیل و بررسی و سپس یک Execution Plan برای اجرای دستور تهیه کرده و برای SQL ارسال مینماید. چنانچه این ویژگی فعال شود SQL در زمان ساخت ایندکس، آمار و اطلاعات مورد نیاز Optimizer را خواهد ساخت در غیر اینصورت باید بصورت دستی و از طریق ردیف Statistics در ذیل نام فایل جدول، آمار و اطلاعات مورد نیاز ساخته شود.
- Use row lock when accessing the index: در زمان ساخت ایندکس تمام فایل قفل شده و هیچ کاربری نمیتواند تا پایان عمل ایندکس سازی با فایل کار کند. با فعال کردن این ردیف تنها همان رکورد که در حال ساخت ایندکس است قفل شده و کاربران اجازه کار با سایر رکوردها را دارند.
- Use page lock when accessing the index: مانند قسمت قبل اگر این ویژگی فعال شود تنها همان Page که ایندکس در حال کار با آن است قفل شده و کاربران میتوانند با سایر Page ها کار کنند.
- Store intermediate sort results in tempdb: اگر این ویژگی فعال شود عملیات میانی در فایل دیتابیس Tempdb انجام شده و درنتیجه باعث افزایش سرعت خواهد شد. در غیر اینصورت عملیات در دیتابیس اصلی انجام خواهد شد. لازم به توضیح است که در زمان ساخت یک دیتابیس، عملاً چهار فایل دیتابیس به شرح زیر ساخته خواهد شد:

- Master: اطلاعات اصلی دیتابیس در این فایل قرار خواهد گرفت.
- MSdB: تمام اطلاعات مربوط به سرویسهای SQL بصورت یک Job در این فایل قرار میگیرد.
- Model: در این فایل، یک دیتابیس نمونه جهت ساختن دیتابیس های جدید قرار میگیرد.
- Tempdb: این فایل جهت نگهداری موقت عملیات میانی استفاده میگردد.

- Set fill factor: این فاکتور میزان درصد اشغال فضای هر Page از اطلاعات ایندکس را مشخص میکند. مثلاً اگر ۱۰۰٪ باشد تمام فضای Page ها را اشغال کرده و در انتهای Page ها فضای خالی درنظر نمی‌گیرد. چنانچه این فاکتور فعال نشده و میزان درصد فضای اشغال شده مشخص نگردد SQL خودش میزان اشغال را تخمین زده و اعمال خواهد کرد. تعیین فضای خالی در پایان هر Page به‌منظور جلوگیری از Page Split های مکرر است. لازم به توضیح است که اطلاعات در Page های ۸۰۶۰ بایتی نگهداری میشود که ۸۰۰۰ بایت آن مربوط به اطلاعات و ۶۰ بایت مربوط اشاره گرهای SQL است. سپس این Page ها به‌وسیله اشاره گر به‌هم متصل میشوند. مثلاً اگر در مثال زیر بخواهیم کد ۲۸ را اضافه کنیم ولی Page اول دارای فضای خالی نباشد یک Page جدید ایجاد شده و سپس رکوردهای پس از رکورد ۲۸ به Page جدید منتقل شده و پس از آن رکورد ۲۸ در انتهای رکوردهای Page اول اضافه خواهد شد. روش اضافه شدن Page را Page Split نامند. ضمناً استفاده از تکنیک نگهداری اطلاعات در Page های کوچک و پیش بینی امکان Page Split، باعث جلوگیری جابجایی سایر رکوردها خواهد شد:



- Pad index: Page های اطلاعات به‌روش B_Tree نگهداری میشود در این روش برخی از Page ها به‌صورت آخرین Node میباشند که به آنها برگ گویند و معمولاً احتمال اضافه شدن Page به انتهای برگها بیشتر است. در حالت عادی فاکتور اشغال یا Set fill factor فقط برای فضای Page های برگ درنظر گرفته میشود ولیکن اگر Pad index فعال شود فاکتور اشغال برای تمامی Page ها اعمال خواهد شد.
- Allow online processing ...: اگر این فاکتور فعال شود همزمان با فهرست گذاری، امکان بروزرسانی، حذف و اضافه کردن اطلاعات برای کاربران وجود خواهد داشت.

- Set maximum degree of parallelism: چنانچه کامپیوتر بیش از یک CPU داشته باشد در این قسمت میتوان تعداد CPU های درگیر عملیات SQL راتعیین کرد. اگر مقدار صفر داده شود SQL خود نسبت به تعیین تعداد آن اقدام خواهد کرد.

۴.۲.۳. Include columns:

در صفحه New index ردیفInclude columns وجود دارد که با انتخاب آن صفحه ای برای تعریف کلیدهای خارج از محدوده ایندکس ظاهر خواهد شد. SQL نیز مانند تمامی نرم افزارهای دیتابیس دارای محدودیتهایی در طول و تعداد فیلدهای کلید میباشد ولیکن میتوان از این امکان برای تعریف فیلدهای با طول بیشتر یا تعداد فیلدهای بیشتر در ترکیب کلید استفاده کرد. لازم به توضیح است ماکزیمم طول کلید ۹۰۰ بایت و حداکثر تعداد فیلدهای کلید ۱۶ فیلد میباشد.

مثلاً اگر بخواهیم کلید ایندکس بر اساس سه فیلد A، B و C به شرح زیر باشد، قطعاً طول کلید بیش از ۹۰۰ بایت خواهد شد. بنابراین لازم است فیلدهای A و B در لیست فیلدهای ایندکس تعریف شده و فیلد C در Include columns تعیین گردد:

- A char(300)
- B char(300)
- C char(301)

۴.۲.۴. Storage:

چنانچه بخواهیم ایندکس و فایل اصلی در دو گروه جداگانه ذخیره شوند از این امکان استفاده خواهد شد. یکی از کاربردهای آن در زمان Backup گیری است و یا هنگامی که چند CPU داریم میتوان CPU پردازش فایل اصلی با CPU ساخت ایندکس متفاوت باشد. ضمناً اگر هارددیسک پارتیشن بندی شده باشد در این قسمت ایندکس میتواند جهت تعیین محل پارتیشن مورد استفاده قرار گیرد.

۴.۲.۵. Fragmentation:

این صفحه اطلاعات کامل مربوط به نگهداری ایندکس شامل فضاهای اشغال شده یا Page های ایندکس را ارایه می‌دهد. این ردیف در زمان تغییر مشخصات ایندکس ظاهر خواهد شد.

۴.۲.۶. Extended Properties:

این صفحه برای یادداشت گذاری برنامه نویس پیش بینی شده و SQL هیچگونه عملیاتی بر روی محتوای این صفحه انجام نخواهد داد. ضمناً این ردیف نیز در زمان تغییر مشخصات ایندکس ظاهر خواهد شد.

۴.۲.۷. سایر موارد:

ضمناً برخی از عملیات، مربوط به پس از ساختن ایندکس میباشد. برای آشنایی با آنها بر روی نام ایندکس کلید سمت راست Mouse را خواهیم زد. در اینصورت موارد زیر مشاهده خواهد شد:

- Rebuild: این قسمت برای ساخت مجدد ایندکس و حذف فضاهای خالی و سازماندهی مجدد ایندکس به‌کار می‌رود. استفاده از این امکان در هر چند روز توصیه میگردد.
- Reorganize: این تابع ایندکس را مجدداً نمیسازد ولی تا حد امکان فضاهای خالی را از بین برده و ایندکس را نسبتاً سازماندهی میکند. این روش نسبت به روش قبل، از سرعت بسیار بالاتری برخوردار است.

- Disable: این قسمت برای غیرفعال کردن ایندکس میباشد. چنانچه یک ایندکس Nonclustered غیر فعال شود تمام فضای اختصاص یافته به ایندکس حذف شده و در زمان ساخت مجدد یا Rebuild آن، ایجاد خواهد شد. ولی اگر یک ایندکس Clustered غیر فعال شود فضای فایل حذف نمیشود اما پس از آن دیگر ترتیب رکوردها رعایت نخواهد شد. مثلاً اگر بخواهیم یک میلیون رکورد به یک دیتابیس که بعنوان مثال دارای ۵۰ فهرست است به یکباره اضافه کنیم بهتر است اول تمام فهرستها غیر فعال شده و پس از بروزرسانی دیتابیس، ایندکسها مجدداً ساخته شود.

۴.۲.۸. Designing an Index:

برای اینکه بتوانیم ایندکسهای ساخته شده را تحلیل کرده و ایندکس خوبی بسازیم باید با دستورات ساختن ایندکس و عملیات جستجو و جایگزینی بر اساس ایندکسها آشنا باشیم. در این قسمت به چند نمونه دستور ارسالی برای ایندکس اشاره می‌کنیم:

- Where f1=12

  این دستور در داخل ایندکس، رکوردهایی که فیلد f1 آنها برابر ۱۲ است را جستجو می کند.
- Where f2 between 12 and 15

  این دستور در داخل ایندکس، رکوردهایی که فیلد f2 آنها بین ۱۲ و ۱۵ است را جستجو می کند.
- Having f3=12

  این دستور بررسی میکند آیا در داخل ایندکس رکوردی که فیلد f3 آن برابر ۱۲ است وجود دارد؟
- Order by f4

  این دستور بررسی میکند آیا فهرستی وجود دارد که فیلد کلید آن f4 باشد؟
- Group by f5

  این دستور ایندکس را بر اساس فیلد f5 گروه بندی میکند.
- T1 inner join T2 on T1.f6=T2.f7

در صورتیکه فیلدهای f6 وf7 دو جدول T1و T2 با هم برابر باشد، این دستور دو جدول فوق را با هم ادغام می‌کند.

۴.۲.۹. OLTP & OLAP:

معمولاً دو نوع سیستم داریم:

- OLTP (OnLine Transactional Processing)
- OLAP (OnLine Analytical Processing)

اگر عملیات پردازش از قبیل اضافه کردن، حذف کردن و بروزرسانی زیاد باشد از نوع OLTP خواهد بود. در این نوع سیستمها هرچه تعداد ایندکسها بیشتر باشد کارایی و سرعت سیستم کاهش خواهد یافت. ولیکن اگر عملیات گزارشگیری و تحلیلی در یک سیستم بیشتر باشد از نوع OLAP بوده و افزایش تعداد ایندکسها در این نوع سیستمها باعث افزایش سرعت و کارایی سیستم خواهد شد. در سیستم OLAP بر خلاف سیستم قبل، عملیات گزارشگیری و تحلیلی بیشتر است و افزایش تعداد ایندکسها در این نوع سیستمها باعث افزایش سرعت و کارایی آن می‌گردد.

برخی از سیستمها نیز ترکیبی بوده و باید در انتخاب نوع و تعداد ایندکسها تعادل برقرار شود.

۵. دستور Select:

فرمت کلی دستور Select به شکل زیر است:

```
Select field_1 [, field_n] from Table_1
    [Inner [or Left or Right or full or Cross] Join on Table_2]
    [Where condition]
    [Group by field_1 [, field_n]]
    [Having condition]
    [Order by field_1 [, field_n]]
```

به مثال صفحه بعد توجه کنید. این دیتابیس شامل ۸ جدول مرتبط با یکدیگر میباشد. مثالهای این بخش مرتبط با این دیتابیس میباشد.

1. `Select * from Authors`

   با اجرای این دستور تمام فیلدها و رکوردهای جدول نویسندگان نشان داده میشود.

2. `Select au_lname,au_fname from Authors`

   این دستور نام و نام خانوادگی نویسندگان را نشان خواهد داد.

3. `Select Authors.au_lname, Authors.au_fname from Authors`

   نتیجه این دستور نیز مانند دستور فوق است ولیکن در برخی موارد که نام فیلد در جداول مختلف مشابه است باید نام فیلد با نام جدول تواماً ذکر شود.

4. `Select au_lname as first,au_fname as last from authors`

   در این مثال نام و نام خانوادگی نویسندگان با عنوان first و last نمایش داده خواهد شد.

5. `Select k.au_lname,k.au_fname from Authors as k`

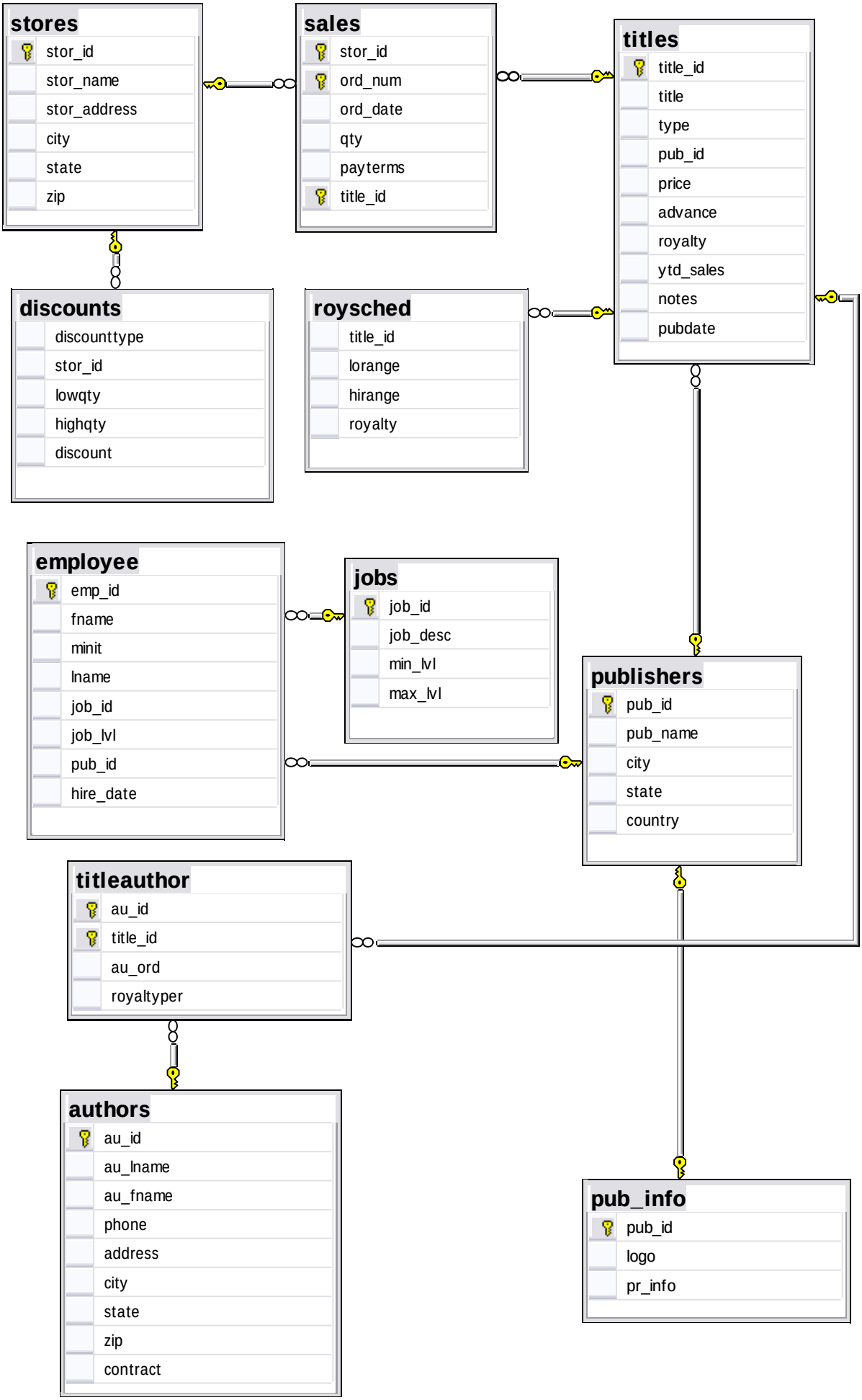
   در این مثال نام جدول نویسندگان با عنوان k شناخته شده و نام و نام خانودگی نویسندگان نشان داده میشود.

