بسم الله الرحمن الرحیم

مدیریت پایگاه داده

(جلد دوم)

زمینه‌ی خدمات

شاخه‌ی: کاردانش

گروه تحصیلی:کامپیوتر

زیرگروه: کامپیوتر

رشته‌ی مهارتی:برنامه‌نویسی پایگاه داده

شماره‌ی رشته مهارتی: ۳۱۵-۱۰۱-۱۷-۳

کد رایانه‌ای رشته‌ی مهارتی: ۶۱۴۰

نام استاندارد مهارت مبنا: کاربر بانک اطلاعاتی access و SQL server

کد استاندارد متولی:۰-۸۴/۸۰/۱/۳/۲

شماره‌ی درس:نظری:۰۴۹۲ عملی:۰۴۹۳

۱۳۸۹

جباریه،علیرضا، ۱۳۴۷-

مدیریت پایگاه داده‌ها جلد اول/علیرضا جباریه، برنامه‌ریزی محتوا و نظارت بر تألیف دفتر برنامه‌ریزی و تألیف آموزش‌های فنی و حرفه‌ای و کاردانش وزارت آموزش و پرورش - تهران: انتشارات جاریان ۱۳۸۹

۳۱۵ص: مصور. - (شاخه‌ی کاردانش; شماره‌ی درس ۰۴۹۲ و ۰۴۹۳)

متون درسی زمینه‌ی‌خدمات، شاخه‌ی کاردانش،گروه تحصیلی‌کامپیوتر زیرگروه کامپیوتر رشته‌ی مهارتی برنامه‌نویسی پایگاه داده‌ها.

۱.پایگاه‌های اطلاعاتی. ۲.پایگاه‌های اطلاعاتی - مدیریت. لف.ایران. وزارت آموزش و پرورش. دفتر برنامه‌ریزی و تألیف آموزش‌های فنی وحرفه‌ای و کاردانش. ب.عنوان،

۱۳۸۹

همکاران محترم ودانش‌آموزان عزیز:

پیشنـهادات و نـظرات خود را درباره‌ی محتوای این‌کتاب به نشانی تهران–صندوق پستی شماره‌ی ۴۸۷۴/۱۵ دفتر برنامه‌ریزی و تألیف آموزش‌های فنی وحرفه‌ای و کاردانش، ارسال فرمایید.

پیام نگار(ایمیل) tvoccd@medu.sch.ir

وب‌گاه (وب‌سایت) www.tvoccd medu.ir

وزارت آموزش و پرورش

سازمان پژوهش و برنامه‌ریزی آموزشی

برنامه‌ریزی محتوا و نظارت بر تالیف: دفتر برنامه‌ریزی و تالیف آموزش‌های فنی و حرفه‌ای کاردانش

عنوان و کد کتاب: مدیریت پایگاه داده جلد دوم/ ۶۱۲/۱۲

مؤلف: دکترعلیرضا جباریه

مجری: انتشارات جاریان

شماره‌ی درس: نظری:۰۴۹۲ عملی:۰۴۹۳

ویراستار ادبی:دکتر حسین داوودی

صفحه‌آرا: نسرین اصغری- الناز نفری

طراح جلد: نسرین اصغری

محتوای این کتاب درنوزدهمین جلســه‌ی مورخ۸۹/۴/۸کمیســیون تخصصی رشــته‌ی کامپیوتر دفتر برنامه‌ریزی و تالیف آموزش‌های فنی و حرفه‌ای و کاردانش با عضویت:بتول عطاران، محمدرضا شــکرریز، سید حمیدرضا ضیایی، افشین اکبری، فرنگیس شاکری، حسن رحیمی مقدم تایید شده است.

ناشر: انتشارات جاریان

نوبت وسال چاپ: چاپ اول ۱۳۸۹

تهران- خیــابان آب‌شناسان- خیابان‌شهیدحاجی خیابانی- کوچه دهم- پلاک ۲۳

تلفن: ۴۴۳۰۳۷۹۲

نظارت برچاپ وتوزیع: اداره‌ی کل چاپ و توزیع کتاب‌های درسی

تهران: خیابان ایرانشهر شمالی- ساختمان شماره‌ی ۴ آموزش و پرورش(شهید موسوی)

تلفـن: ۹-۸۸۸۳۱۱۶۱ ، دورنگـار: ۸۸۳۰۹۲۶۶ ، صندوق پستی: ۱۵۸۴۷۴۷۳۵۹

وب سایت www.chap.sch.ir

چاپخانه: شرکت چاپ و نشر کتاب‌های درسی

ایران:تهران–کیلومتر ۱۷جادّه‌ی‌مخصوص‌کرج–خیابان ۶۱(داروپخش)-تلفــن: ۵- ۴۴۹۸۵۱۶۱، دورنگار:۴۴۹۸۵۱۶۰، صندوق‌پستی:۱۳۴۴/۶۸۴

شابک ۱-۱۸۴۴-۰۵-۹۶۴ ISBN **964-05-1844-1**

اگر بخواهید عزیز و سربلند باشید باید از سرمایه‌های عمر و استعداد جوانی استفاده کنید و با اراده و عزم راسخ خود به طرف علم و عمل و کسب دانش و بینش حرکت نمایید که زندگی زیر چتر علم و آگاهی آن‌قدر شیرین و انس با کتاب و قلم و اندوخته‌ها آن‌قدر خاطره‌آفرین و پایدار است که همه‌ی تلخی‌ها و ناکامی‌های دیگر را از یاد می‌برد.

امام خمینی

عنوان	صفحه

عنوان	صفحه

عنوان	صفحه

عنوان	صفحه

عنوان	صفحه

عنوان	صفحه

توانایی شناخت SQL Server

هدف های رفتاری

هنرجو پس از پایان این فصل، قادر خواهد بود:

- مفهوم SQL Server و کاربرد آن را بیان کند؛
- به سرور پایگاه داده متصل شود؛
- بخش‌های مختلف ابزار Management Studio را شرح دهد؛
- به کمک ابزار Management Studio بتواند بانک اطلاعاتی ایجاد کند؛
- به کمک ابزار Management Studio بتواند مشخصات بانک اطلاعاتی و جدول‌های موجود را ویرایش کند؛
- ارتباط بین جدول‌ها را برقرار کند؛
- واژه ها و متن انگلیسی مربوط به محتوای واحد کار را توضیح دهد.

۱-۸ آشنایی با SQL SERVER و کاربرد آن

سیستم مدیریت بانک اطلاعاتی SQL SERVER یکی از سیستم‌های بانک اطلاعاتی رابطه‌ای است که قدرت فوق‌العاده‌ای دارد. نسخه‌های گوناگونی از این نرم‌افزار ارائه شده است، آن چه که در این کتاب مورد بررسی قرار گرفته است، SQL SERVER 2008 است.

SQL SERVER نسبت به سایر نرم‌افزارهای رقیب، اغلب بانک‌های اطلاعاتی موجود را به خود اختصاص داده است.این نرم افزار یک سیستم بانک اطلاعاتی سرویس‌گیرنده / سرویس‌دهنده است و می تواند به بانک‌های اطلاعاتی کوچک و بزرگ سرویس ارائه دهد. چندین کاربر می‌توانند همزمان به این نرم افزار متصل شوند و هر کدام از سرویس دهنده درخواست‌هایی مانند به هنگام سازی، حذف یا درج رکوردها را داشته باشند.این نرم افزار می‌تواند هم به صورت منفرد روی یک رایانه ی مستقل و هم به صورت شبکه‌ای مورد استفاده قرار گیرد.

البته به دلیل ویژگی هایی که دارد، بهتر است در محیط‌های شبکه‌ای و در طراحی وب سایت‌های پویا مورد استفاده قرار گیرد.توصیه می‌شود به دلیل توانایی SQL SERVER در پردازش تعداد زیادی از تراکنش‌ها، در محیط‌های چند کاربره از این سیستم و در محیط‌های تک کاربره از اکسس استفاده شود.

برای مشاهده‌ی گروه برنامه‌های SQL Server، از منوی Start روی All Programs و سپس Microsoft SQL Server 2008 کلیک کنید[۱]. گروه برنامه‌ی SQL Server شامل تمام برنامه‌های کاربردی است که در طول کار با این سیستم، مورد استفاده قرار خواهید داد.

۱- فرض بر این است که SQL SERVER 2008 روی رایانه نصب شده است. مراحل نصب را می‌توانید در پیوست الف مشاهده کنید.

تحقیــق: با جســت وجو در اینترنت یــا کتاب‌های مرجع، وجوه تشــابه و تمایز SQL Server و Oracle را یادداشت کنید.

۲-۸ آشنایی با SQL Server Management Studio

SQL Server 2008 ابزارهــای مختلفی را ارائــه می‌کند که برای اهداف متفاوتی مثل نصب، پیکربندی، بازبینی و بهینه‌سازی کارآیی، مورد استفاده قرار می‌گیرند.

ابزار اولیه ی مدیر برای تعامل با سیستم، SQL Server Management Studio است. هــم مدیران و هم کاربــران می‌توانند از این ابزار برای مدیریت چندین ســرور، ایجاد پایگاه داده‌ها و موارد دیگری اســتفاده کنند. برای اجرای این ابــزار، از منوی Start، گزینه‌ی All Programs و Microsoft SQL Server 2008 ســپس SQL Server Management Studio را انتخاب کنید. هر کاربری نیز می تواند با دسترســی به سرور پایگاه داده‌ی خاصی از این ابزار استفاده کند.

این ابزار شــامل چهار جزء[1] متفاوت اســت که برای ایجاد و مدیریت کل سیســتم مورد استفاده قرار می‌گیرند. اجزای اصلی این ابزار، عبارت‌اند:

- Registered Servers
- Object Explore
- Query Editor
- Solution Explorer

هر کدام از این اجزاء در ادامه شــرح داده می‌شــوند. برای دسترسی به رابط اصلی، ابتدا باید به سرور متصل شوید.

۱-۲-۸ اتصال به سِرور

هنگامــی کــه SQL Server Management Studio را اجــرا کنیــد، کادر محاوره‌ای

۱- دو جز اول در این فصل و سایر اجزاء در ادامه‌ی کتاب شرح داده خواهند شد.

Connect to Server نمایش داده می‌شود (شکل ۱-۸). این کادر، تعیین پارامترهای ضروری را برای اتصال به سِرور امکان پذیر می‌کند:

- Server Type- برای اهداف این فصل Database Engine را انتخاب کنید.
- Server Name- نام سِروری را که می‌خواهید استفاده کنید، تایپ یا انتخاب کنید (به طور کلی، می‌توان SQL Server Management Studio را به محصول نصب شده‌ای روی سرور خاص، متصل کرد).

شکل (۱-۸) کادر محاوره‌ای Connect to Server

- Authentication- یکی از دو نوع شناسایی کاربر زیر را انتخاب کنید:

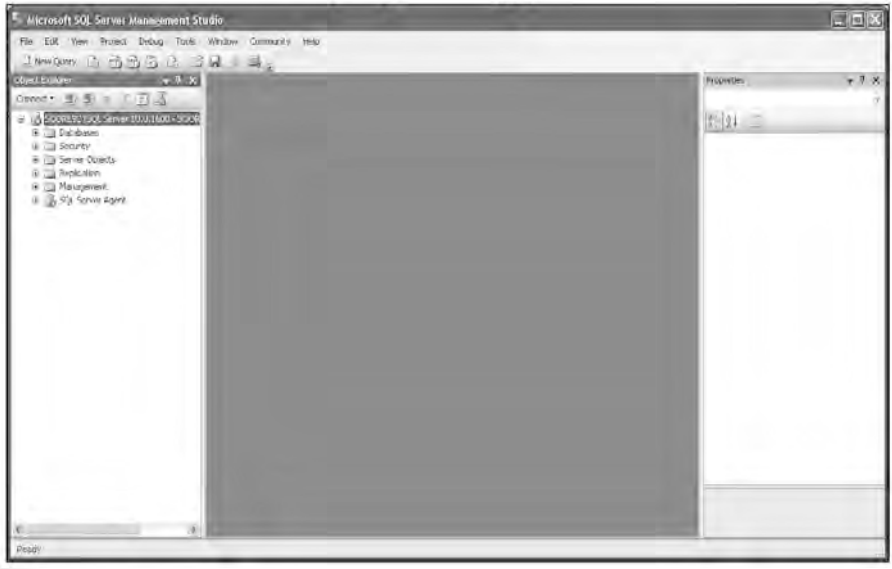
۱- Windows Authentication :

اتصال SQL Server با استفاده از حساب کاربری ویندوز. این گزینه ساده‌تر است و به وسیله‌ی مایکروسافت توصیه می‌شود.

۲- SQL Server Authentication:

Database Engine از شناسایی خاص خودش استفاده می‌کند.

هنگامی که روی Connect کلیک کنید، Database Engine به سرور تعیین شده متصل می‌شود. بعد از اتصال به سرور پایگاه داده، پنجره‌ی SQL Server Management Studio ظاهر می‌شود (شکل ۲-۸). ظاهر پیش‌فرض، شبیه Visual Studio است، بنابراین، کاربران می‌توانند از تجربه ی خودشان در Visual Studio استفاده کنند تا به سادگی پایگاه داده‌ها را مدیریت کنند.



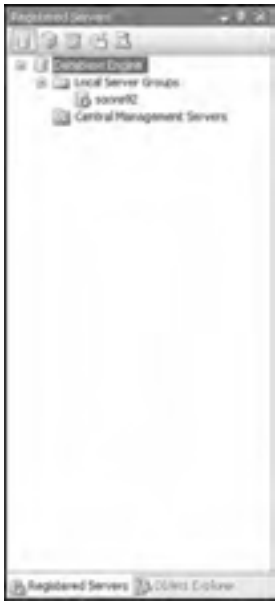
شکل(۲-۸) SQL Server Management Studio: تنظیمات پنجره‌ی پیش فرض

نکته

SQL Server Management Studio یـک رابـط منحصر به فردی را برای مدیریت سرورها و ایجاد پرس‌وجوها از طریق تمام مؤلفه‌های SQL Server ارایـه می‌کنـد. این بدین معنی اسـت کـه SQL Server Management Studio رابطـی را بـرای ، Analysis Services ، Database Engine Integration Services و Reporting Services پیشنهاد می‌کند.