6. `Select * from Titles where price>20`

   این دستور رکوردهایی که قیمت آنها بیشتر از ۲۰ دلار است را نشان میدهد.

7. `Select * from Titles where price>10 and type='Business'`

   برای نشان دادن مشخصات کتابهایی که قیمت آنها بیش از ۱۰ دلار و نوع آنها تجاری است، از این دستور استفاده میگردد.


stores
stor_id
stor_name
stor_address
city
state
zip
sales
stor_id
ord_num
ord_date
qty
payterms
title_id
titles
title_id
title
type
pub_id
price
advance
royalty
ytd_sales
notes
pubdate
discounts
discounttype
stor_id
lowqty
highqty
discount
roysched
title_id
lorange
hirange
royalty
employee
emp_id
fname
minit
lname
job_id
job_lvl
pub_id
hire_date
jobs
job_id
job_desc
min_lvl
max_lvl
publishers
pub_id
pub_name
city
state
country
titleauthor
au_id
title_id
au_ord
royaltyper
authors
au_id
au_lname
au_fname
phone
address
city
state
zip
contract
pub_info
pub_id
logo
pr_info

8. `Select * from Titles where price between 10 and 15`

این مثال برای نمایش مشخصات کتابهایی است که قیمت آنها برابر یا بیشتر از ۱۰ دلار و برابر یا کمتر از ۱۵ دلار می‌باشد.

9. `Select * from Titles order by price`

این دستور مشخصات کتابها را به ترتیب قیمت به صورت صعودی نشان میدهد.

10. `Select * from Titles order by type desc, price desc`

در این مثال مشخصات کتابها به ترتیب نزولی نوع کتاب و سپس برای کتب با نوع برابر، به ترتیب نزولی قیمت نشان داده خواهد شد.

11. `Select * from Titles order by 2`

در این مثال مشخصات کتابها را به ترتیب صعودی فیلد دوم جدول نشان میدهد. نظر به اینکه ممکن است ستونهای جدول جابجا یا کم و زیاد شود لذا استفاده از این روش توصیه نمی گردد.

با توجه به دو جدول T1 و T2 زیر، چندین مثال برای دستور join به شرح ذیل ارایه شده است:

T1

f1	f2
1	a
2	b
3	c

T2

m1	m2	f1
1	p	1
2	r	2
3	q	2
4	s	5

12. `Select * from T1 inner join T2 on T1.f1=T2.f2`

این دستور دو جدول T1 و T2 را در یکدیگر ادغام مینماید و نتیجه به شکل زیر خواهد شد. در این مثال تنها رکوردهایی ایجاد میگردد که مقدار فیلد f1 در هر دو جدول سابقه داشته باشد:

T1 inner join T2

f1	f2	m1	m2	f1
1	a	1	p	1
2	b	2	r	2
2	b	3	q	2

13. `Select * from T1 left join T2 on T1.f1=T2.f2`

این دستور مشابه مثال قبل عمل میکند با این تفاوت که اگر رکوردی در فایل T1 وجود داشته ولی در T2 سابقه نداشته باشد این رکورد نیز نشان داده میشود ولیکن فیلدهای مشابه از فایل T2 آن Null خواهد بود.

T1 left join T2

f1	f2	m1	m2	f1
1	a	1	p	1
2	b	2	r	2
2	b	3	q	2
3	c	Null	Null	Null

14. `Select * from T1 Right join T2 on T1.f1=T2.f2`

این دستور نیز مانند مثال ۱۲ عمل شده ولیکن رکوردهایی از فایل T2 که در T1 سابقه ندارد نیز نشان داده شده و فیلدهای متناظر فایل T1 برابر Null خواهدبود.

T1 right join T2

f1	f2	m1	m2	f1
1	a	1	p	1
2	b	2	r	2
2	b	3	q	2
Null	Null	4	s	5

15. `Select * from T1 full join T2 on T1.f1=T2.f2`

این دستور تمام رکوردهای دو فایل T1 و T2 را در هم ادغام کرده و در صورتی که هر کدام از فایلها در فایل دیگر دارای سابقه نباشد فیلدها فایل دیگر را Null نشان میدهد.

T1 full join T2

f1	f2	m1	m2	f1
1	a	1	p	1
2	b	2	r	2
2	b	3	q	2
3	c	Null	Null	Null
Null	Null	4	s	5

16. `Select * from T1 cross join T2`

با اجرای این دستور بدون درنظر گرفتن ارتباط دو جدول از طریق فیلد f1، برای هریک از رکوردهای T1، تمام رکوردهای T2 نشان داده خواهد شد. در نتیجه جدول جدید دارای ۱۲ رکورد خواهد بود.

T1 cross join T2

f1	f2	m1	m2	f1
1	a	1	p	1
1	a	2	r	2
1	a	3	q	2
1	a	4	s	5
2	b	1	p	1
2	b	2	r	2
2	b	3	q	2
2	b	4	s	5
3	c	1	p	1
3	c	2	r	2
3	c	3	q	2
3	c	4	s	5

تمرین ۱:

دستوری بنویسید که لیست نام ناشر و عنوان کتابهای منتشر شده توسط وی را نشان دهد.

```
Select pub_name,title from publishers
    inner join titles on  publishers.pub_id=titles.pub_id
```

تمرین ۲:

دستوری بنویسید که نام و نام خانوادگی نویسنده و نام کتابهای منتشر شده توسط وی را نمایش دهد.

```
Select au_fname,au_fname,title from authors
     inner join titleauthor on  authors.au_id=titleauthor.au_id
     inner join titles on  titleauthor.title_id=titles.title_id
```

تمرین ۳:

دستوری بنویسید که نام و نام خانوادگی نویسنده و نام ناشرانی را لیست کند که کتابهای همان نویسنده را منتشر میکند:

```
Select au_fname,au_fname,pub_name from authors
     inner join titleauthor on  authors.au_id=titleauthor.au_id
     inner join titles on  titleauthor.title_id=titles.title_id
     inner join publishers on  titles.pub_id=publishers.pub_id
```

۶. توابع تجمیعی (Aggregate functions):

عملیات توابع زیر بر روی تعدادی از رکودهای فایل انجام گردیده و نتیجه بصورت عدد برگردانده میشود:

- Sum (Expression) — جمع فیلد مورد نظر را محاسبه میدهد
- Min (Expression) — حداقل مقدار فیلد را در رکوردها مشخص میکند
- Max (Expression) — حداکثر مقدار فیلد را در رکوردها مشخص میکند
- Avg (Expression) — میانگین مقدار فیلد را محاسبه میکند
- Count (Expression) — تعداد رکوردهایی که فیلد موردنظر Nullنباشد را محاسبه میکند

مثال ۱: دستور زیر، جمع قیمت را در جدول کتابها محاسبه مینماید.

```
Select sum(price) from titles
```

مثال ۲: دستور زیر، تعداد رکوردهایی از جدول کتابها را که فیلد قیمت آنها Null نباشد را محاسبه میکند.

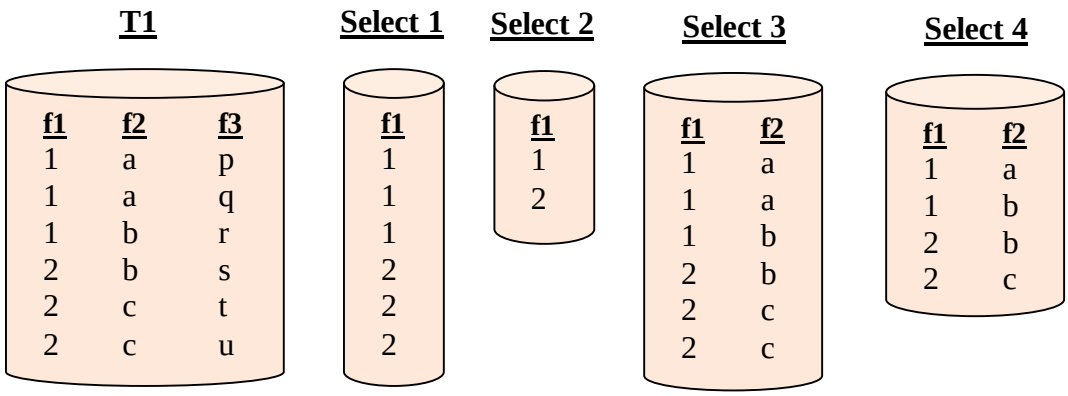
```
Select count(price) from titles
```

نکته: نظر به اینکه توابع فوق از جمله Count به رکوردهایی که فیلد مورد نظر Null باشد کاری ندارند لذا چنانچه بخواهید تعداد رکوردهای فایل را بدست آورید بهتر است از فیلد کلید اصلی یا همان Primary Key استفاده شود. و یا میتوان مانند دستور زیر از فیلد * استفاده کرد:

```
Select count(*) from authors
```

مثال ۳: فرض کنید جدول T1 را به شکل زیر داشته باشیم. با استفاده از عبارت Distinct تنها رکوردهایی که فیلد مورد نظر غیر تکراری باشد را در نظر خواهد گرفت:

```
1. Select f1              from T1
2. Select Distinct f1     from T1
3. Select f1,f2           from T1
4. Select Distinct f1,f2  from T1
```



مثال ۴: چند تا از نویسندگان کتاب دارند؟

```
Select count(Distinct au_id) from titleauthor
```

مثال ۵: با توجه به جدول T1 زیر، جدول را بر اساس فیلد f1 گروه بندی کرده و جمع هر گروه را نمایش دهید:

```
Select f1,sum(f2) as tp from T1 group by f1
```

T1

f1	f2
1	10
1	20
1	30
2	40
2	50
3	60

f1	tp
1	60
1	90
1	60

مثال ۶: برنامه ای بنویسید که کد کتاب و تعداد فروش کتاب را لیست کند.

```
Select title_id,sum(qty) from sales group by title_id
```

نکته: تمام فیلدهایی که خارج از توابع تجمیعی در دستور Select میباشند (مانند Title_id در مثال فوق) باید حتماً در قسمت group by ذکر شوند اگرچه عکس آن ضروری نیست.

مثال ۷: برنامه ای بنویسید که نام کتاب و تعداد فروش کتاب را لیست کند.

```
Select title,sum(qty) as tp from titles
   Inner join sales on titles.title_id=sales.title_id
   group by title
```

مثال ۸: برنامه ای بنویسید که نام کتاب و کل مبلغ فروش کتاب را لیست کند.

```
Select title,sum(qty*price) as tp from titles
    Inner join sales on titles.title_id=sales.title_id
    group by title
```

مثال ۹: برنامه ای بنویسید که نام ناشر و مبلغ فروش ناشر را لیست کند.

```
Select pub_name,sum(qty*price) as tp from publishers
    left join titles on publishers.pub_id=titles.pub_id
    left join sales on titles.title_id=sales.title_id
    group by pub_name
```

مثال ۱۰: برنامه ای بنویسید که نام خانوادگی نویسنده و تعداد کتابهایی که نوشته است را لیست کند.

```
Select au_lname,count(title_id) as tc from authors
    inner join titleauthor on authors.au_id=titleauthor.au_id
    group by au_lname,authors.au_id
```

or

```
Select au_lname,k.tc from authors inner join
    (Select au_id,count(title_id) as tc from titleauthor
              group by  au_id) k
    on authors.au_id=k.au_id
```

نکته: به بخش داخل پرانتز فوق که بنام k نامیده شده است Derived Query (گزارشگیری مشتق) گفته میشود.

مثال ۱۱: برنامه ای بنویسید که کل مبلغ فروش کتاب را نمایش دهد.

```
Select sum(qty*price) from titles
      inner join sales on titles.title_id=sales.title_id
```

مثال ۱۲: با توجه به اینکه ممکن است هر کتاب دارای چند نویسنده باشد و با فرض اینکه مبلغ حاصل از فروش کتابها به نسبت مساوی بین نویسندگان تقسیم میگردد، برنامه ای بنویسید که کد نویسنده و مبلغ سهم هر نویسنده از فروش کتابها را لیست کند.

```
Select au_id,sum(qty*price/tc) as tp from titleauthor
      inner join titles on titleauthor.title_id= titles.title_id
      inner join (select title_id,count(au_id) as tc
                    From  titleauthor group by title_id) k
           on titles.title_id=k.title_id
      inner join sales on titles.title_id=sales.title_id
      group by au_id
```

مثال ۱۳: برنامه ای بنویسید که نام کتابهای از نوع تجاری و مبلغ فروش کتبی را نمایش دهد که بیش از ۲۰۰ دلار از این کتابها فروخته شده است.

```
Select title, sum (qty*price) as tp from titles
        Inner join sales on titles.title_id=sales.title_id
        Where type='business'
        Group by title
        Having sum (qty*price)>200
```

نکته: اگر بخواهیم مقدار خروجی توابع تجمیعی (aggregate functions) کنترل شده و در دستورهای شرطی استفاده شود باید در having قرار گیرد ولی اگر بر روی فیلد شرط بگذاریم باید از where استفاده شود.

مثال ۱۴: برنامه ای بنویسید تا نویسندگانی را پیدا کند که بیش از یک کتاب نوشته اند. سپس نام خانوادگی و نام نویسنده و تعداد کتب آنها را نمایش دهید.

```
Select au_lname, au_fname, count (title_id) from authors
        Inner join titleauthor
        on authors.au_id= titleauthor.au_id
        Group by au_lname, au_fname
        Having count (title_id)>1
```

مثال ۱۵: برنامه ای بنویسید که نام گران قیمت ترین کتاب را مشخص کند.

```
Select title, price from titles
       Where price= (select max (price) from titles)
```

نکته: به دستور داخل پرانتز اصطلاحاً Subquery گویند.

مثال ۱۵: برنامه ای بنویسید که ۱۰ رکورد اول جدول کتب را نمایش دهد.

```
Select top 10 * from titles
```

نکته: درصورت استفاده همزمان از Distinct و Top n، باید Distinct قبل از Top n قرار گیرد.

مثال ۱۶: برنامه ای بنویسید که فایل کتب را بر اساس قیمت مرتب کرده و ۱۰ درصد از رکوردهای اول لیست را نمایش دهد.

```
Select top 10 percent * from titles order by price
```

مثال ۱۷: مثال ۱۴ را با استفاده از Top n حل کنید.

```
Select top 1 title, price from titles order by price desc
```

نکته: اشکال این روش در زمانی است که بیش از یک رکورد دارای قیمت حداکثر باشد. دراینصورت باید از with ties استفاده کرد. چنانچه از این امکان استفاده شود تمام رکوردهای پس از رکورد آخر که فیلد سورت شده برابر مقدار آخرین رکورد باشد را نیز نمایش میدهد. بنابراین باید در زمان استفاده از with ties حتماً از order by هم استفاده شده باشد. با این توضیح دستور فوق به شکل زیر اصلاح میگردد.

```
Select top 1 with ties title, price from titles
      order by price desc
```

مثال ۱۸: پنجمین گران قیمت ترین کتاب را پیدا کنید.

```
Select title, price from titles
    Where price=
    (Select min (price)
        From (Select distinct top 5 price
                  from titles order by price desc) k)
```

مثال ۱۹: برنامه ای بنویسید که نام و قیمت کتاب، قیمت میانگین کل کتابها و اختلاف قیمت هر کتاب با قیمت میانگین را نمایش دهد.

```
Select title, price, ta, price-ta from titles
        Cross join (Select avg (price) as ta from titles) k
```

یا

```
Select title, price, (Select avg (price) from titles),
      Price-(Select avg (price) from titles) from titles
```

مثال ۲۰: برنامه ای بنویسید که تمام نویسندگان را براساس کد نویسنده مرتب کرده و یک شماره ردیف نیز در کنار لیست اضافه کند.

```
Select (select count (au_id) from authors
             where au_id<=k.au_id) as r,*
        From authors as k order by au_id
```

نکته: در SQL Server 2005 از دستور زیر نیز میتوان استفاده کرد. البته در صورت استفاده از این روش، باید حتماً براساس یک فیلد مرتب شود.

```
Select row_number () over (order by au_id) as r,* from authors
```

مثال ۲۱: برنامه ای بنویسید که تمام کتابها بر اساس کد کتاب مرتب شده و شماره ردیف براساس نوع کتاب زده شود.

```
Select row_number ()
   over (partition by [type] order by title_id) as r,*
   from titles
```

مثال ۲۲: فرض کنید جدول Studetnt را به شکل زیر داشته باشیم. برنامه ای بنویسید که شاگرد اول، شاگرد دوم و الی آخر، با ذکر اینکه شاگرد چندم است، را لیست کند. لازم به توضیح است اگر دو یا چند نفر مثلاً شاگرد دوم بودند باید برای تمام آنها رتبه دوم ذکر شود.

```
Select (select count (distinct score) from student
     Where score>=k.score) as Rank,*
     From student as k order by score desc
```

Student

Code	Score
10	110
20	120
25	90
27	110
40	80
52	115

Rank	Code	Score
1	20	120
2	52	115
3	10	110
3	27	110
4	25	90
5	40	80

نکته: در S.S 2005 دستوری جدیدی داریم که همان عمل رده بندی را مانند مثال قبل انجام میدهد.

```
Select rank () over (order by score desc) as Rank,* from student
```

Rank	Code	Score
1	20	120
2	52	115
3	10	110
3	27	110
5	25	90
6	40	80

چنانچه ملاحظه میگردد نتیجه این دستور تفاوت مختصری با نتیجه مثال فوق دارد بدین ترتیب که رده ردیف پنجم برابر 5 شده است درحالی که در مثال قبل رده ردیف پنجم 4 بوده است. ضمناً اگر بخواهیم نتیجه rank() دقیقاً همانند نتیجه مثال قبل شود باید از دستور زیر استفاده شود:

```
Select dense_rank () over (order by score desc) as Rank,*
    from student
```

نکته: با استفاده از دستور IN میتوان subquery های جدید نوشت. دستور IN کنترل میکند آیا مقدار فیلد در یکسری مقادیر وجود دارد یا خیر؟ اگر وجود داشته باشد جواب True خواهد بود و در غیر اینصورت جواب False میباشد. معمولاً دستور IN در where ظاهر میشود. ضمناً دستور Not IN عکس دستور IN عمل میکند. به چند نمونه از فرمت این دستور توجه فرمایید:

```
Select .......    Where field  IN (2, 3, and 4)
Select .......    Where field  Not IN (2, 3, and 4)
Select .......    Where field  IN (select onefield from ..........)
```

مثال ۲۳: برنامه ای بنویسید که تمام نویسندگان همولایتی با نویسنده ای بنام Livia و فامیل Karsen را نمایش دهد.

```
Select * from authors
     where state IN (Select state from authors
            where au_fname='Livia' and au_lname='Karsen')
```

مثال ۲۴: برنامه ای بنویسید که تمام نویسندگانی را پیدا کند که کتاب ندارند.

```
Select * from authors where au_id Not IN
     (Select au_id from titleauthor)
```

مثال ۲۵: برنامه ای بنویسید تا ناشرینی را لیست کند که اصلاً کتاب تجاری منتشر نکرده اند.

```
Select * from publishers where pub_id Not IN
     (select pub_id from titles where type='Bisiness')
```

نکته: نتیجه دستور فوق بدین صورت است که نام ناشرینی که حتی کتاب منتشر نکرده اند را نیز لیست مینماید. ولیکن اگر یک ناشر یک کتاب تجاری و یک کتاب غیر تجاری منتشر کرده باشد را نشان نخواهد داد.
چنانچه از دستور زیر استفاده شود لیست تمام ناشرینی که کتاب غیر تجاری منتشر کرده اند را نشان خواهد داد که با صورت مسیله درخواستی متفاوت است. در اینصورت از ناشران فاقد کتاب صرفنظر شده و ناشرینی که علاوه بر کتاب تجاری، کتاب غیر تجاری نیز منتشر کرده اند را لیست میکند.

```
Select * from publishers where pub_id IN
     (select pub_id from titles where type<>'Bisiness')
```

مثال ۲۶: برنامه ای بنویسید تا ناشرینی را لیست کند که حتماً کتب منتشر کرده اند.

```
Select * from publishers where pub_id IN
     (Select pub_id from titles)
```

نکته: با استفاده از دستور Exists نیز میتوان subquery های جدید نوشت. با اجرای دستور Exists اگر رکوردی توسط Subquery Select داخل پرانتز برگردانده شود نتیجه True خواهد بود و در غیر اینصورت جواب False میباشد. معمولاً دستور Exists نیز در where ظاهر میشود. ضمناً دستور Not Exists عکس دستور Exists عمل می‌کند. فرمت این دستور به شکل زیر است:

```
Select .......    Where Exists (Select ...... from ...... where .........)
Select .......    Where Not Exists (Select ...... from ...... where .........)
```

مثال ۲۷: مثال ۲۵ را به‌کمک دستور Exists حل نمایید.

```
Select * from publishers
       where Not Exists(select * from titles where type='Bisiness'
                        and Publishers.pub_id=titles.pub_id)
```

نکته: هر دستوری-که با IN (یا Not IN) نوشته میشود را میتوان با Exists (یا Not Exists) نیز نوشت با این تفاوت که با دستور IN فقط میتوان برای یک فیلد کنترل گذاشت (مانند فیلد Pub_id در مثال مذکور) ولی با استفاده از Exists در آخر پرانتز و با استفاده از and های متوالی برای هر تعداد فیلد میتوان شرط و کنترل تعریف کرد.

مثال ۲۸: مثال ۲۴ را به‌کمک دستور Exists حل نمایید.

```
Select * from authors
     where Not Exists (Select * from titleauthor
                       where authors.au_id= titleauthor.au_id)
```

نکته: یکی دیگر از دستورات سودمند S.S 2005 دستور Case است که به دو صورت به شرح زیر قابل استفاده میباشد:

```
..... Case  field  when   value1        then Exp1
                   when   Value2        then Exp2
                   ...
                                        else   Expn        end

..... Case         when   Condition1 then Exp1
                   when   Condition2    then Exp2
                   ...
                                        else   Expn        end
```

مثال ۲۹: برنامه ای بنویسید که شرح کامل سه نوع کتاب را به‌همراه نام اختصاری آنها بنویسد.

```
Select title, type,
      case type when 'Trad_cook'    then 'Tradition cooking'
                when 'Mod_cook'     then 'Modern cooking'
                when 'Popular_comp' then 'Popular computing'
                                    else type end as newtype
      from titles
```

مثال ۳۰: برنامه ای بنویسید که نام و قیمت کتاب و قیمت جدید را که براساس ضابطه زیر محاسبه میشود را لیست کند.

اگر قیمت کتاب بیش از ۲۰ دلار بود را ۱۰٪ افزایش داده و درغیر اینصورت ۵٪ کاهش دهد.

```
Select title, price,
       case when price>20 then price*1.1
                          else price*.95 end as newprice
       from titles
```

مثال ۳۱: برنامه ای بنویسید که نام و قیمت کتاب و قیمت جدید را که براساس ضابطه زیر محاسبه میشود را لیست کند.

اگر ناشر کتاب اهل کالیفرنیا بود قیمت کتاب را ۱۰٪ افزایش داده و درغیر اینصورت ۵٪ کاهش دهد.

```
Select title, price, case when state='CA'then price*1.1
                          else price*.95 end as newprice
      from titles inner join publishers
                     on titles.pub_id= publishers.pub_id
```

راه حل دوم:

```
 Select title, price,
      case when pub_id IN (select pub_id from publishers
                           Where state='CA')
           then price*1.1
           else price*.95 end as newprice
      from titles
```

مثال ۳۲: برنامه ای بنویسید که نام و قیمت کتاب و قیمت جدید را که براساس ضابطه زیر محاسبه میشود را لیست کند.

اگر کتاب بیش از ۵۰۰ دلار فروش داشت را ۱۰٪ افزایش داده و درغیر اینصورت ۵٪ کاهش دهد.

```
Select title, price, case when sum (qty*price)>500
                          then price*1.1
                          else price*.95 end as newprice
 from titles inner join sales on titles.title_id=sales.title_id
      group by title,price
```

مثال ۳۳: برنامه ای بنویسید که مشخصات کتاب و قیمت جدید که براساس ضابطه زیر محاسبه میشود را لیست کند.

- اگر ناشر کتاب اهل کالیفرنیا بود قیمت کتاب را ۱۰٪ افزایش دهد.
- در غیر اینصورت اگر کتاب بیش از یک نویسنده داشت قیمت کتاب را ۵٪ افزایش دهد.
- در غیر اینصورت اگر کتاب بیش از ۲۰۰ دلار فروش داشت قیمت کتاب را ۲٪ افزایش دهد.
- در غیر اینصورت قیمت کتاب را ۱٪ افزایش دهد.

```
Select title, price,
     case when pub_id  IN
          (select pub_id from publishers where state='CA')
          then price*1.1
else case when title_id IN
          (select title_id from titleauthor
           group by title_id having count(au_id)>1)
          then price*1.05
else case when title_id IN
          (select titles.title_id from titles
           inner join sales on titles.title_id=sales.title_id
           group by titles.title_id having count(price*qty)>200)
          then price*1.02
else  price*1.01 end end end as newprice
from titles
```

مثال ۳۴: برنامه ای بنویسید که مالیات بر فروش کتابها را با ضابطه زیر محاسبه کند (بصورت پلکانی):

- مالیات برابر ۰٪ فروش برای فروشهای مساوی یا کمتر از ۳۰۰ دلار
- مالیات برابر ۱۰٪ فروش برای فروشهای بین ۳۰۱ و ۵۰۰ دلار
- مالیات برابر ۲۰٪ فروش برای فروشهای بین ۵۰۱ و ۸۰۰ دلار
- مالیات برابر ۲۵٪ فروش برای فروشهای بیش از ۸۰۰ دلار

```
Select title, sum(price*qty),
     case when sum(price*qty)<301  then 0
          when sum(price*qty)<501  then (sum(price*qty)-300)*0.1
          when sum(price*qty)<801  then
                    (sum(price*qty)-500)*0.2+(500-300)*0.1
                              else  (sum(price*qty)-800)*0.25+
                                     (500-300)*0.1+(800-500)*0.2
          end as tax from titles
     inner join sales on titles.title_id=sales.title_id
     group by title
```

راه حل دوم:

```
Select title, tp,
 case when tp<301     then 0
      when tp<501     then (tp-300)*0.1
      when tp<801     then (tp-500)*0.2+(500-300)*0.1
           else  (tp-800)*0.25+(500-300)*0.1+(800-500)*0.2
      end as tax
from (select title,sum(price*qty) as tp from titles
      inner join sales on titles.title_id=sales.title_id
       group by title) k
```

نکته: در S.S 2005 دو نوع دستور CTE (Comman Table Expression) داریم:

- None Recursive CTE
- Recursive CTE

کاربرد این نوع دستورات در برنامه نویسی بسیار رایج و سودمند بوده و فرمت کلی دستورات CTE بصورت زیر است:

```
With
      cte_name1 (fields)        as (select statement)
     [,cte_name2 (fields)       as (select statement) ]
     [,cte_namen (fields)  as (select statement) ]
```

مثال ۳۵: مثال قبل را بکمک دستورات CTE حل نمایید. (این مثال از نوع Non Recursive CTE است.)

```
 With  k (title,tp)
      as (select title,sum(price*qty) as tp from titles
           inner join sales on titles.title_id=sales.title_id
           group by title)
 Select title, tp,case when tp<301  then 0
                       when tp<501  then (tp-300)*0.1
                       when tp<801  then (tp-500)*0.2+
                                         (500-300)*0.1
                 else  (tp-800)*0.25+ (500-300)*0.1+(800-500)*0.2
                          end as tax from k
```

مثال ۳۶: برنامه ای بنویسید که نام کتاب، مبلغ فروش و تعداد نویسنده کتاب را لیست کند.

```
 With Countofauthor(title_id,tc)
      As (Select title_id, count(au_id) as tc
           From titleauthor
           group by title_id),Amounofsale(title_id,tp)
           As (Select titles.title_id,sum(price*qty) as tp
           From titles inner join sales
           on titles.title_id=sales.title_id
           group by title,titles.title_id)
 Select title, tp,tc  From titles as T
      inner join countofauthor as c on t.title_id=c.title_id
      inner join Amounofsale  as a on t.title_id=a.title_id
```

مثال ۳۷: برنامه ای بنویسید که در ابتدا یک فایل شامل کد و نام کشورها/استانها/شهرها/روستاها و کد پدر آنها را ساخته و سپس مقداری اطلاعات در آن اضافه شده و پس از آن به کمک دستورات Recursive CTE علاوه بر فیلدهای فایل، مسیر پدر و اجداد آنها را نیز نمایش دهد.

```
Create Table geo (Code          int primary key,
                  Name          varchar(50),
                  Parentcode    int)
Insert into geo values(1,'Iran',    null)
Insert into geo values(2,'Iraq',    null)
Insert into geo values(3,'Tehran',  1)
Insert into geo values(4,'Karaj',   3)
Insert into geo values(5,'Baghdad', 2)
Insert into geo values(6,'Fars',    1)
Insert into geo values(7,'Laar',    6)
Insert into geo values(8,'Tehran',  3)
Insert into geo values(9,'Shiraz',  6)
With  tree (code, name, parentcode, strpath, lvl)
       as  (select code, name, parentcode,
           cast(name as varchar(max)) as strpath, 1 as lvl
           from geo where parentcode is null)
           union all
select g.code, g.name, t.strpath+'\'+g.name as strpath,
       t.lvl+1 as lvl
      from geo g inner join tree t on g.parentcode=t.code
```

با اجرای دستورات فوق جدول Geo و خروجی نهایی بصورت زیر خواهد بود.