۸-۲-۲ سرورهای ثبت شده

Registered Servers کـه امکان نگه داری اتصال‌هایی را که قبلاً از سـرورها اسـتفاده کرده‌انـد، به صورت ناحیه‌ای نمایش می دهد. می‌توان از این اتصال‌ها برای کنترل وضعیت سـرور یا مدیریت شـیء‌های آن اسـتفاده کرد. هر کاربـری دارای لیسـت جداگانه‌ای از سـرورهای ثبت شده است که به طور محلی ذخیره می‌شوند (اگر این ناحیه قابل رؤیت نیسـت، از منوی View گزینه‌ی Registered Servers را انتخاب کنید).



می‌توان سـرورهای جدیدی را به لیسـت تمام سـرورها اضافـه کـرد یا آن هـا را از لیسـت حذف کـرد. هم چنین، می‌توان سـرورها را در گروه‌های سرور، دسته بندی کرد. هر گروه شامل سرورهایی است که با هم به طور منطقی مرتبط

هستند. می‌توان گروه سرورها را بر اساس نوع سرور نیز تعیین کرد.

Object Explorer ۸-۲-۳

این ناحیه شـامل یک نمای درختی از تمام شـیءهای پایگاه داده در سـرور است (اگر این ناحیـه قابل رؤیت نبود، از منوی View گزینه‌ی Object Explorer را انتخاب کنید). درخت، سلسله مراتبی از شیءهای موجود در سرور را نشان می‌دهد.

برای اتصال Object Explorer به یک سرور، روی نام سرور کلیک راست و Connect را انتخاب کنید. برای قطع اتصال، روی Disconnect در نوارابزار کلیک کنید.

Object Explorer امکان اتصال به چندین سرور را فراهم می‌کند.

نکته

Object Explorer چندیـن ویژگی دیگـر نیز دارد، که بعداً در این فصل شرح داده می‌شوند.

٤-۲-۸ سازماندهی و پیمایش نواحی Management Studio

هــر کدام از نواحی Management Studio را می‌تــوان تثبیت یا پنهان کرد. با کلیک راســت روی نوار عنوان در بالای ناحیه‌ی مربوطه، می‌توان یکی از وضعیت های نمایشی زیر را انتخاب کرد:

- Floating- هنگامــی که وضعیت پنجره، شــناور تعیین شــود، آن پنجره به صورت یک پنجره‌ی شــناور جداگانه در بالای بقیه ی پنجره‌هــای SQL Server Management Studio ایجاد می‌شود. این پنجره نیز مانند هر پنجره‌ی دیگری می‌تواند به هر جای صفحه جا به جا شود.
- Dockable- به شــما این امکان را می‌دهد که نواحی را جا به جا کنید و آن ها را در محل‌هــای مختلــف تثبیت کنید. برای جا به جایی یک مؤلفــه، روی نوار عنوان آن کلیک و درگ کنید.
- Tabbed Document- می‌تــوان یــک گــروه زبانــه‌دار را با اســتفاده از پنجره‌ی Designer ایجاد کرد.با ایجاد این گروه، وضعیت ناحیه از تثبیت شده به سند زبانه‌دار تغییر می‌کند.

Hide- پنجره را پنهان می‌کند. برای نمایش پنجره‌ی بســته شده، نام مؤلفه را از منوی View انتخاب کنید.

- Auto Hide- ناحیه را به حداقل می رساند و آن را در سمت چپ صفحه قرار می‌دهد. برای باز کردن دوباره‌ی چنین ناحیه‌ای، اشاره‌گر ماوس را روی زبانه‌ها در سمت چپ صفحه قرار دهید و روی آن ها کلیک کنید.

بــرای بازیابی پیکربنــدی پیش‌فرض، از منوی Window گزینــه‌ی Reset Window Layout را انتخاب کنید. ناحیه‌ی Object Explorer در ســمت چپ ظاهر می‌شــود. برای

مشاهده ی مشخصات یک شیء روی آن کلیک راست و Properties را انتخاب کنید.

۸-۳ استفاده از Management Studio به همراه Database Engine

SQL Server Management Studio دارای دو هدف اصلی است:

مدیریت سرورهای پایگاه داده

مدیریت شیءهای پایگاه داده

۸-۳-۱ مدیریت سرورهای پایگاه داده

وظایف مدیریتی که می توان با استفاده از SQL Server Management Studio انجام داد، عبارت اند:

- ثبت سرورها
- اتصال به یک سرور
- ایجاد گروه‌های سروری جدید
- شروع و خاتمه‌ی SQL Server

ثبت کردن سرورها

SQL Server Management Studio عملیات ثبت کردن سرورها و کاوش پایگاه داده‌ها و شیءهای آن را از هم جدا می‌کند. (هر دو عملیات، می‌توانند با استفاده از Object Explorer انجام شوند). هر سروری (محلی یا راه دور) باید قبل از استفاده، ثبت شود. هر سروری می‌تواند در طول اولین اجرای SQL Server Management Studio یا بعد از آن ثبت شود.

برای ثبت یک سرور پایگاه داده، روی سرور پایگاه داده در Object Explorer کلیک راست نمائید و Register را انتخاب کنید.در این صورت، کادر محاوره‌ای New Server Registration ظاهر می‌شود (شکل ۳-۸). نام سروری را که می‌خواهید ثبت شود و حالت

شناسایی (Windows یا SQL Server) را انتخاب کنید.

شکل(۳-۸) کادر محاوره‌ای New Server Registration

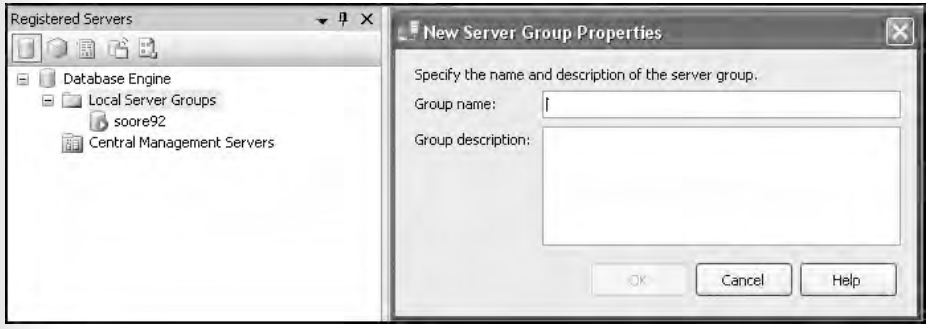
اتصال به سرور

SQL Server Management Studio وظایف ثبت کردن سرور و اتصال به سرور را از هم جدا کرده است. توضیح این که ثبت یک سرور به طور خودکار شما را به یک سرور متصل نمی‌کند. برای اتصال به سرور از طریق پنجره‌ی Object Explorer، روی نام سرور کلیک راست نمائید و Connect را انتخاب کنید.

ایجاد یک گروه سرور جدید

برای ایجاد گروه سرور در ناحیه‌ی Registered Servers، روی Local Server Groups

کلیک راســت نمائید و گزینه‌ی New Server Group را انتخاب کنید. در کادر محاوره‌ای مشــخصات New Server Group، نام یک گروه (منحصر به فرد) و شــرح گروه (اختیاری) را تایپ کنید.



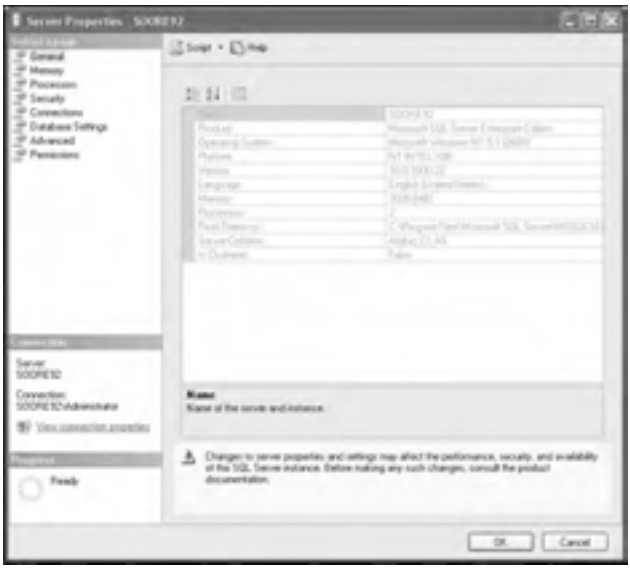
مدیریت سرورهای چندگانه

SQL Server Management Studio، بــا اســتفاده از Object Explorer امکان مدیریت چندین ســرور پایــگاه داده (نمونه‌ها نامیده می‌شــوند)را روی یــک کامپیوتر ارائه می‌کنــد. هــر نمونه از Database Engine مجموعه‌ای از شــیء‌های پایــگاه داده‌ی خاص خود را دارند (پایگاه داده‌های سیستم کاربر)، که بین نمونه‌های مختلف به اشتراک گذاشته نمی‌شوند.

بررســی کنید که روی رایانه‌های هنرستان، چند سرور پایگاه داده قابل دسترس است.

برای مدیریت سرور و پیکربندی آن، روی نام سرور درObject Explorer کلیک راست نمائید و Properties را انتخاب کنید. کادر محاوره‌ای Server Properties (شــکل ۴-۸)

شامل چندین صفحه‌ی مختلف مانند Security ، General و Permissions است.



(۴-۸) کادر محاوره‌ای Server Properties

صفحه‌ی General مشـــخصات کلی سرور را نشـــان می‌دهد. صفحه‌ی Security شامل اطلاعاتی درباره‌ی حالت تائید اعتبار سرور و حالت ورود است. صفحه‌ی Permissions تمام قواعد دسترســـی به سرور را نشان می‌دهد. بخش پایینی صفحه، تمام مجوزهایی که ورود را می‌توانند محدود کنند، نشان می‌دهد.

نام سرور جدیدی را جای گزین سرور موجود کنید. در پنجره‌ی Object Explorer روی ســـرور کلیک راســـت نمائید و Register را انتخاب کنید. اکنون می‌توانید نام سرور را تغییر دهید و شرح آن را در قاب Registered Server اصلاح کنید.

نکته

سـرورها را تغییـر نام ندهید، زیـرا تغییر نام می‌تواند روی سـایر سرورها که به آن ارجاع دارند، تأثیرگذارد.

شروع و خاتمه‌ی سرورها

سـرور Database Engine هر زمانی که سیسـتم عامل ویندوز شروع می‌شود یا SQL Server Management Studio، مورد اسـتفاده قرار گیرد، به طور خودکار شـروع شـود. برای شروع سرور با استفاده از Management Studio، روی سرور انتخاب شده در ناحیه‌ی Object Explorer کلیک راست نمائید و از منوی میانبر، روی Start کلیک کنید. این منو شـامل دسـتورات Stop و Pause نیز هست، که می‌توان سـرور فعال را خاتمه داد یا موقتاً قطع نمود.

۸-۳-۲ مدیریت پایگاه داده با استفاده از Object Explorer

وظایف مدیریتی که می‌توانید با استفاده از SQL Server Management Studio انجام دهید، عبارت‌اند:

- ایجاد پایگاه داده بدون استفاده از زبان Transact-SQL
- ویرایش پایگاه داده بدون استفاده از زبان Transact-SQL
- مدیریت شیءهای پایگاه داده و کاربرد آن‌ها
- تولید و اجرای دستورات (عبارات) SQL

ایجاد پایگاه داده‌ها بدون استفاده از Transact-SQL

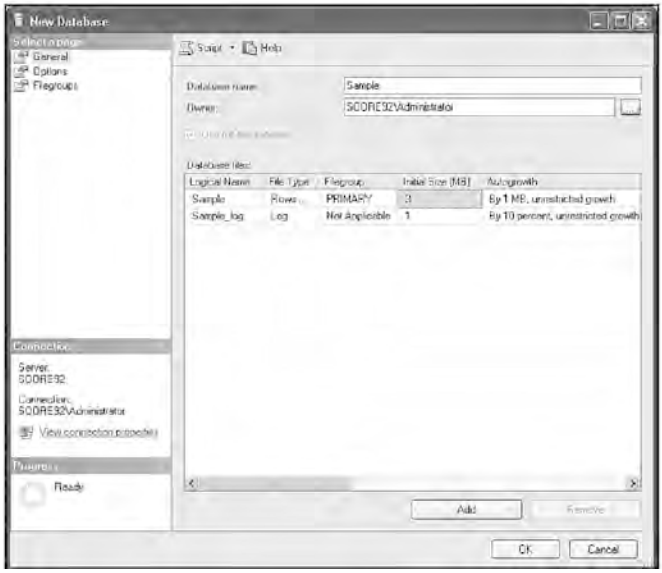
می‌توان یک پایگاه داده‌ی جدید را با استفاده از Object Explorer یا زبان Transact-SQL ایجاد کرد. از Object Explorer برای کاوش شیءهای داخل سرور استفاده می‌شود. از ناحیه‌ی Object Explorer می‌توان تمام شیءهای درون سرور را بررسی و سرور و پایگاه داده را مدیریت کرد. درختواره‌ی موجود، علاوه بر سایر پوشه‌ها، شامل پوشه‌ی Database نیز هست. این پوشه دارای چندین زیرپوشه است که یکی برای پایگاه داده‌های سیستم و دیگری برای هر پایگاه داده‌ی جدیدی است که به وسیله‌ی کاربر ایجاد می‌شود.

برای ایجاد پایگاه داده با استفاده از Object Explorer به شرح زیر اقدام کنید:

۱. روی Databases کلیک راست نمائید و New Database را انتخاب کنید.

۲. در کادر محاوره‌ای New Database (شکل ۵-۸)، نام پایگاه داده‌ی جدید را در فیلد Database Name تایپ کنید.

۳. روی Ok کلیک کنید.