Geo

Code	Name	Parentcode
1	Iran	Null
2	Iraq	Null
3	Tehran	1
4	Karaj	3
5	Baghdad	2
6	Fars	1
7	Laar	6
8	Tehran	3
9	Shiraz	6

Recursive CTE

Code	Name	Parentcode	Strpath	lvl
1	Iran	Null	Iran	1
2	Iraq	Null	Iraq	1
3	Tehran	1	Iran\Tehran	2
4	Karaj	3	Iran\Tehran\Karaj	3
5	Baghdad	2	Iraq\Baghdad	2
6	Fars	1	Iran\Fars	2
7	Laar	6	Iran\Fars\Laar	3
8	Tehran	3	Iran\Tehran\Tehran	3
9	Shiraz	6	Iran\Fars\Shiraz	3

۷. دستورات دستکاری جداول (Tables Manipulate Commands):

۷.۱. دستور union: این دستور دو جدول را، در صورتی که تعداد فیلدهای آن برابر و نوع فیلدها یکسان باشند، به هم می چسباند. ضمناً این دستور رکوردهای تکراری را حذف میکند. دستور union all نیز همانند دستور union عمل کرده ولیکن رکوردهای تکراری را نیز می آورد. به مثال زیر توجه کنید:

```
Create Table T1 (Code    int, Name varchar (50))
Create Table T2 (Code    int, Name varchar (50))
Insert   into   T1 values (1, 'a')
Insert   into   T1 values (2, 'b')
Insert   into   T1 values (3, 'c')
Insert   into   T2 values (1, 'a')
Insert   into   T2 values (2, 'b')
Insert   into   T2 values (3, 'd')
Select code, name from T1
Union
 Select code, name from T2
```

Union

Code	Name
1	a
2	b
3	c
3	d

Union all

Code	Name
1	a
2	b
3	c
1	a
2	b
3	c

۷.۲. دستورات interest و except: دستور intersect اشتراک و except تفاضل دو جدول را نشان میدهد. به مثالهای زیر توجه کنید:

```
Select * from T1
Intersect
Select * from T2
```

⇒

Code	Name
1	a
2	b

```
Select * from T1
Except
Select * from T2
```

⇒

Code	Name
3	c

```
Select * from T2
Except
Select * from T1
```

⇒

Code	Name
3	d

نکته: اگر مقدار T1-T2=0 و T2-T1=0 بود و یا حاصل عبارت (T1-T2)+(T2-T1)=0 بود آنگاه میتوان نتیجه گرفت که دو جدول T1 و T2 برابر میباشند. برنامه زیر برای تشخیص برابر بودن این دو جدول میباشد:

```
Select C1+C2
      From ((Select Count (*) as C1
                 From (Select Code, Name from T1
            Except
            Select Code, Name from T2) M
                ) H
          Cross join
            (Select Count (*) as C2
                From (Select Code, Name from T2
             Except
             Select Code, Name from T1) N
                ) J)
```

۷.۳. دستور pivot: این دستور برای نمایش محتوای یک جدول معمولی بصورت جدول دو بعدی میباشد. به مثال زیر توجه فرمایید:

```
Create Table Sal (Yr int, Qr int, Amt money)
Insert into Sal values (2007, 1, 11)
Insert into Sal values (2007, 2, 12)
Insert into Sal values (2007, 3, 13)
Insert into Sal values (2007, 4, 14)
Insert into Sal values (2008, 1, 21)
Insert into Sal values (2008, 2, 22)
Insert into Sal values (2008, 3, 23)
Insert into Sal values (2008, 4, 24)
Select Yr, [1], [2], [3], [4] from
     (Select Yr, Qr, Amt from Sal) K
      Pivot
     (Sum (Amt) for Qr IN ([1], [2], [3], [4])) M
```

Sal

Yr	Qr	Amt
2007	1	11
2007	2	12
2007	3	13
2007	4	14
2008	1	21
2008	2	22
2008	3	23
2008	4	24

Yr	1	2	3	4
2007	11.00	12.00	13.00	14.00
2008	21.00	22.00	23.00	24.00

۷.۴. دستور unpivot: عملکرد این دستور عکس دستور pivot بوده و برای نمایش محتوای یک جدول دو بعدی بصورت جدول معمولی میباشد. مثال زیر نتیجه این دستور را نشان میدهد:

در ابتدا جدول unpiv ساخته می شود.

```
SELECT yr,[1],[2],[3],[4]
into unpiv
FROM
     (SELECT yr,qr,amt FROM Sal) K
     Pivot
     (Sum(amt) For Qr IN ([1],[2],[3],[4]) ) M
```

پس از آن جدول unpiv با دستورات زیر بصورت حالت اولیه sal نمایش داده میشود:

```
SELECT Yr,Qr,Amt
FROM Unpiv
Unpivot
(amt For qr in ([1],[2],[3],[4])) J
```

۷.۵. دستورات cube و Rollup: این دستورات برای محاسبه جمع بر اساس فیلدهای مختلف استفاده میگردد. مثلاً فرض کنید جدول زیر را داشته باشیم. دستور cube در مثال زیر باعث میگردد که جمع Quantity برای color و item های مختلف نیز نمایش داده شود. ولیکن نتیجه دستور Rollup محاسبه جمع برای Color های مختلف بوده و بر روی Item که بلافاصله بعد از Group by قرار گرفته است جمع محاسبه نمیشود:

Inventory

Item	Color	Quantity
Table	Blue	124
Table	Red	223
Chair	Blue	101
Chair	Red	210

Cube

Item	Color	Quantity
Chair	Blue	101
Chair	Red	210
Chair	Null	311
Table	Blue	124
Table	Red	223
Table	Null	347
Null	Null	658
Null	Blue	225
Null	Red	433

Rollup

Item	Color	Quantity
Chair	Blue	101
Chair	Red	210
Chair	Null	311
Table	Blue	124
Table	Red	223
Table	Null	347
Null	Null	658

```
Select item,color,sum(quantity) as tp from inventory
     Group by item,color with Cube

Select item,color,sum(quantity) as tp from inventory
     Group by item,color with Rollup
```

۷.۶. دستور Insert: این دستور برای اضافه کردن یک یا چند رکورد به جدول استفاده شده و فرمت آن که به دو صورت تکی و جمعی است بصورت زیر میباشد:

```
Insert into Table_name [(Field1[,Fieldn])]  values(Value1 [,Valuen])

Insert into Table_name [(Field1[,Fieldn])]
        Select Field1 [,Fieldn] from Table_name2...
```

مثال: به چندین مثال از دستور Insert توجه فرمایید. فرض کنید جدول Test دارای سه فیلد Code و Name و Lastname باشد:

```
Insert into Test (Code,Name) values(1,'reza')

Insert into Test (Name,Code) values('hamid',2)

Insert into Test              values(3,'karim')
```

اگر نام فیلدها ذکر نشود مقادیر به ترتیب فیلدهای جدول اضافه خواهد شد ولی توصیه میشود حتماً نام فیلدها ذکر شود. در سه مثال فوق مقدار Lastname برابر Null خواهد بود.

```
Insert into Test              values(4,Null,'Saeidi')
```

در دومثال زیر اگر مقدار Default فیلد Name برابر TT تعریف شده باشد مقدار Name برابر TT میشود.

```
Insert into Test (Code,Lastname) values(5,'razavi')

Insert into Test            values(6,Default,'Rezaei')
```

نکته: چنانچه برای فیلد Code ویژگی Indentity تعریف شده باشد سیستم مقدار آنرا بصورت خودکار محاسبه خواهد کرد ولذا نباید در دستور Insert برای این فیلد مقدار تعریف شود.

مثال: در مثال زیر نیز فرض شده است که جدول Amountofsale با فیلدهای Title_id و Amount وجود داشته و می خواهیم جمع فروش هر کتاب را در این جدول اضافه کنیم:

```
insert into amountofsale (title_id,amount)
    select titles.title_id,sum(qty*price) as tp
            from titles inner join sales
                 on titles.title_id=sales.title_id
            group by titles.title_id
```

۷.۷. دستور Update: این دستور برای تغییر مقادیر فیلدهای یک جدول مورد استفاده قرار گرفته و فرمت آن بشکل زیر است:

```
update Table_name set Field1=Exp1[,Fieldn=Expn]
        [From Table_join]
        [where Condition]
```

مثال: در مثالهای زیر مقدار قیمت کتاب بدون شرط یا با شرط خاص ۱۰٪ افزایش می یابد:

```
update titles set price=price*1.01

update titles set price=price*1.01 where type='business'
```

تمرین: برنامه ای بنویسید که با یک دستور Update قیمت همه کتابها را اگر قیمت کتاب بالاتر از ۲۰ دلار بود ۱۰٪ افزایش و در غیر اینصورت ۵٪ کاهش دهد.

```
update titles set price= case when price>20 then price*1.01
                                            else price*0.95 end
```

تمرین: برنامه ای بنویسید که با یک دستور Update قیمت همه کتابهایی که ناشر آن کالیفرنیایی بود ۱۰٪ افزایش و در غیر اینصورت ۵٪ کاهش دهد.

```
update titles set price= case when pub_id IN
                 (select pub_id from publishers where state='ca')
                              then price*1.01
                              else price*0.95 end
```

راه حل دوم:

```
update titles set price= case when state='ca' then price*1.01
                                              else price*0.95 end
          inner join publishers on titles.pub_id=publishers.pub_id
```

تمرین: برنامه ای بنویسید که با یک دستور Update قیمت همه کتابهایی که بیشتر از ۵۰۰ دلار فروش داشته اند را ۱۰٪ افزایش و در غیر اینصورت ۵٪ کاهش دهد.

```
update titles set price=Newprice from titles inner join
  (select titles.title_id,
       case when sum(qty*price)>500 then price*1.1
                                    else price*.95 end as Newprice
       from titles inner join sales
       on titles.title_id=sales.title_id
       group by titles.title_id,price) k
  on titles.title_id=k.title_id
```

تمرین: برنامه ای بنویسید که با یک دستور Update قیمت کتابها را با توجه به ضابطه زیر بروزرسانی کند:

- اگر ناشر کتاب اهل کالیفرنیا بود قیمت کتاب را ۱۰٪ افزایش دهد.
- در غیر اینصورت اگر کتاب بیش از یک نویسنده داشت قیمت کتاب را ۵٪ افزایش دهد.
- در غیر اینصورت اگر کتاب بیش از ۲۰۰ دلار فروش داشت قیمت کتاب را ۲٪ افزایش دهد.
- در غیر اینصورت قیمت کتاب را ۱٪ افزایش دهد.

```
update titles set price=Newprice from titles inner join
  (Select title_id, price,
     case when pub_id  IN (select pub_id from publishers
           where state='CA')
           then price*1.1
else case when title_id IN (select title_id from titleauthor
           group by title_id having count(au_id)>1)
           then price*1.05
else case when title_id IN (select titles.title_id from titles
           inner join sales on titles.title_id=sales.title_id
           group by titles.title_id having count(price*qty)>200)
           then price*1.02
           else  price*1.01 end end end as newprice
           from titles) k
  on titles.title_id=k.title_id
```

راه حل دوم:

```
  with k (title_id,Newprice)
     as (Select title_id,
        case when pub_id  IN (select pub_id from publishers
                      where state='CA') then price*1.1
   else case when title_id IN (select title_id from titleauthor

       group by title_id having count(au_id)>1)then price*1.05
  else case when title_id IN (select titles.title_id from titles
```

```
                inner join sales on titles.title_id=sales.title_id
          group by titles.title_id having count(price*qty)>200)
                          then price*1.02
          else  price*1.01 end end end as newprice from titles)
     update titles set price=Newprice from titles inner join
                k on titles.title_id=k.title_id
```

۷.۸. دستور Delete: این دستور برای حذف یک یا چند رکورد از یک جدول مورد استفاده قرار گرفته و فرمت آن بشکل زیر است:

```
Delete [from] Table_name
       [From Table_join]
       [where Condition]
```

تمرین: برنامه ای بنویسید که تمام رکوردهای فروشهایی که نوع کتاب آنها تجاری بوده، حذف شود.

```
delete from sales where title_id IN
      (select title_id from titles where type='Business')
```

تمرین: برنامه ای بنویسید که تمام نویسندگانی را حذف کند که بیش از یک کتاب نوشته اند.

```
delete from authors where au_id in
  (select au_id from titleauthor
   group by au_id having count(title_id)>1)
```

نکته مهم: اگرچه برنامه فوق صحیح بنظر میرسد ولی بدلیل اینکه رکوردهایی از فایل نویسندگان (پدر) حذف میگردد که در فایل کتب نویسندگان (فرزند) دارای سابقه میباشد لذا سیستم اشکال منطقی گرفته و دستور اجرا نمیشود چون اول باید سابقه فرزندان حذف شده و سپس نویسندگان مذکور در فایل پدر حذف شوند. برای حل این مشکل باید در ابتدا کد نویسندگان مورد نظر در یک جدول موقت نگهداری شده و سپس رکوردهای با کد نویسنده موجود در جدول موقت از جدول فرزند حذف و پس از آن رکوردهای موردنظر از جدول پدر حذف گردد. در S.S دو نوع جدول موقت داریم:

- Local Temporary Table
  Create Table #Tablename

اعتبار این نوع جدول محدود بوده و یا به‌عبارتی Scope آن برای همان کاربر ایجاد کننده معتبر میباشد. ولذا سایر کاربران نیز میتوانند جدول موقت با همین نام ایجاد نمایند. این جدول با دستور Drop حذف شده و یا پس از خروج کاربر از سیستم (Disconnect) خودبه‌خود حذف میگردد. بهتر است بلافاصله پس از استفاده، این جداول با دستور Drop حذف گردد چون در زمان اجرای مجدد برنامه، اگر دستور Create اجرا شود، پیغام تکراری بودن داده خواهد شد.

- Global Temporary Table
  Create Table ##Tablename

اعتبار این نوع جدول برای تمام کاربران سیستم معتبر میباشد ولذا سایر کاربران نمیتوانند جدول موقت با همین نام ایجاد نمایند. این جدول نیز با دستور Drop حذف شده و یا پس از خروج تمامی کاربران از سیستم (Disconnect) خودبخود حذف میگردد.