شکل(۵-۸) کادر محاوره‌ای New Database

هــر پایگاه داده‌ای دارای چندین مشــخصه‌ی مختلــف مثل نوع فایل، انــدازه‌ی اولیه و غیر آن هاســت. مشــخصات پایــگاه داده را می‌توان از ناحیه‌ی چــپ کادر محاوره‌ای New Database انتخاب کرد. در این صورت، چندین صفحه مختلف (گروه‌های مشخصه)به شرح زیر ظاهر می شوند:

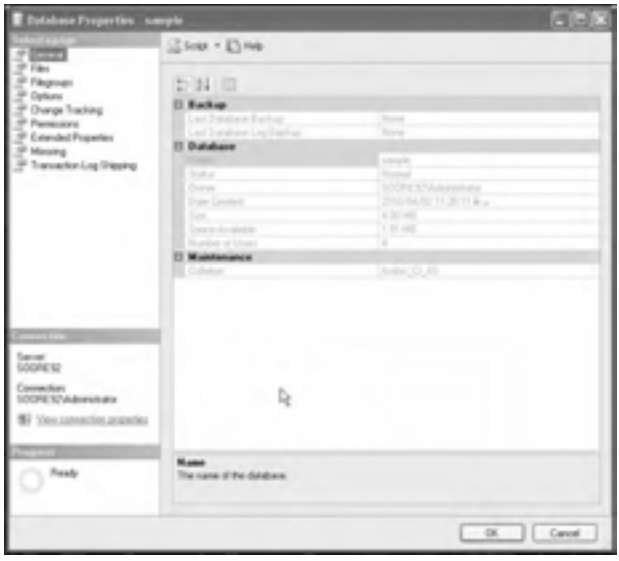
- General
- Files (فقط برای یک پایگاه داده‌ی موجود ظاهر می‌شود)
- Filegroups
- Options
- Permissions (فقط برای پایگاه داده‌ی موجود ظاهر می‌شود)
- Extended Properties (فقط برای پایگاه داده‌ی موجود ظاهر می‌شود)

- Mirroring (فقط برای پایگاه داده‌ی موجود ظاهر می‌شود)
- Transaction Log Shipping (فقط برای پایگاه داده‌ی موجود ظاهر می‌شود)

نکته

سیستم، تمام گروه‌های مشخصه ی لیست شده را برای پایگاه داده‌ی موجود نمایش می‌دهد. برای یک پایگاه داده‌ی جدید، همان طوری که در شکل ۵-۸ نشان داده شده است، سه گروه متفاوت وجود دارد: ، General Options و Filegroups.

صفحه‌ی General از کادر محاوره‌ای Database Properties (شکل ۶-۸)، نام پایگاه داده، مالک پایگاه داده و غیر آن را نمایش می‌دهد. مشخصات فایل‌های داده‌ای که به پایگاه داده‌ی خاصی تعلق دارند، شامل نام و اندازه‌ی فایل، محل ذخیره سازی پایگاه داده و نوع فایل (برای مثال Primary) هستند. پایگاه داده می‌تواند در چندین فایل ذخیره شود.



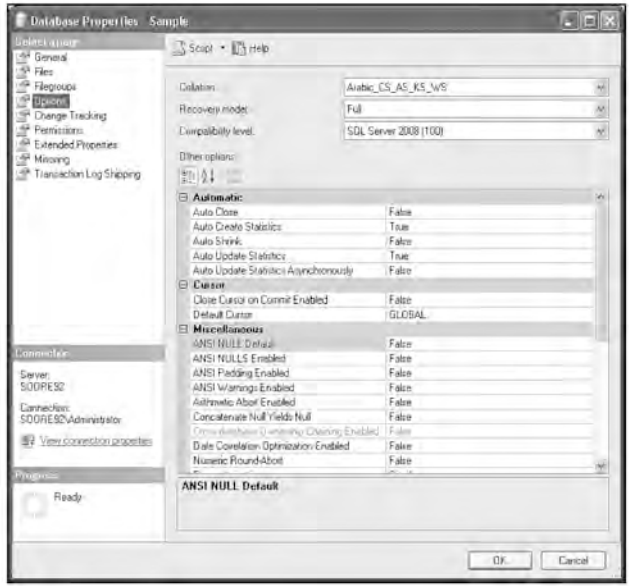
شکل (۶-۸) کادر محاوره‌ای Database Properties: صفحه‌ی General

صفحه‌ی Filegroups از کادر محاوره‌ای Database

Properties نام گروه(های) فایل را،که فایل پایگاه داده به آن تعلق دارد، نمایش می‌دهد.

صفحه‌ی Options از این کادر محاوره‌ای امکان نمایش و اصلاح تمام گزینه‌های سطح پایگاه داده را فراهم می‌کند. چندین گروه از گزینه‌ها وجود دارند که عبارت‌اند: Automatic، Recovery، Miscellaneous، Cursor و State. برای مثال، چهار گزینه‌ی زیر برای State وجود دارد:

- Database Read-Only- امکان دسترسی فقط خواندنی به پایگاه داده را می‌دهد. کاربران را از اصلاح هر داده‌ای منع می‌کند (مقدار پیش فرض، False است).
- Database State- وضعیت پایگاه داده را شرح می‌دهد (مقدار پیش فرض، Normal است).
- Restrict Access - استفاده از پایگاه داده در هر لحظه را فقط برای یک کاربر محدود می‌کند (مقدار پیش فرض، MULTI_USER است).
- Encryption Enabled- وضعیت کدگذاری پایگاه داده را کنترل می‌کند (مقدار پیش فرض، False است).



نکته

برای این کــه پایگاه داده بتواند داده‌های فارســی را ذخیره کند، در صفحه‌ی Options از کادر لیســت Collation گزینه‌ی Arabic_CS_AS_KS_WS را انتخاب کنید.

اگــر صفحه‌ی Permissions را انتخاب کنید، سیســتم، کادر محــاوره‌ای مربوطه را باز می‌کند و تمام کاربران و قواعد مربوط به مجوزهای آن ها را نمایش می‌دهد.

ویرایش پایگاه داده‌ها بدون استفاده از Transact-SQL

از Object Explorer می‌توان برای ویرایش یک پایگاه داده‌ی موجود نیز استفاده کرد. بــا اســتفاده از این مؤلفه، می‌توان فایل‌ها و گروه‌های فایــل مرتبط به پایگاه داده را ویرایش کرد. برای اضافه کردن فایل‌های داده‌ای جدید، روی نام پایگاه داده کلیک راســت نمائید و Properties را انتخــاب کنید. در کادر محاوره‌ای Database Properties، گزینه‌ی Files را انتخاب و روی Add کلیک کنید و نام فایل جدید را تایپ نمائید. هم چنین، می‌توان یک گروه فایل را برای پایگاه داده با انتخاب File Groups و کلیک روی Add، اضافه کرد.

نکته

فقط مدیر سیســتم یا مالک پایگاه داده می‌تواند مشخصات پایگاه داده را اصلاح کند.

بـــرای حذف یک پایگاه داده بـــا اســـتفاده از Object Explorer، روی نام پایگاه داده کلیک راست و Delete را انتخاب کنید.

مدیریت جدول ها بدون استفاده از Transact–SQL

بعـــد از ایجاد پایگاه داده، عمل بعدی ایجاد تمام جدول‌های مرتبط به آن اســـت. دوباره می‌توان جدول‌ها را با اســـتفاده از Object Explorer یـــا Transact-SQL ایجاد کرد. در این جا نیز فقط Object Explorer شرح داده می‌شود.

برای ایجاد جدول، با اســـتفاده از Object Explorer، پوشه‌ی Database را باز و روی زیرپوشه‌ی Tables کلیک راست نمائید و سپس روی New Table کلیک کنید. ایجاد یک جدول و سایر شیء‌های پایگاه داده با استفاده از زبان Transact-SQL، در فصل های بعدی شرح داده خواهد شد.

برای تشریح ایجاد جدول، با اســـتفاده از Object Explorer، جدول Department از پایگاه داده‌ی شـــرکت پیمانکاری را (که در فصل اول مورد بررســـی قرار دادیم)، برای مثال ایجاد می‌کنیم. اسامی تمام ستون‌ها را به همراه مشخصات آن ها وارد کنید. همان طوری که در شکل ۷-۸ نشان داده شده است،باید اسامی ستون‌ها و انواع داده‌هایشان، به همین ترتیب مشخصه‌ی NULL ستون در سطرها و ستون‌ها، وارد شوند.

تمام انواع داده‌هایی[1] که به وســـیله‌ی سیســـتم پشتیبانی می‌شوند، با کلیک روی علامت فلش در ســـتون Data Type نمایش داده می شـــوند. به همین ترتیب، می‌توان ورودی‌ها را در ســـطرهای Precision ، Length و Scale برای انتخاب نوع داده در زبانه‌ی Column Properties تایپ کرد. بعضی از انواع داده‌ها مثل char نیاز به مقدار برای ســـطر Length و بعضی مثل DECIMAL نیاز به مقدار در ســـطرهای Precision و Scale دارند. به عبارت دیگر، انواع داده‌هایی مثل INTEGER برای تعیین شدن به هیچ کدام از این ورودی‌ها نیاز ندارند.

۱- انواع داده‌ها در واحدکار بعدی شرح داده خواهند شد.

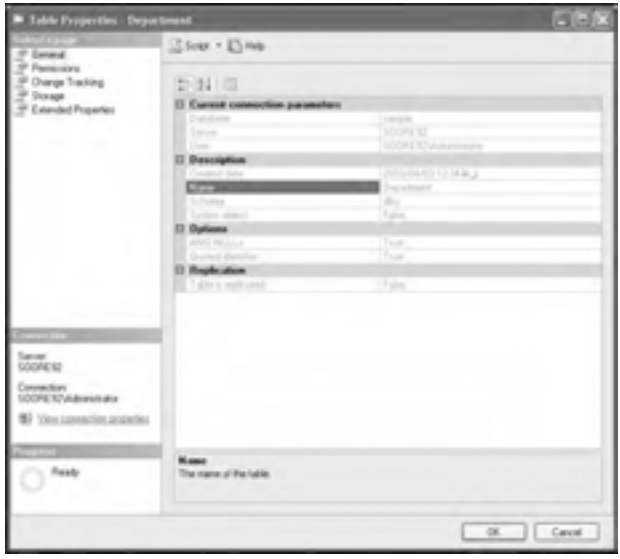
کادر علامت در ستون Allow Nulls را علامت بزنید تا مجوز درج مقادیر Null در آن ستون ارائه شود. به طور مشابه، اگر مقدار پیش فرضی وجود داشته باشد، در سطر Default Value or Binding از زبانه‌ی Column Properties وارد خواهد شد. (مقدار پیش فرض، مقداری است که در یک ستون جدول وارد می‌شود، هنگامی که مقداری برای آن وارد نشود).

شکل(۷-۸) ایجاد جدول بخش با استفاده از SQL Server Management Studio

ستون dept_no کلید اولیه‌ی جدول Department است. برای تعیین یک ستون به صورت کلید اولیه‌ی جدول، باید روی فیلد کلیک راست و Set Primary Key را انتخاب کنید. در پایان، ناحیه‌ی سمت راست را، که مربوط به جدول جدید است، ببندید. بعد از آن، سیستم، کادر محاوره‌ای Choose Name را نمایش خواهد داد که می‌توانید نام جدول را تایپ کنید.

برای مشاهده ی مشخصات جدول موجود، روی پوشه‌ی Tables دابل کلیک کنید. سپس، روی نام جدول کلیک راست نمائید و Properties را انتخاب کنید. شکل ۸-۸، کادر محاوره‌ای Table Properties برای جدول Department را نشان می‌دهد. برای تغییر نام

جدول، روی نام جدول کلیک راســت نمائید و Rename را انتخاب کنید. برای حذف، روی نام جدول کلیک راست نمائید و Delete را انتخاب کنید.



شکل(۸-۸) کادر محاوره ای Table Properties برای جدول Department

اکنون سه جدول دیگر پایگاه داده‌ی Sample را ایجاد کنید

جدول Employee

emp_no	emp_fname	emp_lname	dept_no

جدول Project

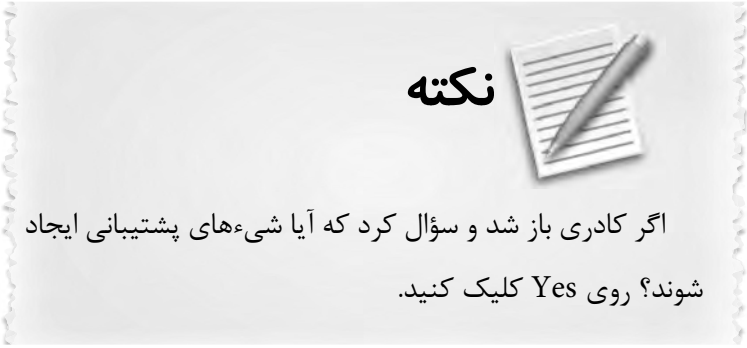
project_no	project_name	budget

جدول works_on

emp_no	project_no	jop	enter_date

بعـد از ایجاد چهار جدول پایگاه داده‌ی project ،department،employee،Sample و works_on) می‌توانیـد از ویژگـی دیگـر SQL Server Management Studio برای نمایـش نمـودار موجودیت رابطه ی ای آر (ER) مربوط به پایگاه داده‌ی Sample اسـتفاده کنیـد (فرآینـد تبدیل جدول‌های موجود پایـگاه داده به نمودار ای آر مربوطه را مهندسـی معکوس می‌نامند).

برای مشـاهده‌ی نمـودار ای آر پایـگاه داده‌ی Sample، روی زیرپوشـه‌ی Database Diagrams پایگاه داده‌ی Sample کلیک راست نمائید و New Database Diagrams را انتخاب کنید.



اولین (و تنها) گام، انتخاب جدول‌هایی است که به نمودار اضافه خواهند شد. بعد از اضافه کـردن تمام چهار جـدول از پایگاه داده‌ی Sample، ویـزارد، کار را کامل و نمودار را ایجاد

می‌کند (شکل ۹-۸).

نمودار شکل ۹-۸، نمودار نهایی پایگاه داده‌ی Sample نیست، زیرا چهار جدول را به همراه ستون‌ها و کلیدهای اولیه نشان می‌دهد ولی ارتباط بین جدول‌ها نمایش داده نمی‌شود. ارتباط بین دو جدول بر اساس کلید اولیه ی یک جدول و کلید خارجی در جدول دیگر است.