بنابراین برای حل تمرین فوق، از برنامه زیر استفاده میشود:

```
Create Table #T (au_id varchar(11))
insert into #T
       select au_id from titleauthor group by au_id
                having count(title_id)>1
delete from titleauthor where au_id in (select au_id from #T)
delete from authors     where au_id in (select au_id from #T)
Drop Table #T
```

۸. توابع از پیش تعریف شده (Built in Functions):

۸.۱ تابع Ascii: این تابع کد اسکی یک کاراکتر را برمیگرداند. فرمت آن بشکل زیر است:

ASCII (character)

۸.۲ تابع Char: این تابع شکل حرفی یک کد اسکی را برمیگرداند. فرمت آن بشکل زیر است:

CHAR (value)

۸.۳ تابع Charindex: این تابع اولین شماره ستون عبارت اول (Pattern) در عبارت دوم (Expression) را برمیگرداند. فرمت آن بشکل زیر است:

CHARINDEX (pattern, expression)

۸.۴ تابع Left: این تابع به تعداد حروف عدد len از سمت چپ عبارت String را برمیگرداند. فرمت آن بشکل زیر است:

LEFT (string, len)

۸.۵ تابع Right: این تابع به تعداد حروف عدد len از سمت راست عبارت String را برمیگرداند. فرمت آن بشکل زیر است:

RIGHT (string, len)

۸.۶ تابع Len: این تابع تعداد حروف عبارت String را برمیگرداند. فرمت آن بشکل زیر است:

LEN (string)

۸.۷ تابع Datalength: این تابع طول اصلی فیلد Variable را برمیگرداند. فرمت آن بشکل زیر است:

DATALENGTH (variable)

۸.۸ تابع Lower: این تابع تمام حروف عبارت String را بصورت حروف کوچک برمیگرداند. فرمت آن بشکل زیر است:

LOWER (string)

۸.۹. تابع Upper: این تابع تمام حروف عبارت String را بصورت حروف بزرگ برمیگرداند. فرمت آن بشکل زیر است:

UPPER (string)

۸.۱۰. تابع Ltrim: این تابع تمامی حروف خالی (Space) سمت چپ عبارت String را حذف میکند. فرمت آن بشکل زیر است:

LTRIM (string)

۸.۱۱. تابع Rtrim: این تابع تمامی حروف خالی (Space) سمت راست عبارت String را حذف میکند. فرمت آن بشکل زیر است:

RTRIM (string)

۸.۱۲. تابع Space: این تابع به تعداد Value حروف خالی (Space) برمیگرداند. فرمت آن بشکل زیر است:

SPACE (value)

۸.۱۳. تابع Str: این تابع یک عبارت اعشاری را بصورت حرفی تبدیل میکند. Float عبارت عددی مورد نظر، Length طول عبارت جدید و Decimal تعداد اعشار است. بعنوان مثال نتیجه STR(1.368,4,2) برابر 1.37 میباشد. فرمت آن بشکل زیر است:

STR (float, length, decimal)

۸.۱۴. تابع Replace: این تابع تمام عبارات String2 را در عبارت String1 به عبارت String3 تبدیل میکند. فرمت آن بشکل زیر است:

REPLACE (string1, string2, string3)

۸.۱۵. تابع Replicate: این تابع به تعداد Value از حرف Char ایجاد مینماید. فرمت آن بشکل زیر است:

REPLICATE (char, value)

۸.۱۶. تابع Reverse: این تابع عبارت String را وارونه یا برعکس مینماید. فرمت آن بشکل زیر است:

REVERSE (string)

۸.۱۷. تابع Substring: این تابع عبارتی بطول Length و از شماره ستون Start از عبارت String ایجاد میکند. فرمت آن بشکل زیر است:

SUBSTRING (string, start, and length)

۸.۱۸. تابع Nchar: این تابع براساس کدینگ Unicode شکل حرفی کدی که برابر Value باشد را برمیگرداند. فرمت آن بشکل زیر است:

NCHAR (value)

۸.۱۹. تابع Patindex: این تابع نیز همانند تابع Charindex ، اولین شماره ستون عبارت اول (Pattern) در عبارت دوم (Expression) را برمیگرداند. فرمت آن بشکل زیر است:

PATINDEX (pattern, expression)

۸.۲۰. تابع Stuff: این تابع بخشی از عبارت String1 بطول Length و از ستون Start را با عبارت String2 تعویض مینماید. مثلاً نتیجه تابع Stuff(”Visual Basic Programming”,1,12, “VB.NET”) برابر “VB.NET Programming” میباشد. فرمت آن بشکل زیر است:

STUFF (string1, start, length, string2)

۸.۲۱. تابع Quotename: این تابع یک عبارت String را در داخل یک زوج علامت Quote_Char قرار میدهد. مثلاً نتیجه تابع Quotename(‘Test’,”{“) برابر {Test} میشود. فرمت آن بشکل زیر است:

QUOTENAME (string, quote_char)

۸.۲۲. تابع Cast: این تابع نوع هر متغیری را به نوع جدید تبدیل می کند. فرمت آن بشکل زیر است:

CAST (Variable As Datatype)

تمرین: برنامه ای بنویسید که نویسندگان را به ترتیب طول نام خانوادگی مرتب و لیست کند.

```
Select * from Authors order by Len(Au_lname)
```

تمرین: برنامه ای بنویسید که ۱۵ حرف از نام خانوادگی را به نام نویسندگان چسبانده و لیست نماید.

```
Select left(au_lname+space(15),15)+au_fname as fullname from Authors
```

۹. متغیرها و دستورات شرطی:

۹.۱. متغیر (Valriable): تمامی متغیرها با علامت @ شروع شده و فرمت تعریف و مقدار دهی آنها به شکل زیر است:

Declare Var_name1 Datatype1 [, Var_namen Datatypen]

Set Var_name = Expression

Set Var_name = (Select.....)

مثال: در مثالهای زیر چند نمونه متغیر تعریف شده است:

```
Declare @x int, @y varchar(50)
Declare @t money, @z decimal(5,2)
```

```
Set @x=123

Set @y='hossein'

Set @t=123.45

Set @z=999.99

Set @x=2*10-5

Set @x=(Select count(*) from titles)
```

۹.۲. دستور شرطی IF: این دستور برای اجرای شرطی برخی از دستورات مورد استفاده قرار میگیرد و فرمت آن به شکل زیر است:

```
IF    condition
      Begin
          ...
      End
[else
      Begin
          ...
      End]
```

۹.۳. دستور شرطی While: این دستور یک سری از دستورات را، مادامی که شرط موردنظر برقرار باشد، اجرا میکند. در حلقه While دستور Break کنترل را به خارج از حلقه و دستور Continue کنترل را به ابتدای حلقه برمیگرداند و فرمت آن به شکل زیر است:

```
While    condition
      Begin
          ...
          IF      condition2
                  Break
          End
          ...
          IF      condition3
                  Continue
          End
          ...
      End
```

تمرین: برنامه ای بنویسید که اعداد را از ۱ تا ۲۰ چاپ کند:

```
Declare @i int
Set @i=1
While @i<=20
     Begin
     Print @i
     Set @i=@i+1

End
```

۱۰. توابع (Functions):

تعریف: در SQL دو نوع تابع به شرح زیر داریم:

- T_SQL Functions
- Clr.Net Functions

توابع نوع دوم در SQL 2005 اضافه شده و با زبانهای #C و VB.NET قابل نوشتن میباشد. هر کدام از انواع توابع فوق به سه گروه زیر تقسیم میگردند:

1. Scaler Valued user Define functions
2. Table Valued user Define functions Inline
3. Table Valued user Define functions Multi statement

۱۰.۱. توابع Scaler Valued user Define functions: در ابتدا توابع گروه اول مورد بحث قرار میگیرند.

فرمت کلی این توابع به شکل زیر است:

```
Create Fumnction F_name (parameters)
    Returns Datatype as
        Begin
        ...
        ...
        Return V_Name
        End
```

مثال: تابع زیر حاصلضرب دو عدد را بر میگرداند:

```
Create Function mymul(@a int, @b int)
   returns int as
      Begin
      Declare @c int
      Set @c=@a*@b
      return @c
      end
```

با اجرای برنامه فوق، تابع mymul بعنوان بخشی از دیتابیس فعال تعریف و ذخیره شده و برای رؤیت آن میتوان از منوی سمت چپ صفحه، بر روی نام دیتابیس کلیک کرده و مسیر زیر انتخاب شود. پس از آن فایل dbo.mymul مشاهده خواهد شد.

Programmability>Functions>Scalar-Valued Functions

روشهای مختلفی برای استفاده از توابع وجود دارد که به اختصار سه روش آن با مثال نشان داده میشود.

مثال ۱: در مثال زیر حاصلضرب برابر ۲۵۰ نشان داده میشود.

```
Select dbo.mymul(25,10)
```

مثال ۲: در مثال زیر نیز حاصلضرب برابر ۲۵۰ نشان داده میشود.

```
Declare @x int
Set @x=dbo.mymul(25,10)
Print @x
```

مثال ۳: در مثال زیر در ابتدا یک جدول شامل دو فیلد عددی ساخته شده و پس از ایجاد تعدادی رکورد، حاصلضرب آنها را چاپ می کند:

```
Create Table T(x int, y int)
insert into T Values(10,12)
insert into T Values(15,14)
insert into T Values(18,20)
Select x,y,dbo.mymul(x,y) as M from T
```

تمرین: تابعی بنام Alltrim بنویسید که یک پارامتر از نوع Varchar(max) را دریافت کرده و همه Space های موجود در آن را حذف کرده و نتیجه را برگرداند. (بدون استفاده از دستور Replace)

```
CREATE Function AllTrim (@Str Varchar(max) )
Returns Varchar(max) as
Begin
     Declare @I int,@L int,@Res Varchar(max), @ch Char(1)
     Set @I = 1
     Set @Res=''
     Set @L  = Len(@Str)
     While @I<= @L
          Begin
               Set @Ch = Substring(@Str,@i,1)
               If @ch <> ' '
                    Set @Res = @Res + @ch
               Set @I = @I + 1
          End
     Return @Res
End
SELECT dbo.AllTrim(' h  o s s e i n   f a r a h an  i ')
SELECT dbo.AllTrim('  h o s s  ein farahani')
SELECT dbo.AllTrim('hossein fara  han   i)
SELECT dbo.AllTrim('hossein farahani)
```

نکته: بدیهی است میتوان همانند مثال زیر از دستور Replace بجای تابع فوق استفاده کرد:

```
CREATE Table TrimTest(Code int, Name Varchar(50))
Insert into TrimTest Values (1,'tehran')
Insert into TrimTest Values (2,'  t e  h r an')
Insert into TrimTest Values (3,'tehra      n')
Insert into TrimTest Values (4,' t e h r a  n')
SELECT * FROM TrimTest Where Name = 'tehran'
SELECT * FROM TrimTest Where dbo.Alltrim( Name) = 'tehran'
SELECT * FROM TrimTest Where replace(Name,' ','') = 'tehran'
Update TrimTest Set Name = dbo.Alltrim( Name)
```

تمرین: تابعی بنویسید که دو تاریخ شمسی را بصورت Char(80) (بشکل yyyymmdd) بعنوان ورودی دریافت کرده و اختلاف روز بین این تاریخها را محاسبه کرده برگرداند:

```
CREATE Function DateToDay(@Date char(8))
Returns int as
Begin
     Declare @y int,@m int,@d int
     Set @y = Cast( Substring(@Date,1,4) as int)
     Set @m = Cast( Substring(@Date,5,2) as int)
     Set @d = Cast( Substring(@Date,7,2) as int)

     Set @y = (@y-1) * 365
     Set @m = Case When @m <= 6 Then (@m-1) * 31
                                Else (@m -7) * 30 + 186
               End
     Set @d = @y + @m + @d
     Return @d
End

CREATE Function DiffDate(@SDate char(8),@EDate char(8))
returns  int as
Begin
     DECLARE @d1 int,@d2 int
     SET @d1 = dbo.DateToDay(@Sdate)
     SET @d2 = dbo.DateToDay(@Edate)
     Return @d2-@d1
End

SELECT dbo.DiffDate('13870101','13870101')
SELECT dbo.DiffDate('13870101','13870102')
SELECT dbo.DiffDate('13870101','13870131')
SELECT dbo.DiffDate('13870101','13870201')
SELECT dbo.DiffDate('13870101','13870631')
SELECT dbo.DiffDate('13870101','13870701')
SELECT dbo.DiffDate('13870101','13871229')
SELECT dbo.DiffDate('13870101','13880101')
SELECT dbo.DiffDate('00010101','13870319')
```

تمرین: با نوشتن دو تابع محاسبه مبلغ فروش هر کتاب و تعداد نویسندگان هر کتاب، برنامه بنویسید که سهم هر نویسنده از فروش کتاب را محاسبه نماید:

```
CREATE Function CountofAuthor(@Title_id nvarchar(6))
Returns int as
Begin
     Declare @c int
     Set @c = (Select Count(au_id) FROM TitleAuthor
          Where Title_id=@Title_id)
     Return @c
end
```

```sql
CREATE Function AmountOfSale(@Title_id Nvarchar(6))
returns money as
Begin
     Declare @c money
     Set @C = (SELECT Sum(qty*Price)
                 FROM Titles Inner join Sales
                         On Titles.Title_id= Sales.Title_id
                 Where Titles.Title_id=@Title_id)
     Set @c = isnull(@c,0)
     Return @C
End

SELECT au_id,Sum(dbo.AmountOfSale(Title_id)/
                 dbo.CountOfAuthor(Title_id)) as Tp
     FROM titleAuthor
```

۱۰.۲. جداول مجازی View: دستور View یک Virtual Table یا Stored Query است که بوسیله آن میتوان یک دستور Select را بشکل Object در SQL ذخیره کرد و فرمت آن بشکل زیر است:

```
Create View vw_name
        As
                Select statement
        Go
```

مثال: دستور زیر پس از اجرا ، یک فایل بنام dbo.AuthorofCA را در قسمت دیتابیس فعال (اگر در زمان اجرای دستور pubs فعال باشد در این دیتابیس) ایجاد مینماید.

```
Create View AuthorofCA
     As
          Select * from Authors where State='CA'
     Go
```

پس از آن می‌توان برای دسترسی به نویسندگان کالیفرنیایی، از دستور زیر استفاده کرد:

```
Select * from AuthorofCA
```

یا می‌توان با دستور زیر کد کتابهای نویسندگان کالیفرنیایی را بدست آورد:

```
Select * from AuthorofCA inner join Titleauthor
          on AuthorofCA.au_id=Titleauthor.au_id
```

نکته ۱: چنانچه یک دستور Select در تعداد زیادی از دستورات مورد استفاده قرار می‌گیرد بهتر است به‌منظور Structured Programing بصورت View تعریف شود.

نکته ۲: میتوان دسترسی برخی از کاربران را برای یک View خاص تعریف و مشخص کرد لذا دستور View میتواند به ایمن سازی سیستم کمک کند.

نکته ۳: پشت سر View هیچ رکوردی ذخیره نمی شود بلکه با اجرای دستور View، دستور Select داخل View اجرا شده و نتیجه آن بازگشت داده میشود. بنابراین فقط به اندازه دستور Select و Execution Plan مربوط به آن، فضای حافظه اشغال میگردد.

نکته ۴: اگرچه برای تغییر یک دستور View میتوان ابتدا آنرا با دستور Drop حذف کرده و سپس به‌وسیله دستور Create ایجاد نمود ولیکن این روش مناسبی نیست چون با حذف آن، تمامی دسترسی های کاربران نیز حذف میگردد لذا توصیه میگردد اگر قبلاً View ایجاد شده باشد برای تغییر آن به‌جای Create از دستور Alter استفاده شود.

نکته ۵: بمنظور افزایش ایمنی سیستم و مخفی کردن محتوای Select داخل View میتوان قبل از کلمه As عبارت With incryption را اضافه نمود.

نکته ۶: با توجه به احتمال تداخل و پیچیده شدن عملیات دستورات مختلف، استفاده از دستورات Insert، Delete و Update در داخل View به‌هیچ عنوان توصیه نمی‌گردد.