شکل (۹-۸) اولین نمودار پایگاه داده‌ی Sample

سه ارتباط بین جدول‌های پایگاه داده‌ی Sample وجود دارد که عبارت‌اند:

۱- جدول‌های employee و department دارای ارتباط ۱:N (یک به چند) هستند، زیرا به ازای هر مقدار در ستون کلید اولیه‌ی جدول(department dept_no)، یک یا چند مقدار مرتبط در ستون dept_no جدول employee وجود دارد.

۲- ارتباطی بین جدولهای employee و works_on وجود دارد، زیرا فقط آن مقادیری که در کلید اولیه‌ی جدول employee وجود دارند (emp_no) در ستون emp_no جدول works_on نیز وجود دارند.

۳- ارتباط سوم بین جدول‌های project و works_on وجود دارد، زیرا فقط مقادیری که

در کلید اولیه‌ی جدول(pr_no) project وجود دارند، در ستون pr_no جدول works_on ظاهر می‌شوند.

برای ایجاد هر کدام از این سه ارتباط، باید کلید خارجی را در نمودار، که به کلید اولیه‌ی جدول دیگر مرتبط است، به هم وصل کرد. انجام این کار مانند اکسس(Access) با درگ کردن کلید خارجی به سمت کلید اصلی یا با انجام مراحل زیر انجام می‌شود. برای مشاهده‌ی چگونگی انجام این کار، از جدول employee استفاده کنید و ستون dept_no آن را به منزله ی کلید خارجی جدول department تعریف کنید:

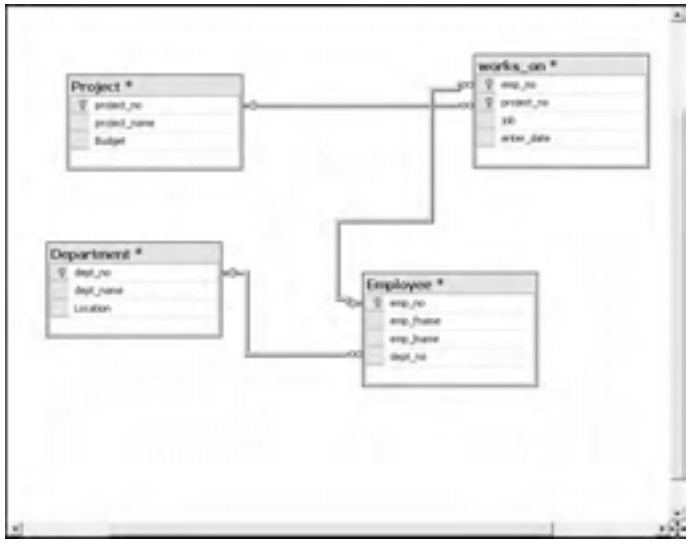
۱- روی diagram ایجاد شـــده کلیک کنید، روی شـــکل گرافیکی جدول employee در ناحیه‌ی detail کلیک راست نمائید وRelationship را انتخاب کنید. درکادر محاوره‌ای Foreign Key Relationship روی Add کلیک کنید.

۲- در کادر محـــاوره‌ای، ســـتون Tables and Columns Specification را باز و روی دکمه‌ی ... کلیک کنید.

۳- جدول را به همراه کلید اولیه‌ی مربوطه انتخاب کنید (جدول department).

۴- ستون dept_no از این جدول را به منزله ی کلید اولیه انتخاب کنید و ستون هم نام با آن را در جدول employee، به منزله ی کلید خارجی، تعیین و روی ok، هم چنین روی close کلیک کنید.

شـــکل ۱۰-۸، نمودار اصلاح شده را بعد از این که ســـه رابطه در پایگاه داده‌ی Sample ایجاد شدند، نمایش می‌دهد.



شکل (۱۰-۸) نمودار نهایی پایگاه داده‌ی نمونه (Sample)

۱- بانک اطلاعاتی بیمارستان (Hospital)را، که در واحدکارهای قبلی با آن آشنا شده‌اید، ایجاد کنید.

۲- جدول‌های آن را ایجاد کنید.

۳- ارتباط بین جدول‌ها را برقرار کنید.

۸-٤ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی آن را درک کند.

واژه	معنی	واژه	معنی
topic	موضوع - عنوان	recovery	بازیابی
describe	شرح دادن	options	گزینه‌ها -انتخاب‌ها
create	ایجاد کردن	modify	اصلاح کردن ویرایش
connect	اتصال	filegroup	گروه فایلی
instance	نمونه	property	مشخصه-ویژگی
expand	گسترش - بسط	column	ستون
accept	پذیرفتن	statement	عبارت
default	پیش‌فرض	full-text	متن کامل
value	مقدار	primary	اولیه - اصلی
otherwise	در غیر این	transaction	تراکنش
continue	صورت ادامه دادن	appropriate	مناسب
optional	انتخابی-اختیاری	collation	تطبیق دادن
owner	مالک	engine	موتور

This topic describes how to create a database by using SQL Server Management Studio.

To create a database

1-In **Object Explorer**, connect to an instance of the SQL Server Database Engine and then expand that instance.

2- Right-click **Databases**, and then click **New Database**.

3- In **New Database**, enter a database name.

4- To create the database by accepting all default values, click **OK**; otherwise, continue with the following optional steps.

5- To change the owner name, click (**…**) to select another owner.

6- To enable full-text search on the database, select the **Full-text indexing** check box.

7- To change the default values of the primary data and transaction log files, in the **Database files** grid, click the appropriate cell and enter the new value.

8- To change the collation of the database, select the **Options** page, and then select a collation from the list.

9- To change the recovery model, select the **Options** page and select a recovery model from the list.

10- To change database options, select the **Options** page, and then modify the database options.

11- To add a new filegroup, click the **Filegroups** page. Click **Add** and then enter the values for the filegroup.

12- To add an extended property to the database, select the **Extended Properties** page.

a- In the **Name** column, enter a name for the extended property.

b- In the **Value** column, enter the extended property text. For example, one or more statements that describe the database.

13- To create the database, click **OK**.

SQL SERVER یک سیستم مدیریت پایگاه داده ی توانمند است که امکان ایجاد و مدیریت پایگاه داده‌های رابطه‌ای را فراهم می‌کند و دارای ویژگی هایی است که به کارگیری آن ها را برای کاربران ساده کرده است.

این واحدکار، مهم ترین ابزار SQL Server را پوشش داد که عبارت است: SQL Server Management Studio. این ابزار هم برای کاربران عادی و هم مدیران پایگاه داده، خیلی مفید است. مهم ترین وظایف این ابزار عبارت اند از ایجاد پایگاه داده و جدول و ویرایش مشخصات آن ها.

SQL Server Management Studio شامل چندین جزء است که دو جزء آن عبارت اند:

- Registered Servers- امکان ثبت نمونه‌های SQL Server و اتصال به آن ها را فراهم می‌کند.
- Object Explorer- شامل یک نمای درختی از تمام شیء‌های پایگاه داده در یک سرور است.

چگونگی ایجاد ارتباط بین جدول‌ها و ترسیم نمودار بانک اطلاعاتی نیز از مواردی بود که در این فصل آموختید.

خودآزمایی

۱- با استفاده از SQL Server Management Studio ، پایگاه داده‌ای به نام Books (برای نگه داری مشـخصات کتاب‌های کتابخانه شخصی خودتان) ایجاد کنید. پایگاه داده را در فایلی به نام books_a در پوشه‌ی C:\temp ذخیره کنید.

۲- بـه کمـک SQL Server Management Studio، جـدول (هـای) پایگاه داده‌ی Books را به همراه تمام ستون‌هایشان ایجاد کنید.

۳- بررسی کنید که تفاوت‌های بین Access و SQL Server در چیست؟

۴- چگونه می‌توان میزان فضای ذخیره‌سازی برای یک پایگاه داده را در هنگام ایجاد آن تعیین کرد؟ (راهنمایی: در کادر New Database)

۵- آیـا می‌توان بدون اتصال به یک سـرور پایگاه داده، پایـگاه داده‌ی جدیدی را ایجاد کرد؟ چرا؟

مؤلفه‌های SQL

هدف های رفتاری

پس از پایان این فصل، هنرجو قادر خواهد بود:

- شیءهای اصلی SQL را شرح دهد؛
- انواع داده‌ها را بیان کند؛
- توابع Transact-SQL را توضیح دهد و به کار برد؛
- عملگرهای اسکالر را بیان کند؛
- مقادیر پوچ را شرح دهد؛
- واژه ها و متن انگلیسی مربوط به محتوای واحد کار را توضیح دهد.

۱-۹ شیءهای اصلی SQL

ویژگی های اصلی زبان Transact-SQL(Database Engine)(مانند سایر زبان‌های برنامه‌نویسی) عبارت اند از:

- ثابت‌ها
- حائل‌ها
- توضیحات
- شناسه‌ها
- کلیدواژه‌های رزرو شده

بخش‌های زیر، این ویژگی ها را شرح می‌دهند.

۱-۱-۹ ثابت‌ها

ثابت می‌تواند رشته‌ای، مبنای شانزده یا عددی باشد. ثابت رشته‌ای شامل یک یا چند کاراکتر است که درون دو تک کوتیشن یا زوج کوتیشن قرار می‌گیرد. ثابت‌های مبنای شانزده برای نمایش کاراکترهای غیر چاپی و سایر داده‌های دودویی مورد استفاده قرار می‌گیرند. مثال‌های ۱ و ۲ بعضی از ثابت‌های رشته‌ای و شانزده‌تایی معتبر و نامعتبر را نمایش می‌دهند.

مثال ۱) بعضی از ثابت‌های رشته‌ای و شانزده‌تایی معتبر عبارت اند از:

'Tehran'
“Hafez, Ave 710”
‘9876’
‘Apostrophe is displayed like this: can»t’
0x53514C0D

مثال ۲) موارد زیر، ثابت رشته‌ای نیستند:

'AB'C' (تعداد فرد علامت کوتیشن مارک)

"Tabriz' (نوع یکسانی از تک کوتیشن یا زوج کوتیشن باید استفاده شود.)

ثابت‌های عددی شامل مقادیر عددی صحیح، اعشاری ممیز ثابت و ممیز شناور با علامت و بدون علامت هستند.

مثال ۳) نمونه‌هایی از ثابت‌های عددی عبارت‌اند از:

130

-130.00

-0.357E5

22.3E-3

یک ثابت همیشه دارای یک نوع داده و طول است و هر دو بستگی به قالب ثابت دارد. به علاوه، هر ثابت عددی دارای یک عامل دقت و اندازه است.

۹-۱-۲ توضیحات

دو روش مختلف برای تعیین توضیح در عبارت Transact-SQL وجود دارد.یکی استفاده از زوج کاراکترهای /* و*/ که متن توضیح را در بر می‌گیرند. در این حالت، توضیح ممکن است چندین خط باشد. دیگر،وجود کاراکترهای - - (دو خط تیره) که مشخص می‌کند خط جاری، یک توضیح است.

۹-۱-۳ شناسه‌ها

در Transact-SQL، شناسه‌ها برای شناسایی شیءهای پایگاه‌داده مثل پایگاه‌داده، جدول‌ها و شاخص‌ها مورد استفاده قرار می‌گیرند. شناسه‌ها با رشته‌های کاراکتری نمایش داده می‌شوند که ممکن است حداکثر ۱۲۸کاراکتر باشند و می‌تواند حروف، اعداد یا کاراکترهای - ، @، #، $ را شامل شوند.

هر نام باید با حرف یا یکی از کاراکترهای - ، @، # شـروع شـود. کاراکتر # در ابتدای نام جدول یا یک روال، یک شیء موقتی را بیان می‌کند، در حالی که @ در ابتدای نام، یک متغیر را نشان می‌دهد.

٤-۱-۹ کلیدواژه‌های رزرو شده

هر زبان برنامه‌نویسـی دارای مجموعه‌ای از اسـامی به همراه معانی رزرو شـده است که باید در قالب تعریف شـده نوشـته شـوند و مورد اسـتفاده قرار گیرند. به این نوع اسـامی، کلیدواژه‌های رزرو شـده می‌گویند. Transact-SQL از اسـامی متنوعی اسـتفاده می‌کند (مانند سـایر زبان‌های برنامه‌نویسـی) که نمی‌توانند به عنوان نام شـیء‌ها به کار روند، مگر این که شیء‌ها به صورت شناسه‌های جداکننده تعیین شده باشند.

۹-۲ انواع داده‌ها

تمام مقادیر داده‌ای یک سـتون باید از یک نوع باشـند (تنها اسـتثناء برای مقادیر نوع داده‌ی SQL_VARIANT اسـت). Transact-SQL از انـواع داده‌های مختلف اسـتفاده می‌کند که می‌توانند به صورت زیر دسته بندی شوند:

- عددی
- کاراکتری
- زمانی (تاریخ و ساعت)
- متفرقه

DECIMAL با نوع ذخیره‌ی VARDECIMAL

بخش‌های زیر، این دسته بندی‌ها را شرح می‌دهند:

۹-۲-۱ انواع داده‌های عددی

انواع داده‌های عددی، برای نمایش اعداد مورد استفاده قرار می‌گیرند. جدول زیر، لیست تمام انواع داده‌های عددی را نشان می‌دهد:

انواع داده‌ها	شرح
INTEGER	مقادیر عددی صحیح را ارائه می‌کند که در ۴ بایت ذخیره می‌شوند. محدوده‌ی مقادیر از ۲,۱۴۷,۴۸۳,۶۴۸- تا ۲,۱۴۷,۴۸۳,۶۴۷ است. int مخفف آن است.
SMALLINT	مقادیر عددی صحیح که می‌توانند در ۲ بایت ذخیره شوند. محدوده‌ی مقادیر از ۳۲۷۶۸- تا ۳۲۷۶۷ است.
TINYINT	مقادیر عددی صحیح غیرمنفی که در یک بایت ذخیره می‌شوند. محدوده‌ی مقادیر از ۰ تا ۲۵۵ است.
BIGINT	مقادیر عددی صحیح که می‌توانند در ۸ بایت ذخیره شوند. محدوده‌ی مقادیر از 2^{63}- تا $2^{63}-1$ است.
DECIMAL(P,[S])	مقادیر اعشاری ممیز ثابت را ارائه می‌کند. آرگومان P تعیین می‌کند که عدد چند رقمی است (با در نظر گرفتن رقم‌های اعشار S از سمت راست). فضای مورد نیاز برای ذخیره‌ی مقادیر DECIMAL بستگی به مقدار P دارد که می‌تواند از ۵ تا ۱۷ بایت باشد. DES مخفف آن است.
NUMERIC(P,[S])	معادل DECIMAL
REAL	برای مقادیر اعشاری ممیز شناور مورد استفاده قرار می‌گیرد. محدوده‌ی مقادیر مثبت تقریباً از ۲,۲E-۳۰۸ تا ۱,۷۹E۳۰۸ و مقادیر منفی از ۱,۱E-۳۸- تا ۱,۱E-۳۸+ است.
FLOAT[(P)]	مانند REAL مقادیر اعشاری ممیز شناور را ارائه می‌کند. P دقت را تعیین می‌کند. اگر p<۲۵ باشد، ۴ بایت و P>=۲۵ یعنی دقت مضاعف باشد، ۸ بایت را اشغال می‌کند.
MONEY	برای نمایش مقادیر پولی مورد استفاده قرار می‌گیرد. مقادیر MONEY به مقادیر DECIMAL ۸ بایتی مرتبط هستند و رقم‌های بعد از ممیز به ۴ رقم گرد می‌شوند.
SMALL MONEY	مشابه نوع داده‌ی MONEY است ولی در ۴ بایت ذخیره می‌شود.

۹-۲-۲ انواع داده‌های کاراکتری

دو شـکل کلی از انـواع داده‌های کاراکتری وجـود دارد. آن ها می‌توانند رشـته‌هایی از کاراکترهای تک بایتی یا رشته‌هایی از کاراکترهای Unicode باشند.(Unicode از چندین بایت برای تعیین کاراکتر اسـتفاده می‌کند). رشـته‌ها می‌توانند طول ثابت یا متغیر داشـته باشند. انواع داده‌های کاراکتری زیر مورد استفاده قرار می‌گیرند:

انواع داده‌ها	شرح
CHAR[(n)]	یک رشـته با طول مثبت از کاراکترهای تک بایتی را ارائه می‌دهد که n تعداد کاراکترهای رشـته اسـت. حداکثر مقدار n می‌تواند ۸۰۰۰ باشد. CHARACTER(n) معادل دیگری است. اگر n مقداردهی نشود، طول رشته، ۱ فرض می‌شود.
VARCHAR[(n)]	یک رشـته‌ی طول متغیر از کاراکترهای تک رشته‌ای را ارائه می‌کند. (۸۰۰۰=>n>۰)
NCHAR[(n)]	رشـته‌های با طول ثابت از کاراکترهای Unicode را ذخیره می‌کند. تفاوت اصلی بین CHAR و NCHAR در این است که هر کاراکتری از نوع داده‌ی NCHAR در ۲ بایت ذخیره می‌شود ولی در CHAR یک بایت است. بنابراین، حداکثر تعداد کاراکترها در یک ستون از نوع داده‌ی NCHAR ، ۴۰۰۰ است.
NVARCHAR[(n)]	کاراکترهـای Unicode با طـول متغیر را ذخیـره می‌کند. تفاوت اصلـی بیـن VARCHAR و NVARCHAR در این اسـت که NVARCHAR ۲ بایت و VARCHAR یک بایت فضا اشـغال می‌کند. حداکثر تعداد کاراکترها در ستون NVARCHAR، ۴۰۰۰ است.

نکته

نوع داده‌ی VARCHAR با نوع داده‌ی CHAR یک سان است، با این تفاوت که اگر محتوای یک رشته‌ی (CHAR)n کوچک تر از کاراکترهای n باشد، بقیه‌ی رشته با فضاهای خالی پر می‌شود.

۳-۲-۹ انواع داده‌های زمانی

Transact-SQL از نوع داده‌های زمانی زیر پشتیبانی می‌کند:

- DATETIME
- SMALLDATETIME
- DATE
- TIME
- DATETIME۲
- DATETIMEOFFSET

انواع داده‌های DATETIME و SMALLDATETIME مقادیر تاریخ و زمان را تعیین می‌کنند و مقادیر هر کدام از این نوع داده‌ها به ترتیب در ۴ و ۲ بایت ذخیره می‌شوند. مقادیر این دو نوع داده به صورت دو مقدار عددی مجزا ذخیره می‌شوند. مقدار داده‌ی DATETIME در محدوده‌ی 01 /1753/01 تا 9999/31/12 است ولی بازه‌ی SMALLDATETIME در محدوده‌ی 01 /1900/01 تا 06 /2079/06 است. مؤلفه‌ی زمان برحسب ثانیه و به صورت فیلد ۴ بایتی ذخیره می‌شود.

استفاده از DATETIME و SMALLDATETIME فقط برای ذخیره‌ی بخش DATE و TIME خیلی ساده نیست. به همین دلیل، SQL Server 2008 انواع داده‌های جدید DATE یا TIME از DATETIME را به ترتیب ذخیره می‌کند. نوع داده‌ها می‌توانند برای طول‌های متغیر تعریف شوند که بستگی به نیازها دارد (اندازه‌ی ذخیره‌سازی بین ۶ تا ۸ است). دقت بخش زمان، ۱۰۰نانو ثانیه است و در ۳ تا ۵ بایت ذخیره می‌شود. نوع داده‌ی DATE در ۳ بایت ذخیره می‌شود و دارای محدوده‌ی 01 /01/ 0001 تا 9999/31/12 است. نوع داده‌ی DATETIME2 نیز نوع داده‌ی جدیدی است که داده‌های تاریخ و زمان با دقت بالا را ذخیره می‌کند.

تمام انواع داده‌های زمانی که تا این جا شرح دادیم از ناحیه‌ی زمانی (Time Zone) پشتیبانی نمی‌کنند. نوع داده‌ی جدیدی به نام DATETIMEOFFEST دارای بخش افست ناحیه‌ی زمانی است. به همین دلیل، در ۶ تا ۸ بایت ذخیره می‌شود.

مقادیر تاریخی در Transact-SQL به طور پیش فرض به صورت رشته‌ای در قالبی شبیه 'mmm dd yyyy' (مثل 'jan 10 1993') تعیین می‌شوند. به طور مشابه، مقدار زمان نیز در قالب 'hh:mm' مشخص می‌شود و Database Engine از زمان ۲۴ ساعتی استفاده می‌کند (برای مثال، ۲۳:۲۴).مثال‌های ۴ و ۵ روش‌های مختلف نوشتن مقادیر تاریخ و زمان با استفاده از قالب‌های متفاوت را نشان می‌دهند.

مثال ۴) تاریخ به صورت‌های زیر می‌تواند مورد استفاده قرار گیرد:

'1959/5/28'
‘MAY 28, 1959’
‘1959 MAY 28’

مثال ۵) عبارات زمانی زیر می‌توانند به کار گرفته شوند:

' 8:45 AM '
' 4 PM '

۹-۳ توابع Transact-SQL

توابع SQL می‌توانند توابع تجمعی یا اسکالر باشند. بخش‌های زیر، این توابع را شرح می‌دهند.

۹-۳-۱ توابع تجمعی

این نوع توابع به گروهی از مقادیر داده‌ای یک ستون اعمال می‌شوند. این توابع همیشه یک مقدار برمی‌گردانند. Transact-SQL از چندین گروه از توابع تجمعی، پشتیبانی می‌کند:

- توابع تجمعی عادی
- توابع تجمعی آماری
- توابع تجمعی تعریف شده به وسیله‌ی کاربر
- توابع تجمعی تحلیلی

توابع تجمعی نوع اول در این جا شرح داده می‌شوند:

- **AVG**- میانگین مقادیر داده‌ای درون ستون را محاسبه می‌کند. ستون باید شامل مقادیر عددی باشد.
- **MAX و MIX**- بیشینه و کمینه‌ی مقادیر داده‌ای ستون را محاسبه می‌کند. ستون می‌تواند شامل مقادیر عددی، رشته‌ای و تاریخ و زمان باشد.

- **SUM**- مجموع تمام مقادیر یک ستون را محاسبه می‌کند. ستون باید شامل مقادیر عددی باشد.
- **COUNT**- تعداد مقادیر داده‌ای (غیر پوچ) یک ستون را شمارش می‌کند.
- **COUNT_BIG**- در مقایسه با COUNT تنها تفاوت آن این است که یک مقدار از نوع داده‌ی BIGINT را برمی‌گرداند. از این توابع همراه با دستور SELECT می‌توان استفاده کرد.

۹-۳-۲ توابع اسکالر

علاوه بر توابع تجمعی، Transact-SQL چندین تابع اسکالر را ارائه می‌کند که در ایجاد عبارات اسکالر مورد استفاده قرار می‌گیرند(یک تابع اسکالر روی یک مقدار یا لیستی از مقادیر، عمل می‌کند در حالی که توابع تجمعی روی داده‌های چندین سطر اعمال می‌شوند). توابع اسکالر می‌توانند به گروه‌های زیر تقسیم شوند:

- توابع عددی
- توابع تاریخی
- توابع رشته‌ای
- توابع سیستمی
- توابع دادگان (Metadata)

شرح این توابع را می‌توانید در پیوست ب مشاهده کنید.

۹-۳-۳ عملگرهای اسکالر

عملگرهای اسکالر برای انجام عملیات با مقادیر اسکالر به کار می‌روند. Transact-SQL از عملگرهای عددی و منطقی پشتیبانی می‌کند.

عملگرهای ریاضی به صورت یک تایی و دوتایی وجود دارند. عملگرهای یک تایی عبارتند از + و- (به صورت علامت). عملگرهای دوتایی عبارت‌اند از + ،- ،*، / و ٪. (۴ عملگر اولی همان معنی ریاضی خودشان را دارند در حالی که ٪ عملگر باقی‌مانده است).

عملگرهای منطقی دارای دو نماد متفاوت هستند و بستگی به عملوندهای آن‌ها دارد که رشته‌های بیتی یا سایر انواع داده‌ها هستند. عملگر NOT، AND و OR به تمام انواع داده‌ها به جز BIT اعمال می‌شوند (آن‌ها در ادامه‌ی کتاب شرح داده می‌شوند).

عملگرهای بیتی برای کارکردن با رشته‌های بیتی به صورت زیر هستند. مثال ۸ چگونگی

به کارگیری آن ها را نشان می‌دهد:

~ مکمل (مثل NOT):صفر را به ۱ و ۱ را به صفر تبدیل می‌کند.

& به هم پیوستگی رشته‌های بیتی (مثل AND): فقط در صورتی ۱ است که هر دو بیت ۱ باشند.

¦ انفصال رشته‌های بیتی (مثل OR): اگر یکی از بیت‌ها ۱ باشند، حاصل ۱ است.

^ انفصال انحصاری (مثل XOR): فقط در صورتی ۱ است که یکی از بیت‌ها ۱ باشد.

جدول زیر این عملگرها را نشان می‌دهد:

~p	p ^ q	p | q	p & q	q	p
0	0	1	1	1	1
0	1	1	0	0	1
1	1	1	0	1	0
1	0	0	0	0	0

مثال ۸)

```
~ (1001001) = (0110110)
(11001001) ¦ (10101101) = (11101101)
(11001001) & (10101101) = (10001001)
(11001001) ^ (10101101) = (01100100)
```

عملگر+ نیز می‌تواند برای به هم پیوستگی دو رشته‌ی کاراکتری یا بیتی مورد استفاده قرار گیرد.

متغیرهای عمومی

این نوع متغیرها، متغیرهای سیستمی خاصی هستند که می‌توانند به صورت ثابت‌های اسکالر استفاده شوند. Transact-SQL از چندین متغیر عمومی که قبل از نامشان @@ ظاهر می‌شود، پشتیبانی می‌کند. جدول صفحه‌ی بعد، چندتا از این متغیرها را شرح می‌دهد.

متغیر	شرح
@@CONNECTIONS	تعداد ورودهای پذیرفته شده از زمان شروع به کار سیستم را برمی‌گرداند.
@@CPU_BUSY	کل زمان پردازنده (بر حسب میلی ثانیه) از زمان شروع به کار سیستم را برمی‌گرداند.
@@ERROR	اطلاعاتی درباره‌ی مقدار برگشتی آخرین عبارت اجرا شده‌ی SQL را برمی‌گرداند.
@@IDENTITY	آخرین مقدار درج شده در ستونی با مشخصه‌ی IDENTITY را برمی‌گرداند.
@@LANGID	شناسه‌ی زبانی را که در حال حاضر به وسیله‌ی سیستم پایگاه داده به کار می‌رود برمی‌گرداند.
@@LANGUAGE	نام زبانی را که در حال حاضر به وسیله‌ی سیستم پایگاه داده به کار می‌رود برمی‌گرداند.
@@MAX_ CONNECTIONS	حداکثر تعداد اتصالات واقعی به سیستم را برمی‌گرداند.
@@PROCID	شناسه‌ی روال در حال اجرا را برمی‌گرداند.
@@ROWCOUNT	تعداد سطرهایی را که به وسیله‌ی آخرین دستور اجرا شده‌ی SQL تحت تأثیر قرار گرفته است برمی‌گرداند.
@@SERVERNAME	اطلاعات مربوط به سرور پایگاه داده‌ی محلی را بازیابی می‌کند.
@@SPID	شناسه‌ی فرآیند سِرور را برمی‌گرداند.
@@VERSION	نسخه‌ی جاری نرم افزار سیستم پایگاه داده را برمی‌گرداند.

٤-۹ مقادیر پوچ

مقدار NULL، مقدار خاصی است که ممکن است برای یک ستون تعیین شود. این مقدار معمولاً هنگامی که اطلاعات ستون ناشناخته یا غیرکاربردی باشند، مورد استفاده قرار می‌گیرد. برای مثال، در حالتی که شماره ی تلفن منزل برای کارمند شرکت مشخص نباشد، توصیه می‌شود که مقدار NULL برای ستون home_telephone تعیین شود.

اگر هر عملوندی دارای مقدار NULL باشد، حاصل عبارت ریاضی نیز NULL خواهد

بود. بنابراین در یک عبارت ریاضی یک تایی، در صورتی که A برابر NULL باشـد، هم A+ و هـم A- مقدار NULL را برمی‌گردانند. در عبارت‌های دوتایی، اگر یک (یا هر دو) عملوند A یا B دارای مقدار NULL باشند، A+B، A-B ، A*B، A/B، AbB نیز مقدار NULL را بر می‌گردانند(عملوندهای A و B باید عبارات عددی باشند).