۱۰.۳. توابع Table Valued user Define functions Inlin: این توابع نیز شبیه View یک دستور Select را بصورت مجازی تعریف می‌نماید ولیکن چون پارامتر دریافت می‌کند جایگزین بسیار خوبی برای View می‌باشد. فرمت کلی ساخت این توابع به‌شکل زیر است:

```
Create Fumnction F_name (parameters)
        Returns Table as
        Return (Select Statement)
```

مثال: تابع زیر مشابه مثال View فوق میباشد با این تفاوت که میتوان کد ایالت را بصورت پارامتر مشخص نمود:

```
Create Function Authorof(@state char(2))
      Returns Table As
      Return(Select * from Authors where State=@state)
```

روش اجرای تابع نیز به‌شکل زیر است:

```
Select * from dbo.Authorof('CA')
```

برای نمایش کد کتابهای نویسندگان یک ایالت خاص از دستور زیر استفاده میشود:

```
Select * from dbo.Authorof('CA') as K
      inner join Titleauthor on K.au_id=Titleauthor.au_id
```

تمرین: تابعی از نوع Inline بنویسید که یک پارامتر از نوع money بگیرد و نام کتاب و مبلغ فروش کتابهایی را برگردند که فروششان از مقدار پارامتر بیشتر باشد.

```
Create Function getdata(@sal money)
      Returns Table As
      Return(   Select title,sum(qty*price) as tc
                from titles inner join sales
                on titles.title_id=sales.title_id
                group by title
                having sum(qty*price)>@sal)
```

۱۰.۴. توابع Table Valued user Define functions Multi statement: فرمت ساخت آن عبارتنداز:

```
Create Fumnction F_name (parameters)
        Returns @t_name Table (field1 datatype, [, fieldn datatype])
        As
        Begin
                ...
        Return
End
```

تمرین: تابعی از نوع multi statement بنویسید که نام همه نویسندگان، ناشرها و فروشگاههایی که در یک ایالت خاص که از طریق پارامتر دریافت میگردد را نمایش دهد.

```
Create Function x910(@State char(2))
      returns @H table (Name varchar(200),Type char(1))
      as
      Begin
      Insert into @H
           SELECT au_lname + ' ' + au_fname as Name,'A' as type
                FROM Authors
                Where State = @State
      Insert into @H
           SELECT Pub_name,'P' as type
                FROM Publishers
                Where State=@State
      Insert into @H
           SELECT Stor_name,'S' as Type
                FROM Stores
                Where State = @State
      return
      end

SELECT * FROM dbo.x910('ca')
```

تمرین: تابعی از نوع multi statement بنویسید که نام کتاب، کدکتاب و مبلغ جدید کتاب را که از رابطه زیر محاسبه می کند برگرداند (درصورتی که قیمت کتاب بیش از ۳۰ دلار شد به قیمت ۳۰ دلار تبدیل کند):

- اگر ناشر کتاب اهل کالیفرنیا بود قیمت کتاب را ۱۰٪ افزایش دهد.
- در غیر اینصورت اگر کتاب بیش از یک نویسنده داشت قیمت کتاب را ۵٪ افزایش دهد.
- در غیر اینصورت اگر کتاب بیش از ۲۰۰ دلار فروش داشت قیمت کتاب را ۲٪ افزایش دهد.
- در غیر اینصورت قیمت کتاب را ۱٪ افزایش دهد.

```
CREATE Function x911()
      Returns @H table(title_id nvarchar(6),
                       Title   varchar(100),
                       NewPrice money)
      As
      Begin
      Insert into @H
           SELECT Title_id,title,Price *1.1 as NewPrice
                FROM Titles
                Where Pub_id IN (SELECT Pub_id FROM Publishers
                                  Where State='ca')
      Insert into @H
           SELECT Title_id,title,Price *1.05 as NewPrice
                FROM Titles
                Where Title_id IN (SELECT Title_id
                                   FROM titleAuthor
                                   Group by Title_id
                                   having count(au_id) > 1)
                AND title_id  Not IN (SELECT Title_id FROM @H)
      Insert into @H
           SELECT Title_id,title,Price *1.02 as NewPrice
                FROM Titles
                Where Title_id IN (SELECT Titles.Title_id
                              FROM Titles Inner join Sales
                              On Titles.Title_id=SAles.Title_id
                              Group by Titles.Title_id
                              Having Sum(qty*Price) > 500)
                AND title_id  Not IN (SELECT Title_id FROM @H)

      Insert into @H
           SELECT Title_id,title,Price *1.01 as NewPrice
                FROM Titles
                Where title_id  Not IN (SELECT Title_id FROM @H)
  Update @h Set NewPrice = 30 Where NewPrice > 30
  Return
End

SELECT * FROM dbo.x911()
```

۱۱. روال‌ها (Procedure):

تعریف: در SQL 2005 چهار نوع Procedure به شرح زیر وجود دارد:

1. User Defined Stored Procedure
2. System Stored Procedure
3. Extent Stored Procedure
4. Clr.net Stored Procedure

۱۱.۱. روالهای تعریف شده توسط کاربران User Defined Stored Procedure: روالها نیز همانند توابع برای تعریف یک سری دستورالعمل می‌باشند که در بخشهای مختلف سیستم به دفعات استفاده می‌گردند ولیکن دو تفاوت عمده بین روالها و توابع وجود دارد:

- با اجرای توابع یک مقدار مشخص برگردانده میشود ولی لزوماً Procedure مقداری را برنمی گرداند.
- در توابع مجاز به استفاده از Insert، Delete و Update نبوده و همچنین عملیات ایجاد جداول و ایندکسها نیز در داخل توابع امکان پذیر نیست ولیکن در داخل Procedure ها تمامی این عملیات مجاز و ممکن است.

فرمت ساخت Procedure به شکل زیر است:

```
Create Procedure Proc_name(Parameters)
    As
      Begin
     .....
      End
    Go
```

مثال ۱: فرض کنید جدولی بنام Test شامل دو فیلد Code (بصورت Primary Key) و Name داشته باشیم. یک Procedure جهت اضافه نمودن رکورد به این جدول بنویسید.

```
Create Proc Testins1(@Code int,@name varchar(50))
      As
      Begin
            Insert into Test values(@Code,@Name)
      End
      Go
```

و دستورات زیر روش استفاده از این روال را نشان میدهد:

```
Exec Testins1 1,'Reza'
Exec Testins1 2,'Hamid'
```

مثال ۲: چنانچه بخواهیم در مثال قبل Code توسط Procedure تعیین شود برنامه به شکل زیر خواهد بود:

```
Create Proc Testins2(@name varchar(50))
      As
      Begin
            Declare @ncode int
            Set @ncode=(Select Max(Code) from Test)
            Set @ncode=isnull(@ncode,0)+1
            Insert into Test values(@ncode,@Name)
      End
      Go
```

و دستورات زیر روش استفاده از این روال را نشان میدهد:

```
Exec Testins2 'Akbar'
Exec Testins2 'Hoda'
```

مثال ۳: و اگر بخواهیم در مثال قبل، مقدار Code تولید شده نشان داده شود به شرح زیر خواهد بود:

```
Create Proc Testins3(@name varchar(50),@code int output)
      As
      Begin
            Declare @ncode int
            Set @ncode=(Select Max(Code) from Test)
            Set @ncode=isnull(@ncode,0)+1
            Insert into Test values(@ncode,@Name)
            Set @code=@ncode
      End
      Go
```

و دستورات زیر روش استفاده از این روال را نشان میدهد:

```
Declare @t int
Exec Testins3 'Akbar',@t output
Print @t
```

مثال۴: در این بخش یک مثال کامل از طراحی یک سیستم و ارتباط بین VB.Net و SQL Server 2005 نشان داده شده است. اجزاء برنامه به شرح زیر است:

- در ابتدا وارد SQL Server 2005 شده و یک دیتابیس بنام ClassTest می‌سازیم.
- سپس یک جدول بنام Test با دو فیلد Code و Name ایجاد نموده چند رکورد بعنوان نمونه به آن اضافه مینماییم.
- بعد از آن یک جدول بنام TestLog برای ذخیره کردن رکوردهای حذف شده و تاریخ حذف آنها ایجاد می‌گردد.
- پس از آن سه Procedure بنامهای TestIns، TestUpd و TestDel برای اضافه کردن، به‌روزرسانی و حذف رکوردهای جدول Test ایجاد مینماییم.

کد برنامه های فوق تا این مرحله که در S.S 2005 نوشته شده است به شکل زیر میباشد:

```
CREATE Database ClassTest
Go
Use ClassTest
Go
CREATE Table Test(Code int Primary key,Name Varchar(50))
Go
Insert into Test Values (1,'Reza')
Insert into Test Values (2,'Hamid')
Go

Create Proc TestIns(@name varchar(50),@Code int output)
as
Begin
  Declare @NCode int
  Set @NCode  = (SELECT Max(Code) FROM Test)
  Set @NCode = Isnull(@NCode,0) + 1
  Insert into Test Values (@NCode,@Name)
  Set @Code = @NCode
End
go

CREATE Proc TestUpd(@Code int,@Name varchar(50))
as
Begin
  Update Test Set [Name] = @Name Where Code = @Code
End
Go
CREATE Proc TestDel(@Code int)
as
Begin
  Delete FROM Test Where Code = @Code
End
Go
```

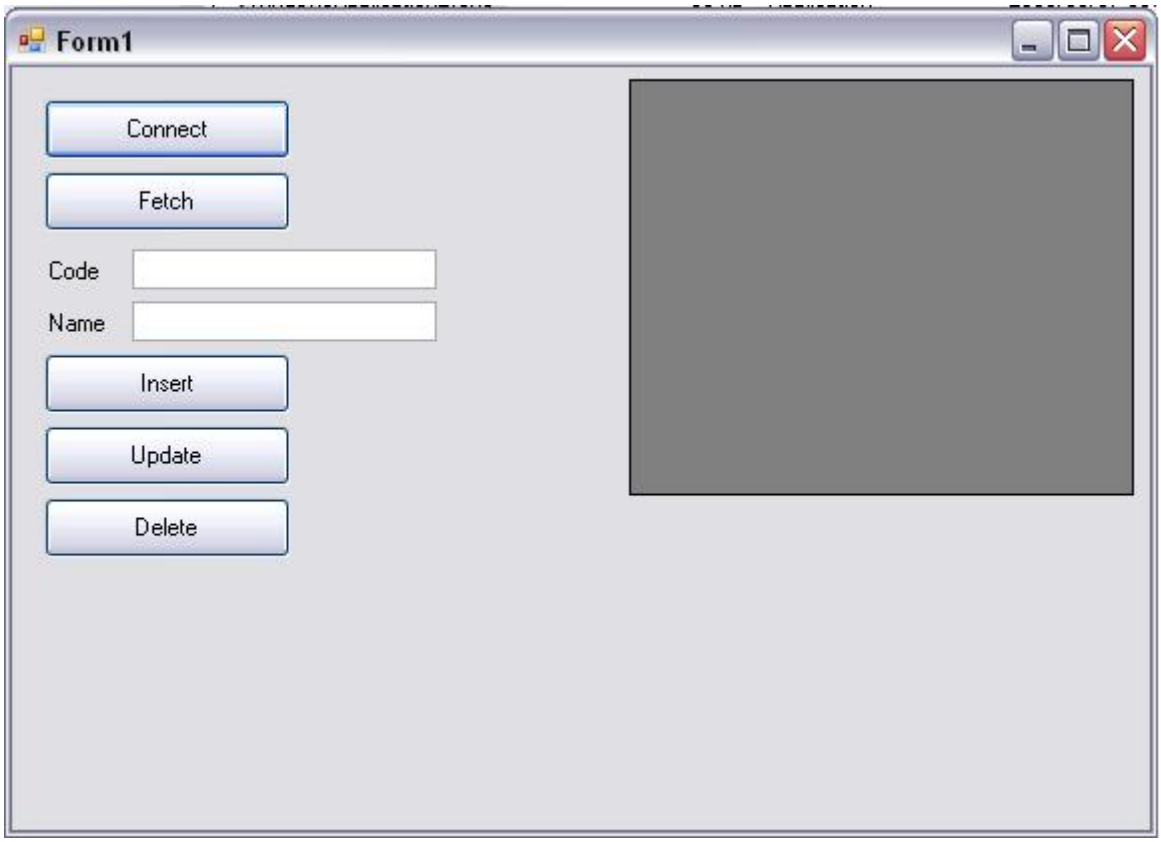
- پس از آن از طریق Visual Studio و Windows Application وارد محیط برنامه نویسی VB.Net شده و سیستم ساده‌ای شامل اجزاء زیر مینویسم. (بدیهی است دستورات و روش برنامه نویسی به زبان VB.Net در کلاس دیگری تدریس شده و در این کلاس مورد بحث قرار نخواهد گرفت)
- اولین بخش سیستم، ایجاد یک قسمت بنام Connect برای اتصال به دیتابیس میباشد.
- پس از آن دو پنجره ورودی برای تایپ و دریافت Code و Name ایجاد می‌گردد.
- سپس سه قسمت Insert، Update و Delete برای افزودن، بروزرسانی و حذف رکوردهای جدول Test درنظر گرفته میشود.
- در انتها بخشی از صفحه برای نمایش رکوردهای جدول Test پیش بینی میگردد که بوسیله دکمه Fetch اطلاعات جدید نشان داده میشود.

کد برنامه مذکور به شکل زیر خواهد بود:

```
Public Class Form1
    Dim cnt As New SqlClient.SqlConnection
    Dim cmd As New SqlClient.SqlCommand
    Dim dt As New DataTable
    Dim da As New SqlClient.SqlDataAdapter

    Private Sub btnConnect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnConnect.Click
        With cnt
            .ConnectionString = "server=server33;database=Classtest;uid=sa;pwd=123;"
            .Open()
        End With
    End Sub

    Private Sub btnFetch_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFetch.Click
        With cmd
            .Connection = cnt
            .CommandType = CommandType.Text
            .CommandText = "SELECT * FROM test"
        End With
        dt.Clear()
        With da
            .SelectCommand = cmd
            .Fill(dt)
        End With
        DataGridView1.DataSource = dt
    End Sub

    Private Sub btnInsert_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInsert.Click

        With cmd
            .Connection = cnt
            .CommandType = CommandType.StoredProcedure
            .CommandText = "TestIns"
             With.Parameters
                .Clear()
                .Add("@Name", SqlDbType.VarChar, 50)
                .Add("@Code", SqlDbType.Int)
                .Item("@Code").Direction = ParameterDirection.Output
                .Item("@Name").Value = txtName.Text
             End With
            .ExecuteNonQuery()
             txtCode.Text =.Parameters("@Code").Value
        End With

    End Sub

    Private Sub btnUpdate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnUpdate.Click
        With cmd
            .Connection = cnt
            .CommandType = CommandType.StoredProcedure
            .CommandText = "TestUpd"
             With.Parameters
                .Clear()
                .Add("@Code", SqlDbType.Int)
                .Add("@Name", SqlDbType.VarChar, 50)
```

```
                .Item("@Code").Value = txtCode.Text
                .Item("@Name").Value = txtName.Text
            End With
           .ExecuteNonQuery()
        End With
    End Sub

    Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDelete.Click
        With cmd
           .Connection = cnt
           .CommandType = CommandType.StoredProcedure
           .CommandText = "TestDel"
            With.Parameters
                .Clear()
                .Add("@Code", SqlDbType.Int)
                .Item("@Code").Value = txtCode.Text
            End With
           .ExecuteNonQuery()
        End With

    End Sub
           End Class
```

و نمونه شکل ورودی این سیستم به شکل زیر میباشد:



۱۱.۲. روالهای سیستمی (System Stored Procedure): شرکت عرضه کننده S.S یک‌سری روال سودمند را برای استفاده کاربران از پیش تهیه نموده و تحت دیتابیس Master قرارداده است. این روالها را System S.P نامند و از طریق آدرس زیر میتوان با نام آنها آشنا شد. چنانچه مشاهده خواهد شد نام این روالها با sys. sp_ شروع میشود:

Database > master > Programmability > Stored Procedure > System S.P

۱۱.۳. روالهای گسترده (Extent Stored Procedure): برخی از مواقع نیاز به انجام کارهایی داریم که S.S توان انجام آن‌را ندارد. مثلاً اگر بخواهیم از داخل S.S یک Email را دریافت یا ارسال کنیم که دستوری توسط S.S برای این‌کار پیش بینی نشده است، S.S برای حل این مشکل امکان صدا زدن و اجرای Procedure های نوشته شده بزبان C++ را پیش بینی نموده است. نام اینگونه روالها با xp_ شروع شده و تعدادی از این روالها که توسط Microsoft تهیه شده است در همان بخش روالهای سیستمی وفوق الذکر قابل مشاهده میباشد.