اگر یک عبارت شـامل عملیات رابطه‌ای باشـد و یک (یا هر دو) عملوند مقدار NULL داشـته باشـند، نتیجه‌ی این عملیات نیز NULL خواهد شـد. بنابراین، هر کدام از عبارات A<B ، A> <B، A=B و A>B نیز مقدار NULL را برمی‌گردانند.

۹-۵ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شـده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی آن را درک کند.

معنی	واژه	معنی	واژه
متغیر	variable	انواع داده‌ها	data types
تابع	function	شیء	object
برگرداندن - بازگشت	return	شامل – در برگرفتن	contain
خاص	specific	وابسـته کـردن - وابسته	associate
واگـذار کـردن – تعییـن کردن	assign	تعریف کردن	define
صفت – مشخصه - ویژگی	attribute	نوع	kind
طول	length	کاراکتری	character
دقت		عدد صحیح	integer
مقیاس	scale	دودویی	binary
عدد - عددی	numeric	ستون	column
پارامتر		دیدگاه	view
		روال ذخیره شده	stored procedure

Data Types (Database Engine)

Objects that contain data have an associated data type that defines the kind of data; for example, character, integer, or binary, the object can contain. The following objects have data types:

- Columns in tables and views.
- Parameters in stored procedures.
- Variables.
- Transact-SQL functions that return one or more data values of a specific data type.
- Stored procedures that have a return code, which always has an **integer** data type.

Assigning a data type to an object defines four attributes of the object:

- The kind of data contained by the object.
- The length or size of the stored value.
- The precision of the number (numeric data types only).
- The scale of the number (numeric data types only).

ویژگی‌های اصلی Transact-SQL شـامل انواع داده‌ها، مسـتندات و توابع است. انواع داده‌هـا با انواع داده‌های ANSI SQL۹۲ مطابقت دارنـد. Transact-SQL از چندین نوع تابع سیستمی مفید پشتیبانی می‌کند.

Transact-SQL نیز مانند زبان‌های برنامه‌نویسـی از انواع داده‌ها و عملگرها پشتیبانی می‌کنـد. آشـنایی با انواع داده‌ها و عملگرهای هر زبانی، مقدمات شـروع بـه کار با آن زبان محسوب می‌شود.

انواع داده‌هایی که زبان Transact-SQL پشتیبانی می‌کند، عبارت‌اند از:

عددی

کاراکتری

زمانی (تاریخ و ساعت)

متفرقه

DECIMAL با نوع ذخیره‌ی VARDECIMAL

Transact-SQL از انواع عملگرهای محاسـباتی، منطقی، بیتی و رشـته‌ای پشـتیبانی می‌کند.

Transact-SQL دارای توابع متعددی است که برای انجام عملیات روی داده‌ها می‌توان آن‌ها را به کار برد که با برخی از آن‌ها در این فصل آشنا شدید.

مقدار پوچ، مقداری است که در برخی از فیلدهایی که مقدار ندارند، درج می‌شود که هنگام ایجاد جدول می‌توان برای هر فیلد تعیین کرد که آیا مقدار پوچ را بپذیرد یا نه؟

خودآزمایی

۱- تفاوت بین انواع داده‌های عددی SMALLINT، INT و TINYINT چیست؟

۲- تفاوت بین انواع داده‌های CHAR و VARCHAR چیست؟ موارد استفاده هر کدام را بیان کنید.

۳- چگونه می‌توان قالب ستونی را از نوع داده‌ی DATE تعیین کرد تا مقادیر وارد شده به شکل 'yyyy/mm/dd' باشند؟

۴- با استفاده از عملگرهای بیتی & ، | ، ^ عملیات زیر را انجام دهید:

(۰۱۰۱۰۱۱۱) & (۱۱۱۰۰۱۰۱)

(۱۰۰۱۱۰۱۱) | (۱۱۰۰۱۰۰۱)

(۱۰۱۱۰۱۱۱) ^ (۱۰۱۱۰۰۰۱)

۵- نتیجه‌ی عبارت‌های زیر چیست؟ (A یک عبارت عددی و B منطقی است.)

A+NULL

NULL = NULL

B OR NULL

B AND NULL

۶- چه زمانی می‌توان از تک کوتیشن و زوج کوتیشن برای تعریف رشته و ثابت‌های موقتی استفاده کرد؟

زبان تعریف داده‌ها

هدف های رفتاری

پس از پایان این فصل، هنرجو قادر خواهد بود:

- شیءهای پایگاه داده را ایجاد کند؛
- شیءهای پایگاه داده را ویرایش کند؛
- شیءهای پایگاه داده را حذف کند.

این واحدکار تمام عبارات SQL مربوط به زبان تعریف داده‌ها (DDL) را شـرح می‌دهد. عبارات DDL به سـه گروه تقسـیم می‌شوند که در ادامه شـرح داده می‌شوند. اولین گروه شـامل عباراتی اسـت که شـیء‌ها را ایجاد می‌کند، گروه دوم شـامل عباراتی برای ویرایش ساختار شیء‌هاست و گروه سوم، عباراتی است که شیء‌ها را حذف می‌کنند.

۱-۱۰ ایجاد شیء‌های پایگاه داده

همان طور که می‌دانید،پایگاه داده شـامل چندین شـیء مختلف اسـت. تمام شیء‌های پایگاه داده می‌توانند فیزیکی یا منطقی باشند. شیء‌های فیزیکی به سازماندهی داده‌ها روی رسـانه‌ی فیزیکی (دیسک) مرتبط هستند. شـیء‌های فیزیکی موتور پایگاه داده عبارت‌اند از فایل‌هـا و گروه‌های فایلی. شـیء‌های منطقی، یک دیدگاه کاربـری از پایگاه داده را ارائه می‌کنند. پایگاه داده‌ها، جدول‌ها، سـتون‌ها و دیدگاه‌هـا (جدول‌های مجازی)، مثال‌هایی از شیء‌های منطقی هستند.

اولین شـیء پایگاه داده،که باید ایجاد شود، خود پایگاه داده است. Database Engine هـم پایگاه داده‌های سیسـتمی و هم کاربری را مدیریت می‌کنـد. یک کاربر معتبر می‌تواند پایگاه داده‌های کاربری را ایجاد کند، در حالی که پایگاه داده‌های سیسـتمی در طول نصب سیستم پایگاه داده تولید می‌شوند.

پایگاه داده‌های سیستمی عبارت‌اند از:

- master
- tempdb
- model
- msdb
- resource

این فصل، ایجاد، ویرایش و حذف پایگاه داده‌های کاربری را شرح می‌دهد.

۱-۱-۱۰ ایجاد یک پایگاه داده

دو روش اصلی برای ایجاد پایگاه داده وجود دارد. اولین روش، استفاده از Object Explorer در SQL Server Management Studio است(فصل هفتم). دومین روش، استفاده از عبارت CREATE DATABASE در SQL است. این عبارت دارای شکل کلی زیر است که جزئیات آن در ادامه شرح داده می‌شود:

```
CREATE DATABASE db_name
[ON [PRIMARY] { file_spec1} ,...]
[LOG ON {file_spec2} ,...]
[COLLATE collation_name]
[FOR {ATTACH | ATTACH_REBUILD_LOG } ]
```

db_name نام پایگاه داده است. حداکثر اندازه‌ی نام پایگاه داده، ۱۲۸کاراکتر است. حداکثر پایگاه داده‌هایی که به وسیله‌ی یک سیستم مدیریت می‌شوند، ۳۲۷۶۷ واحد است. نام پایگاه داده در یک سرور باید منحصر به فرد باشد و از قاعده ی نام‌گذاری شناسه‌ها پیروی کند (ترکیبی از حروف A تا Z و کاراکترهای عددی).

پایگاه داده‌ها در فایل‌ها ذخیره می‌شوند. این فایل‌ها می‌توانند به طور واضح به وسیله‌ی مدیر سیستم تعیین شوند یا به وسیله‌ی سیستم ارائه شوند. اگر گزینه‌ی ON در دستور CREATE DATABASE وجود داشته باشد، تمام فایل‌ها داده‌ی پایگاه داده به طور مشخص، تعیین می‌شوند.

file_spec۱ مشخصه‌ی فایل را ارائه می‌کند که شامل گزینه‌هایی مانند نام منطقی فایل، نام فیزیکی و اندازه ی آن است. گزینه‌ی PRIMARY اولین (و مهم ترین) فایلی را که شامل جدول‌های سیستم و سایر اطلاعات درونی مربوط به پایگاه داده است تعیین می‌کند. اگر

گزینه‌ی PRIMARY نوشته نشود، اولین فایل لیست شده در مشخصات به صورت فایل اصلی مورد استفاده قرار می‌گیرد.

یک حساب ورود به Database Engine را که برای ایجاد پایگاه داده به کار می‌رود مالک پایگاه داده می‌نامند. هر پایگاه داده‌ای می‌تواند یک مالک داشته باشد که همیشه مرتبط به یک نام حساب کاربری است.

حساب کاربری که مالک پایگاه داده است دارای نام خاص dbo است. این نام همیشه در ارتباط با پایگاه داده‌ای که مالک آن است، به کار می‌رود. dbo از گزینه‌ی LOG ON برای تعریف یک یا چند فایل به عنوان محل فیزیکی ثبت تراکنش[۱] پایگاه داده استفاده می‌کند. اگر گزینه‌ی LOG ON تعیین نشود، ثبت تراکنش پایگاه داده باز هم ایجاد خواهد شد، زیرا هر پایگاه داده‌ای باید دارای حداقل یک فایل ثبت تراکنش باشد.

با گزینه‌ی COLLATE، می‌توان تلفیق پیش فرض پایگاه داده را تعیین کرد. اگر این گزینه مشخص نشود، پایگاه داده تلفیق پیش فرض را از پایگاه داده‌ی model تعیین می‌کند که همان تلفیق پیش فرض سیستم پایگاه داده است.

گزینه‌ی FOR ATTACH تعیین می‌کند که پایگاه داده با ضمیمه کردن مجموعه‌ای از فایل‌های سیستم عامل، ایجاد شده است. اگر این گزینه مورد استفاده قرار گیرد، باید ابتدا فایل اصلی را تعیین کنید.

در طول ایجاد یک پایگاه داده‌ی جدید، Database Engine از پایگاه داده‌ی model به صورت الگو استفاده می‌کند. مشخصات پایگاه داده‌ی model بر اساس نظر مدیر سیستم می‌تواند تغییر یابد.

```
USE master;
CREATE DATABASE sample;
```

۱- مجموعه‌ای از عملیاتی است که به ترتیب اجرا می‌شوند و داده‌های بانک اطلاعاتی را تغییر می‌دهند. در ادامه راجع به مفهوم تراکنش بیش‌تر توضیح خواهیم داد.

ایـــن مثال، پایگاه داده‌ای به نام sample را ایجاد می‌کند. این کوتاه ترین شـــکل ممکن از دســـتور CREATE DATABASE است، زیرا تمام گزینه‌های آن مقادیر پیش فرض را خواهند داشـــت. سیستم به طور پیش فرض، دو فایل ایجاد می‌کند. نام منطقی فایل داده‌ها sample اســـت و اندازه‌ی آن ۲ مگابایت می‌باشـــد. به طور مشـــابه نام منطقی فایل ســـابقه تراکنش، sample_Log و اندازه‌ی آن یک مگابایت اســـت(هر دو اندازه مربوط به مشخصات پایگاه داده‌ی جدیدندو بستگی به تنظیمات پایگاه داده‌ی model دارند).

با استفاده از دستور CREATE DATABASE می‌توان یک بانک اطلاعاتی و فایل‌هایی را بـــرای ذخیره‌ی بانک اطلاعاتی ایجاد کرد. SQL SERVER این دســـتور را در دو مرحله پیاده‌سازی می‌کند:

۱- SQL SERVER با استفاده از بانک اطلاعاتی model، بانک اطلاعاتی جدید را ایجاد می‌کند.

۲- SQL SERVER بقیـــه ی بانک اطلاعاتی را با صفات خالی پر می‌کند. به جز صفاتی که شامل اطلاعاتی راجع به چگونگی بهره‌برداری از فضای بانک اطلاعاتی هستند.

بنابراین، تمام شیءهایی که در پایگاه داده‌ی model وجود دارند، در پایگاه داده‌ی جدید کپی می‌شـــوند. هر جزئی از پایگاه داده را می‌توان به model اضافه کرد تا در همه ی پایگاه داده‌هـــای جدید قرار گیرند. توجه کنید که پایگاه داده‌هـــای جدید، تنظیمات پایگاه داده‌ی model را به ارث می‌برند.

برای ذخیره‌ی پایگاه داده سه نوع فایل، مورد استفاده قرار می گیرند:

فایل داده ی اصلی که شـــامل اطلاعات راه‌اندازی بانک اطلاعاتی اســـت. فایل اصلی برای ذخیره‌ی داده‌ها نیز به کار می‌رود. هر بانک اطلاعاتی دارای یک فایل اصلی است که معمولاً با پسوند .mdf ذخیره می‌شود.

فایل‌های داده ی کمکی که شامل داده‌هایی هستند که در فایل داده ی اصلی نمی‌گنجند. اگر فایل اصلی به اندازه ی کافی بزرگ باشد، به نحوی که تمام داده‌ها را در برگیرد، نیاز به فایل‌های کمکی نخواهد بود.

فایل‌های ثبت تراکنش، اطلاعاتی را برای ترمیم بانک اطلاعاتی در اثر خرابی، ذخیره می‌کنند. برای هر بانک اطلاعاتی، حداقل یک فایل تراکنش باید وجود داشته باشد، گرچه ممکن است چند فایل تراکنش وجود داشته باشد. حداقل اندازه فایل ثبت تراکنش، ۵۱۲ کیلوبایت است.

مثال ۱، یک پایگاه داده را به همراه مشخصات آن ایجاد می‌کند.

مثال ۱)

```
USE master;
CREATE DATABASE projects
ON (NAME=projects_dat, FILENAME = 'C:\projects.mdf', SIZE = 10,
MAXSIZE = 100, FILEGROWTH = 5)
LOG ON (NAME=projects_log, FILENAME = 'C:\projects.ldf', SIZE = 40,
MAXSIZE = 100, FILEGROWTH = 10);
```

مثال ۱، پایگاه داده‌ای به نام projects را ایجاد می‌کند. به دلیل این که گزینه‌ی PRIMARY حذف شده است، اولین فایل به عنوان فایل اصلی فرض می‌شود. این فایل با نام منطقی .Projects mdf ذخیره می‌شود. اندازه‌ی این فایل، ۱۰ مگابایت است. بخش‌های اضافه بر ۵ مگابایت به وسیله‌ی سیستم برای مواقع ضروری، تخصیص یافته‌اند. اگر گزینه‌ی MAXSIZE تعیین نشده باشد یا با UNLIMITED مقداردهی شود، فایل تا زمانی که دیسک پر نشده است، افزایش می‌یابد.

هم چنین یک فایل ثبت تراکنش به همراه نام منطقی Projects_Log و نام فیزیکی Projects.ldf وجود دارد. تمام گزینه‌های فایل ثبت با فایل داده، یک سان و هم معنی هستند. با استفاده از زبان Transact-SQL، می‌توان از دستور USE برای تغییر محتوای پایگاه داده

به پایگاه داده‌ی مشخص، استفاده کرد(روش جای گزین، انتخاب نام پایگاه داده در منوی بازشوی Database از نوار ابزار SQL Server Management Studio است).

این دستورات Transact-SQL در کجا باید تایپ و اجرا شوند؟

پاسخ: در محیط Query Editor که در ادامه شرح داده می‌شود.

۲-۱-۱۰ ثبت عملیات با استفاده از SQL Server Management Studio

SQL Server Management Studio یک محیط کامل را برای نوشتن تمام انواع دستورات SQL را ارائه می‌کند. می‌توان پرس‌وجوها را ایجاد،ذخیره،بازیابی و ویرایش کرد. SQL Server Management Studio به شما امکان می‌دهد که روی پرس‌وجوها بدون اتصال به سرور خاصی، کار کنید. این ابزار امکان ایجاد پرس‌وجوها با پروژه‌های مختلف را نیز فراهم می‌کند.

Query Editor

برای اجرای Query Editor،روی دکمه‌ی New Query در نوارابزار SQL Server Management Studio کلیک کنید. به طور پیش فرض، یک پرس‌وجوی Database Engine جدیدی ارائه می‌شود.

بعد از این که Query Editor را باز کردید، نوار وضعیت در پایین بیان می‌کند که آیا پرس‌وجو متصل است یا نه؟ اگر به صورت خودکار به سرور متصل نشوید، کادر محاوره‌ای Connect to SQL Server ظاهر می‌شود که می‌توانید نام سرور پایگاه داده‌ی مورد نظر را تایپ و حالت شناسایی را انتخاب کنید.

نکته

ویرایش غیرمتصل، انعطاف پذیری بیش تری دارد. می‌توانید پرس‌وجوها را بدون این که مجبور به انتخاب سروری باشید، ویرایش کنید. برای انجام این کار، در کادر محاوره‌ای Connect to SQL Server روی دکمه ی Cancel کلیک کنید.

Query Editor می‌تواند به وسیله‌ی کاربران عادی برای انجام عملیات زیر، مورد استفاده قرار گیرد:

- تولید و اجرای عبارات Transact-SQL
- ذخیره‌ی عبارات Transact-SQL در یک فایل
- تولید و تجزیه و تحلیل اهداف اجرایی برای پرس‌وجوهای تولید شده
- تشریح گرافیکی هدف از اجرای یک پرس‌وجوی انتخاب شده

Query Editor شامل یک ویراستار متن داخلی و مجموعه‌ای از دکمه‌ها در نوارابزار است. پنجره‌ی اصلی به دو ناحیه‌ی بالایی (ناحیه‌ی پرس‌وجو) و ناحیه‌ی پایینی (نتایج) تقسیم می‌شود. کاربران عبارات Transact-SQL را در ناحیه‌ی پرس‌وجو وارد می‌کنند و بعد سیستم، پرس‌وجو را پردازش می‌کند و خروجی در ناحیه‌ی نتایج نمایش داده می‌شود.

مثال نشان داده شده در شکل ۱-۱۰ یک پرس‌وجوی وارد شده در Query Editor و خروجی برگردانده شده را شرح می‌دهد. اولین عبارت در ناحیه‌ی پرس‌وجو (USE)، پایگاه داده‌ی Sample را به عنوان پایگاه داده‌ی جاری، تعیین می‌کند. دومین عبارت (Select)، سطرهای جدول works_on را بر می‌گرداند. کلیک روی دکمه‌ی Query در نوارابزار

Query Editor's و سپس انتخاب Excute یا فشار دادن کلید F5 نتایج این عبارات را در ناحیه‌ی نتایج برمی‌گرداند.

شکل(۱-۱۰) Query Editor به همراه یک پرس‌وجو و نتایج آن

نکته

می‌توان چندین پنجره‌ی مختلف را باز کرد (چندین اتصال مختلف به یک یا چند نمونـــه‌ی Database Engine وجود دارد).اتصال‌های جدیـــدی را بـــا کلیک روی دکمـــه‌ی New Query در نوارابزار ایجاد کنید.

اطلاعات زیر،که مربوط به اجرای عبارات اســـت، در نوار وضعیت پایین پنجره‌ی Query

Editor نمایش داده می‌شوند:

- وضعیت عملیات فعلی (برای مثال، اجرای موفق پرس‌وجو)
- نام سرور پایگاه داده
- نام کاربر جاری و شناسه‌ی پردازش سرور
- نام پایگاه داده‌ی جاری
- زمان سپری شده برای اجرای آخرین پرس‌وجو
- تعداد سطرهای بازیابی شده

یکی از ویژگی های اصلی SQL Server Management Studio این است که به کارگیری آن ساده است و مؤلفه‌ی Query Editor را نیز ارائه می‌کند. Query Editor از ویژگی های فراوانی که کدنویسی عبارات Transact-SQL را ساده تر می‌کند پشتیبانی می‌نماید. Query Editor از برجسته کردن قواعد برای بهبود خوانایی عبارات Transact-SQL استفاده می‌کند. این ویراستار تمام کلمات رزرو شده را به رنگ آبی، متغیرها را به رنگ سیاه، رشته‌ها را به رنگ قرمز و توضیحات را به رنگ سبز نمایش می‌دهد.

Object Explorer نیز می‌تواند در ویرایش پرس‌وجوها به شما کمک کند. برای مثال، در صورتی که می‌خواهید عبارت CREATE DATABASE مربوط به پایگاه داده‌ی sample را مشاهده کنید، روی نام جدول کلیک راست نمائید و Script Database as و سپس CREATE to / New Query Editor Window را انتخاب کنید. شکل ۲-۱۰،پنجره‌ی Query Editorرا به همراه عبارت CREATE DATABASE (مربوط به پایگاه داده‌ی sample)نشان می‌دهد.

شکل (۲-۱۰) پنجره‌ی Query Editor همراه با عبارت CREATE DATABASE

در صورتی که بخواهید یک طرح اجرایی گرافیکی را برای یک پرس‌وجوی خاص نمایش دهید، Object Explorer خیلی مفید است. اگر از منوی Query گزینه‌ی Display Estimated Execution Plan را انتخاب کنید، سیستم طرح گرافیکی را به جای نتیجه‌ی پرس‌وجو نمایش خواهد داد (شکل ۳-۱۰).

شکل(۳-۱۰) طرح اجرایی برای پرس‌وجوی شکل ۱-۱۰

پایـگاه داده‌ای بـه نـام Test ایجاد کنید. پایـگاه داده را در فایلـی بـه نـام testdate_a در فهرسـت C:\temp ذخیره و فضـای ۱۰ مگابایت بـرای آن در نظر بگیرید. فایـل پایگاه داده را بـه نحـوی پیکربندی کنید که به صـورت افزایش ۲ مگابایتی رشـد کند و از ۲۰ مگابایت تجاوز نکند. فایل ثبت تراکنش برای پایگاه داده‌ی Test را با اندازه‌ی اولیه‌ی ۳مگابایت و اجازه‌ی رشد۲۰درصدی و امکان افزایش خودکار ایجاد کنید.

اتصال و قطع اتصال پایگاه داده‌ها

اتصال تمام داده‌های یک پایگاه داده را می‌توان از یک سرور پایگاه داده قطع کرد و سپس به همان سرور یا سرور دیگری متصل نمود. قطع اتصال و اتصال یک پایگاه داده باید هنگامی که می‌خواهید پایگاه داده را انتقال دهید، انجام شود.

می‌توان اتصال یک پایگاه داده را از یک سـرور پایگاه داده با اسـتفاده از روال سیستمی sp_detach_db قطع کرد(پایگاه داده‌ی قطع اتصال شده باید در حالت تک کاربر باشد).

برای اتصال پایگاه داده، از دسـتور CREATE DATABASE یا روال سیسـتمی sp_ attach_db استفاده کنید. هنگامی که یک پایگاه داده را متصل می‌کنید، باید تمام فایل‌ها قابل دسـترس باشند. اگر هر فایل داده‌ای مسیر متفاوتی دارد، هنگام ایجاد پایگاه داده، باید مسیر جاری فایل را تعیین کنید.

۳-۱-۱۰ ایجاد جدول

دستور CREATE TABLEیک جدول پایه‌ی جدیدی را به همراه تمام ستون‌ها و انواع داده‌هایشان ایجاد می‌کند. شکل کلی این دستور به صورت زیر است:

```
CREATE TABLE table_name
(col_name1 type1 [NOT NULL| NULL]
[{, col_name2 type2 [NOT NULL| NULL]} ...])
```

نکته

بـــه غیر از جدول‌هـــای پایه، انواع خاصـــی ازجدول‌های موقتی و دیدگاه‌ها نیز وجود دارند.

table_name نام جدول پایه‌ای اســـت که ایجاد می‌شود. حداکثر تعداد جدول‌ها در هر پایگاه داده به تعداد شـــیء‌های پایگاه داده محدود اســـت (می‌توان بیش از ۲ میلیارد شـــیء شـــامل جدول‌ها، دیدگاه‌ها، روال‌هـــای ذخیره شـــده، triggers و محدودیت‌ها ایجاد کرد). col_name1، col_name2 و... اسامی ستون‌های جدول هستند. type1، type2 و... انواع داده‌های مربوط به ستون‌ها هستند.

نکته

نام شیء پایگاه داده می‌تواند شامل چهار بخش باشد:

Server_name.[db_name.[schema_name.]]]object_name]

object_name نام شـــیء پایگاه داده است. shema_name نام شمایی است که شیء

به آن تعلق دارد. server_name و db_name اسامی سرور و پایگاه داده‌ای است که شِیء پایگاه داده متعلق به آن است. اسامی جدول که با نام شما ترکیب می‌شوند، باید درون پایگاه داده منحصر به فرد باشـــند. به طور مشابه اسامی ســـتون‌ها نیز باید داخل جدول، انحصاری باشند.

اولین مطلبی که در این جا شرح می‌دهیم، وجود داشتن و نداشتن مقادیر Null در یک ســـتون است. اگر Not Null برای یک ستون تعیین شده باشد، قرار دادن مقادیر Null غیر ممکن خواهد بود(در صورت Null گذاشتن ستون، پیغام خطا برگردانده می‌شود).

همان طور که بیان شـــد، یک شـــیء پایگاه داده (در این حالت، جدول)، همیشـــه درون شـــمایی از پایگاه داده ایجاد می‌شـــود. یک کاربر می‌تواند جدول را فقط در شمایی که مجوز ALTER دارد، ایجـــاد کند. هر کاربری با نقش‌های db_ddladmin ، sysadmin یا db_ owner می‌تواند جدول را در هر شمایی ایجاد کند.

ایجاد کننده‌ی جدول الزاماً نباید مالک آن باشـــد.به عبارت دیگر شـــما می‌توانید جدولی را که متعلق به شـــخص دیگری اســـت ایجاد کنید. به طور مشـــابه، یک جدول ایجاد شده با CREATE TABLE الزاماً مربوط به پایگاه داده‌ی جاری نیست. اگر جدولی از پایگاه داده‌ی دیگری را بخواهیم ایجاد کنیم، قبل از نام جدول، نام پایگاه داده را ذکر می‌کنیم.

شـــمایی که جدول مربوط به آن اســـت دارای دو نام پیش فرض اســـت. اگر جدول بدون ذکر نام شمای تعیین شده باشد، سیستم، نام جدول را در شمای پیش فرض مربوطه بررسی می‌کند. اگر نام شـــیء در شـــمای پیش فرض یافت نشود، سیستم، شمای dbo را جست جو می‌کند.

نکته

همیشه باید نام جدول را به همراه نام شمای مربوطه تعیین کنید.
با این عمل از اشکالات احتمالی جلوگیری می‌کنید.

جدول‌هـای موقتی، نوع خاصی از جدول مبنا هسـتند. این جدول‌هـا در پایگاه داده‌ی tempdb ذخیره می‌شوند و به طور خودکار در پایان جلسه کاری از بین می‌روند.

مثـال ۳، ایجاد تمام جدول‌های پایگاه داده‌ی sample را نشـان می‌دهد. (پایگاه داده‌ی sample باید پایگاه داده‌ی جاری باشد).

مثال ۳)

```
USE sample;
CREATE TABLE employee (emp_no INTEGER NOT NULL,
emp_fname CHAR(20) NOT NULL, emp_lname CHAR(20) NOT NULL,dept_no CHAR(4) NULL);
CREATE TABLE department(dept_no CHAR(4) NOT NULL,
dept_name CHAR(25) NOT NULL, location CHAR(30) NULL);
CREATE TABLE project (project_no CHAR(4) NOT NULL,
project_name CHAR(15) NOT NULL, budget FLOAT NULL);
CREATE TABLE works_on (emp_no INTEGER NOT NULL,
project_no CHAR(4) NOT NULL, job CHAR (15) NULL,
enter_date DATE NULL);
```

علاوه بر نوع داده و قابلیت داشتن مقادیر پوچ، مشخصات ستون می‌تواند شامل گزینه‌های زیر باشد:

- عبارت DEFAULT
- مشخصه‌ی IDENTITIY

عبارت DEFAULT در تعریف ستون، مقدار پیش فرض ستون را تعیین می‌کند (هر زمانی که رکورد جدیدی در جدول درج می‌شود، مقدار پیش فرض ستون در صورتی که مقداری تعیین نشود، مورد استفاده قرار خواهد گرفت).یک مقدار ثابت مثل توابع سیستمی USER،CURRENT_USER،SESSION_USER،SYSTEM_USER،CURRENT_TIMESTAMP و NULL می‌توانند به صورت مقادیر پیش فرض استفاده شوند.