۱۱.۴. روالهای زبان C# و VB.Net (Clr.Net Stored Procedure): با توجه به کابرد مفید روالهای گسترده فوق الذکر، امکان تهیه روال بزبان C# و VB.Net نیز در S.S 2005 اضافه گردید. اینگونه روالها راClr Net S.P. نامند. با توجه کاربرد زیاد این روالها به چند نمونه از آنها اشاره میگردد:

نکته مهم: بوسیله برخی از دستورات SQL میتوان حداقل به ۲۱ روش مختلف وارد سیستم و دیتابیس دیگران شده و عملیات خرابکارانه انجام داد ولذا از این نظر جای نگرانی جدی وجود داشته و باید در زمان برنامه نویسی بزبان SQL کنترلها و مراقبتهای لازم انجام گردد. در ادامه به چندین روش اشاره میگردد:

A. Xp_cmdshell: دیتابیس صفحه دستورات Dos تحت Windows را shell مینامند و میتوان بوسیله برخی از دستورات S.S همانند دستورات Dos عمل کرد. بعنوان مثال دستور زیر لیست فایلهای سیستمی که فعال میباشد را نمایش میدهد:

```
Exec xp_cmdshell 'Dir C:\'
```

B. یکی دیگر از روشهای ورود به سیستم و دیتابیس دیگران استفاده از دستور Xp_cmdshell بصورت زیر است. با دستور اول یک User با نام کاربری vahid و کلمه رمز 123 بر روی دستگاه کاربری خاص ایجاد میگردد و بوسیله دستور بعدی همین کاربر بصورت Administrator تعریف شده است.

```
Exec xp_cmdshell 'Net user Vahid 123/add'

Exec xp_cmdshell 'Net localgroup administrator Vahid 123/add'
```

C. فرض کنید جدول Users را با مشخصات زیر داشته باشیم. با دستور Select اول میتوان بروش عادی و با کلمه رمز وارد سیستم شد ولی با دستور Select های بعدی و با اجرای دستور Exec در ادامه دستور اول، میتوان بدون اطلاع کاربر اصلی نسبت به ایجاد یک کاربر جدید حتی بصورت Admin اقدام نمود:

```
Create Table Users(Code     int,
                   Name     varchar(50),
                   Pwd      varchar(50))

Insert into Users Values(1,'reza','123')

Select Count(*) from users where Name='reza' and Pwd='123'

Select Count(*) from users where Name='reza' and Pwd='123';
      exec sp_addlogin 'mm','111'

Select Count(*) from users where Name='reza' and Pwd='123';
      exec sp_addsrvrolemember 'mm','sysadmin'
```

D. فرض کنید هیچگونه اطلاعی از نام فیلدهای یک جدول نداریم و بخواهیم نام فیلدها را بدست آوریم با دستور اول زیر، سیستم پیغام خطا میدهد که باید فیلد Code در قسمت Group by قرار گیرد. دستورات بعدی نیز پیغام مشابه برای قراردادن فیلدهای Name و Pwd در Group by را میدهد. بنابراین به همین سادگی میتوان از طریق پیغامهای خطا نام فیلدهای یک جدول را بدست آورده و عملیات خرابکارانه خود را به مرحله اجرا گذاشت!

```
select * from users having count(*)>1

select * from users Group by Code having count(*)>1

select * from users Group by Code,Name having count(*)>1

select * from users Group by Code,Name,Pwd having count(*)>1
```

۱۲. دستور Trigger:

۱۲.۱. دستور Trigger معمولی : کاربرد و خصوصیات Trigger ها دقیقاً مشابه Procedure هاست و تنها در دو مورد زیر با هم متفاوت میباشند:

- Trigger ها برای جدول ها نوشته می شوند و در اثر اجرای سه دستور Insert; Delete و Update که Trigger به این رویدادها متصل شده است، اجرا میشوند. بنابراین Procedure ها در مکان و زمان مشخص صدا زده و اجرا میشوند ولی Trigger ها بطور اتوماتیک اجرا و فراخوانی میگردند.
- Procedure ها پارامتر پذیرند ولی Trigger ها پارامتر نمی پذیرند.

فرمت کلی دستور Trigger به شکل زیر است:

```
Create Trigger Tr_name on Table_name
    For / After [Insert] , [Delete] , [Update]
  As
     Begin
    .....
     End
  Go
```

نکته ۱: از لحظه اجرای یک Trigger تا پایان اجرای آن، دو جدول مجازی بنامهای Inserted و Deleted ساخته شده و دراختیار برنامه نویس قرار می گیرد که این دو جدول حاوی اطلاعات زیر است:

- اگر Trigger در اثر اجرای دستور Insert فراخوانی شده باشد یک کپی از همه رکوردهای اضافه شده در جدول Inserted قرار می گیرد.
- اگر Trigger در اثر اجرای دستور Delete فراخوانی شده باشد یک کپی از همه رکوردهای حذف شده در جدولِ Deleted قرار می گیرد.
- اگر Trigger در اثر اجرای دستور Update فراخوانی شده باشد یک کپی از همه مقادیر قبلی رکوردهای بروزرسانی شده در جدول Inserted و یک کپی از مقادیر جدید رکوردهای بروزرسانی شده در جدول Deleted قرار می گیرد.

نکته ۲: برای یک جدول میتوان بیش از یک For Trigger نوشت.

نکته ۳: زمانی عملیات Ins/Del/Upd روی یک جدول، موفق انجام میشوند که هم خود دستورات درست اجرا شوند و هم تمام Trigger های مرتبط با دستور روی جدول صحیح اجرا شوند، درغیر اینصورت تمام عملیات بازگشت (RollBack) داده میشوند.

نکته ۴: Trigger ها به ازای دستورات Ins/Del/Upd اجرا میشوند یعنی به ازای اجرای هر دستور Ins/Del/Upd، دستور Trigger فقط یکبار اجرا میشود حتی اگر دستور Ins/Del/Upd بیش از یک رکورد را تحت تاثیر قرار دهد. به‌عبارتی دیگر دستوراتی که بیش از یک رکورد را تحت تاثیر قرار میدهد به ازای عمل روی هر رکورد، Trigger فراخوانی نخواهد شد بلکه بعد از تحت تاثیر قرار گرفتن کل رکوردها، یک بار Trigger فراخوانی میشود. بدیهی است در هنگام فراخوانی Trigger همه رکوردهای تحت تاثیر دستور Ins/Del/Upd در جداول Inserted یا Deleted (با توجه به نوع دستور) قرار می‌گیرد.

مثال ۱: فرض کنید جدول Test با دو فیلد Code int و Name varchar(50) و جدول TestLog با سه فیلد Code int و Name varchar(50) و DelDate datetime را داشته باشیم. می‌خواهیم به‌کمک Trigger در زمان حذف رکورد از Test، یک کپی از رکوردهای حذف شده در جدول TestLog کپی شود.

```
Create Trigger Testlogtrg on Test
    for Delete
    As
    Begin
    Insert into TestLog Select Code,Name,Getdate()from Deleted
    End
```

نکته ۱: هر اتصال یا Connection کاربر به SQL دارای یک شماره Spid خاص می‌باشد که با دستور @@spid قابل دریافت می‌باشد. مثلاً دستور زیر شماره Spid را نشان میدهد:

```
Select @@Spid
```

نکته ۲: ضمناً SQL مشخصات کاربران متصل به SQL را در جدول sys.sysprocesses قرار میدهد که میتوان اطلاعات بیشتری را از روی این جدول استخراج و در Trigger از آن استفاده کرد. دستور زیر نیز نام فیلدها و مشخصات کامل را نشان می‌دهد:

```
Select * from sys.sysprocesses
```

مثال ۲: فیلدهای Info1 الی Info5 از نوع Varchar(500) به فیلدهای جدول Test مثال قبل اضافه شده و سپس برخی از اطلاعات مهم کاربری، که رکورد را حذف نموده است، را در جدول TestLog قرار دهید.

```
Create Trigger Testlogtrg2 on Test
  for Delete
  As
  Begin
  Declare @info1    varchar(500),
          @info2    varchar(500),
          @info3    varchar(500),
          @info4    varchar(500),
          @info5    varchar(500)
  Select @info1=hostname,
         @info2=net_address,
         @info3=loginame,
         @info4=net_library,
         @info5=login_time
         from sys.sysprocesses where spid=@@spid
  Insert into TestLog
     Select Code,Name,Getdate(),@info1,@info2,@info3,@info4,@info5
     from Deleted
   End
```

مثال ۳: فرض کنید دو جدول Factor و FatcorItem به شرح زیر برای ثبت فاکتورهای فروش درنظر گرفته شده است. جدول Factor برای ثبت اطلاعات کلی فاکتور از قبیل شماره، تاریخ و جمع مبلغ فاکتور و جدول FactorItem برای ثبت اقلام ریز فاکتور پیش بینی شده است. برنامه Trigger زیر برای بروزرسانی مبلغ کل فاکتور نوشته شده‌است به‌گونه‌ای که اگر رکوردی از اقلام فاکتور در جدول FactorItem اصلاح، حذف یا اضافه شود مبلغ کل فاکتور در جدول Factor اصلاح و بروزرسانی خواهد شد. (بدیهی است در زمان اضافه کردن رکورد به جدول FactorItem باید قبلاً مشخصات این فاکتور در جدول Factor ثبت شده باشد یا به‌عبارتی دیگر با این برنامه تنها میتوان به اقلام یک فاکتور ثبت شده قبلی، یک قلم جدید اضافه کرد)

```
CREATE TABLE dbo.Factor(FNo int NOT NULL,
                        FDate char (8) NULL,
                        FCust varchar (50) NULL,
                        TotalPrice money NULL)
CREATE TABLE dbo.FactorItem(FNo int NOT NULL,
                            Row int NOT NULL,
                            Kala varchar (50) NULL,
                            Qty money NULL,
                            Price money NULL)
ALTER TABLE dbo.Factor WITH NOCHECK ADD
  CONSTRAINT PK_Factor PRIMARY KEY CLUSTERED(FNo)
GO
ALTER TABLE dbo.FactorItem WITH NOCHECK ADD
  CONSTRAINT PK_FactorItem PRIMARY KEY CLUSTERED(FNo,Row)
GO
ALTER TABLE dbo.FactorItem ADD
  CONSTRAINT FK_FactorItem_Factor
  FOREIGN KEY (FNo)REFERENCES dbo.Factor(FNo)
GO
Create TRIGGER ComputeTotalPrice ON dbo.FactorItem
  FOR INSERT, UPDATE, DELETE
  AS
  Begin
    update factor set totalprice=isnull(Tp,0) from factor inner join
        (select factor.FNo,sum(qty*price) as Tp
              from factor left join factoritem
              on factor.FNo=factoritem.FNo
              where factor.FNo IN(select fno from inserted)
                 or factor.FNo IN(select fno from deleted)
              group by factor.FNo) k
              on factor.FNo=k.FNo
  End
```

راه حل دوم:

```
Create TRIGGER ComputeTotalPrice ON dbo.FactorItem
  FOR INSERT, UPDATE, DELETE
  AS
  Begin
    update factor set totalprice=totalprice+ts from factor inner join
        (select inserted.FNo,sum(qty*price) as ts
              from inserted group by inserted.FNo) k
              on factor.FNo=k.FNo
    update factor set totalprice=totalprice-ts from factor inner join
        (select deleted.FNo,sum(qty*price) as ts
              from deleted group by deleted.FNo) k
              on factor.FNo=k.FNo
  End
```

۱۲.۲. Instead of Trigger:

تعریف: دستور Instead of Trigger بجای دستورات Ins/Del/Upd فراخوانی و اجرا میشود. به‌عبارتی دیگر خود دستورات Ins/Del/Upd که سبب فراخوانی Trigger شده‌اند اجرا نشده (اما مقادیر تحت تاثیر این دستورات در جداول Inserted و Deleted جهت استفاده Trigger قرار خواهد گرفت) و فقط سبب فراخوانی Trigger خواهد شد.

فرمت کلی دستور Instead of Trigger به شکل زیر است:

```
Create Trigger Tr_name on Table_name
     Instead of [Insert], [Delete], [Update]
    As
       Begin
      .....
       End
    Go
```

مثال ۱: فرض کنید دو جدول Test1 و Test2 با فیلدهای مشابه Code int و Name varchar(50) داشته‌باشیم. در برنامه زیر نشان می‌دهد چنانچه بخواهیم رکوردی به جدول Test1 اضافه کنیم این رکورد بجای اینکه در این جدول اضافه شود در جدول Test2 اضافه می‌گردد. (این مثال اصلاً کاربردی نبوده و صرفاً برای درک این نوع از Trigger میباشد.)

```
Create Trigger Chg_Ins on Test1
  Instead of insert
  As
  Begin
  Insert into Test2 Select * from inserted
  End
```

نکته ۱: بیشترین استفاده Instead of Trigger ها در کنترلها و Validation داده ها قبل از اضافه شدن به جداول می‌باشد.

نکته ۲: اگرچه امکان نوشتن چندین Trigger برای یک Ins/Del/Upd وجود داشت ولیکن برای هر رویداد Ins/Del/Upd روی جداول، صرفاً میتوان یک Instead of Trigger تعریف نمود.

نکته ۳: دستورات Trigger خاصیت Recursive ندارند یعنی اگر از داخل Trigger مثلاً دستور Insert اجرا شود مجدداً Trigger فعال نمی‌شود.

مثال ۲: جدول Users با فیلدهای Code int p.k و Name varchar(50) و Pwd varchar(50) را درنظر بگیرید. Trigger ای بنویسید که هیچیک از مقادیر Pwd که با دستور Insert در جدول قرار میگیرد کمتر از ۶ حرف نباشد. به‌عبارتی دیگر اگر طول Pwd حتی یک رکورد کمتر از ۶ حرف بود هیچ رکوردی به جدول اضافه نشود.

```
Create Trigger Ins_Pwd on Users
   Instead of insert
   As
   Begin
   if exists ( Select * from inserted where Len(Pwd)<6)
        Raiserror('Wrong Password',16,1)
   else
        Insert into Users Select * from inserted
   End
```

مثال ۳: فرض کنید یک جدول بنام Test با فیلدهای Code int و Name varchar(50) و DelFlag tinyint داشته باشیم. اگر قرار شد از جدول Test یک رکورد حذف شود قرار نیست بار اول حذف شود بلکه دفعه اول، فقط فیلد DelFlag برابر ۱ شده و رکورد بصورت واقعی حذف نمیشود ولیکن در دفعه دوم که قبلاً فیلد DelFlag آن قبلاً ۱ شده است، رکورد بصورت فیزیکی حذف خواهد شد.

```
Create Trigger Del_Rec on Test
   Instead of delete
   As
   Begin
   Delete from Test where Code IN
        (Select Code from deleted where Delflag=1)
   update test set Delflag=1 where Code IN
        (Select Code from deleted where Delflag<1)
   End
```

۱۳. دستور Transaction:

تعریف: یکی از مهمترین دستورات و امکانات SQL، جلوگیری از ثبت اطلاعات ناقص در دیتابیس میباشد. تعریف چندین دستور بصورت Transaction این امکان را بوجود خواهد آورد. چنانچه یک یا چندین دستورالعمل دارای چهار خاصیت زیر (مخفف آنها را ACID مینامند) باشند اصطلاحاً به آنها Transaction گویند:

- Atomicity

یعنی چند دستور بصورت بهم پیوسته در نظر گرفته شده و همانند یک دستور فرض میشود یعنی اگر یک یا چند دستور از آنها اجرا شده و الباقی اجرا نشوند در حقیقت مجموعه دستورات کامل انجام نشده و عملیات دستورات اجرا شده نیز برگشت داده خواهد شد.

- Consistency

یعنی همواره باید دیتابیس از حالت منطقی پیشین به حالت منطقی جدید منتقل شود.

- Isolation
- Durability

یعنی هر عملی کامل و موفق اجرا شده است که کلیه اطلاعات از حافظه موقت به حافظه دایم منتقل شوند.

- Inplicit Transaction: هرکدام از دستورات SQL به تنهایی یک Transaction بوده و لذا کلیه خواص ACID را دارا میباشند اگر مثلاً وسط اجرای یک دستور پیچیده Select برق قطع شود یا سیستم با مشکل مواجه شود کلیه عملیات اجرا شده برگشت داده میشود. اصطلاحاً دستورات تکی SQL را Inplicit Transaction نامند.
- Explicit Transaction: اگر چندین دستور SQL تشکیل یک Transaction داده باشند به آنها Explicit Transaction گویند و فرمت آن به شکل زیر است. در دستور زیر اگر شرط cond1 برقرار باشد باعث اتمام کار کلیه دستورات Transaction و ثبت اطلاعات مرتبط با آنها میگردد و در غیر اینصورت باعث بازگشت نتیجه عملیات کلیه دستورات Transaction خواهد شد. لازم به توضیح است چنانچه در وسط عملیات یک Transaction وقفه‌ای سخت افزاری از قبیل قطع برق بوجود آید SQL بصورت خودکار به قبل از دستورات آن Transaction برگشته و عمل Rollback را انجام خواهد داد.
  - Begin Transaction
    - ....
    - ....
    - If cond1
    - Commit Transaction
    - Else
    - Rollback

- Nested Transaction: چنانچه چندین Explicit Trans داخل یکدیگر تعریف شوند به آن Nested Transaction گویند. در مثال زیر اگر عملیات Transaction داخلی ناقص اجرا شده و شرط cond1 برقرار نشود ولی عملیات Transaction اصلی بصورت کامل اجرا شده و شرط cond2 برقرار شود باعث میگردد که کلیه عملیات Transaction داخلی Rollback شود ولی عملیات Transaction اصلی مورد قبول واقع شده و Commit خواهد شد. ولیکن برعکس آن چنین نخواهد بود، بدین صورت که اگر عملیات transaction اصلی ناقص بوده و شرط cond2 برقرار نشود عملیات کلیه Transactionها مردود اعلام شده و Rollback خواهد شد.

```
Begin Transaction
    ....
    ....
      Begin Transaction
          ....
          ....
        If cond1
            Commit Transaction
        Else
            Rollback
    ....
    ....
    If cond2
      Commit Transaction
    Else
      Rollback
```

Trans داخلی

Trans اصلی

مثال: در مثال زیر Transaction اصلی مربوط به محاسبه حقوق بوده و Transaction فرعی مربوط به صدور سند حسابداری حقوق میباشد. چنانچه به هر دلیلی Transaction حقوق Commit نشده و Rollback شود نباید سند حسابداری نیز صادر شود لذا باید Transaction دوم نیز Rollback شود:

```
Create Procedure Salary
    As
    Begin Transaction
        ....
        Exec Sal_Acc
        ....
        If cond2
            Commit Transaction
        Else
        Rollback
Go
```

```
Create Procedure Sal_Acc
    As
    Begin Transaction
        ....
        ....
        If cond2
            Commit Transaction
        Else
        Rollback
Go
```

۱۴. قفل کردن رکوردها (Locking) :

تعریف: یکی دیگر از تواناییهای SQL، امکان قفل کردن رکوردها یا جداول در حال بروزرسانی میباشد. فرض کنید سه خاصیت A، C و D از خواص ACID (در قسمت قبل به تفصیل توضیح داده شد) را داشته باشیم ولیکن امکان Loking یا قفل شدن را نداشته باشیم در اینصورت چهار مشکل زیر بوجود خواهد آمد:

- Lost Updates

در زمان بروزرسانی همزمان یک رکورد توسط چند نفر، ممکن است تغییرات کاربران توسط تغییرات ثبت شده نفر آخر از بین برود.

- Uncommitted Dependency (Dirty Read)

اگر یکی از کاربران نتیجه بروزرسانی شده یک رکورد توسط یک Transaction کاربر دیگر را فراخوانی کرده و مورد استفاده قرار دهد ولیکن بنابه‌دلایلی آن Transaction ناقص اجرا شده و Rollback شود در نتیجه عملاً رکورد خوانده شده توسط کاربر دوم صحیح نمیباشد.

- Inconsistent Analysis (Nonrepeatable Read)

در مواقعی که یک فیلد بیش از یکبار باید خوانده شود و در بین دفعات خواندن ، مقدار فیلد توسط کاربر دیگری تغییر نماید در نتیجه مقدار فیلد در دفعات مختلف متفاوت خواهد بود بنابراین تحلیل داده ای با خطا مواجه میگردد.

- Phantom Reads

این اشکال در زمانی اتفاق می افتد که جهت اضافه کردن ، حذف کردن و یا بروزرسانی توسط چند کاربر تداخل بوجود آید مثلاً دو کاربر همزمان بخواهند یک سند جدید تولید نمایند و لذا ممکن است شماره سند تکراری شود.

Isolation Level : جهت حل مشکلات چهارگانه فوق ، میتوان در زمان تعریف Transaction نوع قفل کردن رکوردها و یا اصطلاحاً سطح Isolation را مشخص نمود. بدیهی است اگرچه قفل کردن تمام رکوردهای جدول ساده ترین راه برای جلوگیری از بروز اشکالات فوق است ولیکن اینکار باعث معطلی سایر کاربران و درنتیجه کند شدن سیستم خواهد شد و لذا SQL برای ایجاد تعادل ، انواع قفل نمودن رکوردها را در جدول زیر تعیین نموده است و برنامه نویس باید سعی نماید پائین ترین سطح Locking را اعمال نماید. لازم بتوضیح است Transaction اولین ایراد فوق الذکر (Lost Updates) را پوشش خواهد داد و یا بعبارتی دیگر با تعریف Transaction اشکال Lost Updates خودبخود برطرف خواهد شد ولذا در جدول زیر ذکر نشده است :

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	Yes
Serializable	No	No	No

فرمت Isolation : فرمت تعریف سطح قفل نمودن رکوردهای جدول به شرح زیر است که باید بجای عبارت Iso_Lev یکی عبارات ردیفهای جدول فوق ذکر شود :

```
Set Transaction Isolation Level   Iso_Lev
    Begin Transaction
    . . .
```

مثال ۱ : در مثال فرضی زیر ، اگر جدول TA توسط کاربر دیگری قفل نشده باشد ، در هنگام اجرای دستور Select قفل خواهد شد و سپس جدول TB نیز اگر آزاد باشد در زمان اجرای دومین دستور Select قفل میگردد. (بنابراین جداول در زمان شروع Transaction قفل نمی شوند بلکه در زمان اولین فراخوانی و دسترسی به جدول قفل میشوند) ولیکن با اجرای دستور Commit یا پایان اجرای Rollback آزاد خواهند شد.

```
Create Procedure Sp1
        As
        Set Transaction Isolation Level Serializable
            Begin Transaction
            Select * from TA
            Select * from TB
            If  Cond1
               Commit Transaction
            Else
               Rollback Transaction
```

مثال ۲ : فرض کنید برنامه Sp1 مثال قبل و Sp2 مثال زیر (در این مثال ، دستورات جداول TA و TB جابجا شده است) توسط دو کاربر جداگانه و بطور همزمان اجرا شود. در ابتدا فرض شده است که جداول TA و TB آزاد باشند. کاربر اول جدول TA و کاربر دوم جدول TB را دراختیار گرفته و قفل میکند و هر کدام جهت دراختیار گرفتن جدول بعدی منتظر می مانند چون کاربر اول نیاز به جدول TB دارد که توسط کاربر دوم قفل شده و کاربر دوم نیز متقابلاً به جدول TA نیاز دارد که توسط کاربر اول قفل شده است بنابراین حالت Deadlock اتفاق می افتد.

```
Create Procedure Sp2
    As
    Set Transaction Isolation Level Serializable
        Begin Transaction
        Select * from TB
        Select * from TA
        If  Cond1
           Commit Transaction
        Else
           Rollback Transaction
```

توضیح : برای اینکه مشکل Deadlock اتفاق نیفتاد میتوان یکی از دو راهکار زیر را انتخاب کرد:

- ترتیب دراختیار گرفتن جداول در برنامه های مختلف ، یکسان باشد.
- از یک جدول کمکی استفاده کرد. مثلاً با استفاده از دستور Select * from C در ابتدای هر دو برنامه ، هرکدام از کاربران که زودتر این دستور را اجرا نماید میتواند برنامه را ادامه داده و کاربر دیگر در انتظار میماند تا کار کاربر قبلی تمام شود.

نکته : در SQL فقط رکوردهائی که در دستور Select مورد دسترسی قرار میگیرد قفل میشوند. مثلاً اگر در دستور Select از Where استفاده شود تنها رکوردهائی که شرط Where برای آنها صدق میکنند قفل خواهند شد و الباقی رکوردها آزاد خواهند بود.

۱۵. کنترل خطا (Error Handling) :

تعریف : در SQL Server 2005 جهت کنترل خطا میتوان از روش زیر استفاده کرد. در این حالت اگر یکی از دستورات داخل بلوک Try با خطا مواجه شود کنترل برنامه به ابتدای بلوک Catch منتقل میشود و در صورتی که دستورات داخل بلوک Try بدون خطا اجرا شوند دستورات بلوک Catch اجرا نخواهند شد.

```
. . .
Begin Try
. . .
End Try
Begin Catch
. . .
End Catch
. . .
```

در داخل بلوک Catch ، توابع زیر قابل استفاده میباشند :

- تابع Error_number() : شماره خطا
- تابع Error_message() : شرح لاتین خطا
- تابع Error_severity() : میزان اهمیت خطا (بین ۱ تا ۲۵)
- تابع Error_state() : وضعیت یا حالت خطا (بین ۱ تا ۱۲۷)
- تابع Error_line() : شماره خطی از بلوک Try که خطا در آنجا اتفاق افتاده است.
- تابع Error_procedure() : نام برنامه ای که خطا در آنجا اتفاق افتاده است.

نکته : شماره میزان اهمیت (severity) در سه رده زیر دسته بندی میشود و چنانچه مقدار آن کمتر از ۱۱ باشد سطح خطا warning بوده و خطا جدی نمی باشد و لذا دستورات Catch اجرا نخواهد شد :

Severity	=	1 - 10	warning
		11 - 20	Error
		21 - 25	Fatal Error

مثال : تمرین Testins3 (مثال ۳ فصل ۱۷ در صفحه ۶۰) ممکن است مشکل Phantom Reads را بوجود آورد و لذا برای جلوگیری از این خطا ، برنامه به شکل زیر اصلاح میگردد :

```
Create Proc Testins4(@name varchar(50),@code int output,@Errno int output)
      As
      Begin
            Declare @ncode int
            Set Transaction Isolation Level Serializable
            Begin Transaction
                  Begin Try
                        Set @ncode=(Select Max(Code) from Test)
                        Set @ncode=isnull(@ncode,0)+1
                        Insert into Test values(@ncode,@Name)
                        Set @code=@ncode
                        Set @Errno=0
                        Commit Transaction
                  End Try
                  Begin Catch
                        Rollback Transaction
                        Set @errno=Error_number()
                  End Catch
            Set Transaction Isolation Level Read Committed
      End
      Go
```

۱۶. دستور Cursor:

تعریف : این دستور در هنگامی مورد استفاده قرار میگیرد که بخواهیم عملیات بر روی گروه خاصی از رکوردها انجام شود. فرمت استفاده از این دستور بشکل زیر است. با دستور Open فضای موردنیاز اختصاص داده شده و با دستور Fetch محتوای رکوردها داخل آن فضا قرار گرفته و اشاره گر X به آن مجموعه از رکوردها اشاره میکند. با اجرای دستور Close فضا آزاد شده و دستور Deallocate اعتبار اشاره گر را ملغی می کند :

```
Declare X Cursor for Select . . . from . . .
Open X
. . .
Fetch next from X into . . .
. . .
Close X
Deallocate X
```

مثال ۱ : بعنوان مثال دستورات زیر ، نام و نام خانوادگی نویسندگان را نمایش می دهد :

```
Declare X Cursor for Select au_fname,au_lname,au_id from authors
Declare @au_fname  varchar(50),
        @au_lname  varchar(50),
        @au_id          varchar(11)
Open X
Fetch next from X into @au_fname,@au_lname,@au_id
While  @@fetch_status=0
        Begin
        Print @au_fname,@au_lname,@au_id
        Fetch next from X into @au_fname,@au_lname,@au_id
        End
Close X
Deallocate X
```

مثال ۲ : با استفاده از Cursor برنامه ای بنویسید (مثال ۳۰ از فصل ۱۲ در صفحه ۳۶) که نام و قیمت کتاب و قیمت جدید را که براساس ضابطه زیر محاسبه میشود را لیست کند.

- اگر قیمت کتاب بیش از ۲۰ دلار بود را ۱۰٪ افزایش دهد.
- درغیر اینصورت ۵٪ کاهش دهد.

```
Declare X Cursor for Select title_id,Price from Titles
Declare @Title_id  nvarchar(6),@Price  money
Open X
Fetch next from X into @title_id,@Price
While  @@fetch_status=0
       Begin
       if  @price>20
             Set @price=@price*1.10
       else
             Set @price=@price*0.95
       Update Titles Set Price=@price where Title_id=@Title_id
       Fetch next from X into @title_id,@Price
       End
Close x
Deallocate X
```

مثال ۳ : فرض کنید جدول زیر شامل فیلدهای Code و Score جهت ذخیره سازی شماره حساب و امتیاز دارندگان حساب قرض الحسنه وجود داشته باشد. در این جدول فیلد Taj مقدار تجمیعی امتیاز دارندگان حساب میباشد. بکمک دستور Cursor برنامه ای بنویسید که این فیلد را محاسبه و رکوردهای جدول را بروزرسانی کند.

Code	Scare	Taj
1	110	
2	2000	
3	50	
4	120	

Code	Scare	Taj
1	110	110
2	2000	2110
3	50	2160
4	120	2280

```
Declare X Cursor for Select Code,Scare from Account
Declare @Code  nchar(10),@Scare  int,@taj  money
Open X
Set @Taj=0
Fetch next from X into @Code,@Scare
While  @@fetch_status=0
      Begin
      Set @taj=@Scare+@Taj
      Update Account Set Taj=@Taj where Code=@Code
      Fetch next from X into @Code,@Scare
      End
Close x
Deallocate X
```