یک ستون با مشخصه‌ی IDENTITY فقط به مقادیر عددی صحیح اعمال می‌شود که معمولاً به وسیله‌ی سیستم تعیین می‌شوند. هر مقداری که در ستون درج خواهد شد، با افزایش مقدار درج شده‌ی ستون، محاسبه می‌شود. بنابراین، تعریف یک ستون با مشخصه‌ی IDENTITY شامل یک مقدار آغازین و یک افزایش است. این مشخصه در فصل بعدی، شرح داده خواهد شد.

نکته

به دلیل این که موتور پایگاه داده مقادیر را به همراه مشخصه‌ی IDENTITY تولید می‌کند، این مقادیر همیشه متفاوت هستند (حتی هنگامی که چندین کاربر سطرها را به طور هم‌زمان اضافه می‌کنند). این ویژگی در یک محیط چندکاربره مفید است.

در خاتمه‌ی این بخش،به مثال ۴ که ایجاد یک جدول با ستونی از نوع SQL_VARIANT را نشان می‌دهد،توجه کنید.

مثال ۴)

```
USE sample;
CREATE TABLE Item_Attributes (
item_id INT NOT NULL, attribute NVARCHAR(30) NOT NULL,
value SQL_VARIANT NOT NULL, PRIMARY KEY (item_id, attribute) )
```

در مثال ۴، جدول شامل ستون value است که از نوع SQL_VARIANT است. همان طور که در فصل قبل نیز بیان شد، نوع داده‌ی SQL_VARIANT می‌تواند انواع مختلف داده‌ها مثل مقادیر عددی، رشته‌ای و تاریخ را به طور هم‌زمان ذخیره کند. توجه کنید که در مثال ۴، نوع داده‌ی SQL_VARIANT برای مقادیر ستون استفاده شده است، زیرا مقادیر صفت‌های مختلف ممکن است نوع داده‌ی متفاوتی داشته باشند. برای مثال، صفت اندازه، یک مقدار عددی صحیح را ذخیره می‌کند و صفت نام، یک رشته‌ی کاراکتری را ذخیره می‌کند.

٤-١-١٠ ایجاد جدول و تعریف محدودیت‌های جامعیت[1]

یکی از مهم ترین ویژگی‌هایی که یک DBMS باید ارائه کند، روش نگه داری جامعیت داده‌هاست. محدودیت‌هایی را که برای بررسی تغییر یا درج داده‌ها مورد استفاده قرار می‌گیرند محدودیت‌های جامعیت می‌نامند. وظیفه‌ی نگه داری جامعیت می‌تواند به وسیله‌ی کاربر در برنامه‌ی کاربردی یا به وسیله‌ی DBMS انجام شود. مهم ترین مزایای اجرای جامعیت به وسیله‌ی DBMS عبارت اند از:

- افزایش قابلیت اعتماد داده‌ها

1- integrity

- کاهش زمان برنامه‌نویسی
- نگه داری ساده

استفاده از DBMS برای تعریف جامعیت، قابلیت اعتماد داده‌ها را افزایش می‌دهد،زیرا امکان فراموش شدن آن به وسیله‌ی برنامه‌نویس از بین می‌رود.

جامعیتی که به وسیله‌ی DBMS اعمال نشود، باید در هر برنامه‌ی کاربردی کننده از داده‌ها استفاده می‌کند، تعریف شود. ولی جامعیت یک سان به وسیله‌ی DBMS و فقط یک بار تعریف می‌شود. به علاوه، محدودیت‌های اعمال شده در برنامه‌ی کاربردی معمولاً از محدودیت‌های پایگاه داده پیچیده‌ترند.

اگر یک جامعیت به وسیله‌ی DBMS اجرا شود، تغییر ساختار جامعیت فقط یک بار انجام می‌شود. تغییر یک ساختار در برنامه‌ی کاربردی به ویرایش هر برنامه‌ای که شامل کد مربوطه است،نیاز دارد.

دو گروه از محدودیت‌های جامعیت که به وسیله‌ی DBMS اجرا می‌شوند، عبارت‌اند از:

- محدودیت‌های جامعیت تعریف
- محدودیت‌های جامعیت روالی که به وسیله‌ی ریزبرنامه‌ها اجرا می‌شوند.

محدودیت‌های تعریفی با استفاده از دستورات DDL مثل CREATE TABLE و ALTER TABLE تعریف می‌شوند. این محدودیت‌ها می‌توانند در سطح ستون یا سطح جدول باشند. محدودیت‌های سطح ستون همراه با نوع داده‌ها و سایر مشخصات ستون در تعریف ستون قرار می‌گیرند، در حالی که محدودیت‌های سطح جدول، همیشه در پایان دستور CREATE TABLE یا ALTER TABLE، بعد از تعریف تمام ستون‌ها، تعریف می‌شوند.

نکته

فقط یک تفاوت بین محدودیت‌های سطح ستون و سطح جدول وجود دارد: یک محدودیت سطح ستون فقط به یک ستون اعمال می‌شود، در حالی که محدودیت سطح جدول، یک یا چند ستون از جدول را تحت تأثیر قرار می‌دهد.

هر محدودیت تعریفی دارای نامی است.نام محدودیت می‌تواند با استفاده از گزینه‌ی CONSTRAINT در دستور CREATE TABLE یا ALTER TABLE تعریف شود. اگر گزینه‌ی CONSTRAINT حذف شود، موتور پایگاه داده، نامی را برای آن تعیین می‌کند.

تمام محدودیت‌های تعریفی می‌توانند در چندین گروه تقسیم بندی شوند:

- عبارت DEFAULT
- عبارت UNIQUE
- عبارت PRIMARY KEY
- عبارت CHECK
- عبارت FOREIGN KEY و جامعیت ارجاعی

تعریف مقدار پیش فرض با استفاده از عبارت DEFAULT، قبلاً در این فصل شرح داده شد. تمام محدودیت‌های دیگر در بخش‌های زیر شرح داده می‌شوند.

عبارت UNIQUE

بعضی مواقع بیش از یک ستون یا گروهی از ستون‌های جداول دارای مقادیر منحصر به فرد هستند. بنابراین، می‌توان از آن ها به عنوان کلید اصلی استفاده کرد. تمام ستون‌ها را یا گروهی از ستون‌ها که به عنوان کلیدهای اصلی شناسایی می‌شوند،کلیدهای کاندید می‌نامند. هر کلید کاندید با استفاده از عبارت UNIQUE در دستور CREATE TABLE یا ALTER TABLE تعریف می‌شود.

عبارت UNIQUE دارای شکل کلی زیر است:

```
[CONSTRAINT c_name]
UNIQUE [CLUSTERED | NONCLUSTERED] ({ col_name1} ,...)
```

گزینه‌ی CONSTRAINT در عبارت UNIQUE یک نام مشخصی برای کلید کاندید تعیین می‌کند. گزینه‌ی CLUSTERED یا NONCLUSTERED مربوط به این است که موتور پایگاه داده شاخصی را برای هر کلید کاندید جدول تولید می‌کند. شاخص می‌تواند خوشه‌بندی شود (ترتیب فیزیکی از مقادیر ستون). اگر ترتیب مشخص نشود، شاخص خوشه‌بندی نمی‌شود. مقدار پیش فرض، NONCLUSTERED است. col_name۱ نام ستونی است که کلید کاندید را ایجاد می‌کند (حداکثر تعداد ستون‌ها به ازای هر کلید کاندید، ۱۶ تاست).

مثال ۵، استفاده از عبارت UNIQUE را نشان می‌دهد.

مثال ۵)

```
USE sample;
CREATE TABLE projects (project_no CHAR(4) DEFAULT 'p1',
project_name CHAR(15) NOT NULL, budget FLOAT NULL
CONSTRAINT unique_no UNIQUE (project_no));
```

هر مقدار از ستون project_no از جدول projects منحصر به فرد است که شامل مقدار NULL نیز هست. اگر مقدار تکراری در ستون project_no درج شود، سیستم، آن را قبول نمی‌کند. نام محدودیتی که در مثال ۵ تعریف شده، unique_no است.

عبارت PRIMARY KEY

Primary key یک جدول، ستون یا گروهی از ستون‌هاست که مقادیر آن ها در هر سطر متفاوت است. هر کلید اصلی با استفاده از عبارت PRIMARY KEY در دستور CREAT TABLE یا ALTER TABLE تعریف می‌شود.

شکل کلی عبارت PRIMARY KEY به صورت زیر است:

```
[CONSTRAINT c_name]
PRIMARY KEY [CLUSTERED | NONCLUSTERED] ({col_name۱} ,...)
```

تمام گزینه‌های عبارت PRIMARY KEY دارای همان معنی در عبارت UNIQUE هستند. ستون PRIMARY KEY باید NOT NULL باشد و مقدار پیش فرض آن CLUSTERED است.

مثال ۶، مشخصات کلید اولیه برای جدول employee از پایگاه داده‌ی sample را نشان می‌دهد.

نکته

قبل از اجرای مثال زیر، باید جدول employee را که قبلا ایجاد کرده‌اید،حذف کنید (DROP TABLE employee).

مثال ۶)

```
USE sample;
CREATE TABLE employee (emp_no INTEGER NOT NULL,
emp_fname CHAR(۲۰) NOT NULL,
emp_lname CHAR(۲۰) NOT NULL,
dept_no CHAR(۴) NULL,
CONSTRAINT prim_empl PRIMARY KEY (emp_no));
```

جدول employee دوباره ایجاد شـــده و کلید اصلی آن در مثال ۶ تعریف شـــده است. کلید اصلی جدول با استفاده از محدودیت جامعیت تعریفی به نام prim_empl تعیین شده اســـت. این جامعیت، یک محدودیت ســـطح جدولی است، زیرا بعد از تعریف تمام ستون‌های جدول employee تعیین شده است.

مثال ۷ معادل مثال ۶ اســـت با این تفاوت که مشخصات کلید اصلی جدول employee به صورت یک محدودیت سطح ستونی است.

نکته

باز هم قبل از اجـــرای مثال زیر، باید جدول employee را حذف کنید (DROP TABLE employee).

مثال ۷)

```
USE sample;
CREATE TABLE employee
```

```
(emp_no INTEGER NOT NULL CONSTRAINT prim_empl Primary Key,
emp_fname CHAR(20) NOT NULL,
emp_lname CHAR(20) NOT NULL,
dept_no CHAR(4) NULL);
```

در مثال ۷، عبارت PRIMARY KEY مربوط به تعریف ستون است که همراه نوع داده آمده است. به همین دلیل، محدودیت در سطح ستونی نامیده می‌شود.

عبارت CHECK

check constraint برای داده‌های درج شـــده در یک ستون، شرایط تعیین می‌کند. هر ســـطر درج شـــده در جدول یا هر مقدار ویرایش شده در ستون، باید حائز این شرایط باشند. عبارت CHECK برای تعیین محدودیت‌های بررســـی،به کار گرفته می‌شـــود. این عبارت در دســـتور CREATE TABLE یا ALTER TABLE تعریف می‌شود. شکل کلی این عبارت به صورت زیر است:

```
[CONSTRAINT  c_name]
CHECK [NOT FOR REPLICATION] expression
```

expression باید یک مقدار منطقی (true یا false) باشد و می‌تواند به هر ستونی در جدول جاری و نه در جدول دیگر، ارجاع داده شود. اگر گزینه‌ی NOT FOR REPLICATION وجود نداشته باشد، در زمان تکرار داده‌ها، عبارت CHECK اعمال نمی‌شود.

مثال ۸، چگونگی استفاده از عبارت CHECK را نشان می‌دهد.

مثال ۸)

```
USE sample;
CREATE TABLE customer
(cust_no INTEGER NOT NULL,
cust_group CHAR(3) NULL,
```

```
CHECK (cust_group IN ('c1', 'c2', 'c10')));
```

جدول customer که در مثال ۸ ایجاد شده است شامل ستون cust_group با محدودیت بررسی مربوطه است. سیستم پایگاه داده، در صورتی که ستون cust_group بعد از ویرایش مقادیر موجود یا بعد از درج یک سطر جدید شامل مقداری متفاوت از مجموعه مقادیر (’c10‘ و ’c2‘ و ’c1‘) باشد، خطای احتمالی را برمی‌گرداند.

عبارت FOREIGN KEY

foreign key ستون یا گروهی از ستون‌ها در جدول است که شامل مقادیری است که با مقادیر کلید اصلی در جدول دیگر منطبق است. کلید خارجی با استفاده از عبارت FOREIGN KEY همراه با عبارت REFERENCES تعریف می‌شود.

شکل کلی این عبارت به صورت زیر است:

```
[CONSTRAINT c_name]
[[FOREIGN KEY] ({col_name۱} ,...)]
REFERENCES table_name ({col_name۲},...)
[ON DELETE {NO ACTION| CASCADE | SET NULL | SET DEFAULT}]
[ON UPDATE {NO ACTION | CASCADE | SET NULL | SET DEFAULT}]
```


عبارت FOREIGN KEY تمام ستون‌ها را به طور روشن تعریف می‌کند که بستگی به کلید خارجی دارد. عبارت REFERENCES نام جدول را به همراه تمام ستون‌هایی که کلید اولیه ی مربوطه را ایجاد می‌کنند، تعیین می‌نماید. تعداد و نوع داده‌های ستون‌ها در عبارت FOREIGN KEY باید مطابق تعداد و نوع داده‌های ستون‌ها در عبارت REFERENCES باشد(و البته هر دوی این ها باید مطابق با کلید اولیه در جدول ارجاع شده باشند).

جدولی را که شـامل کلید خارجی اسـت جدول ارجاع و جدولی را که شامل کلید اصلی اسـت را جدول والد یا جدول رجوع کننده می‌نامند. مثال ۹، مشـخصات کلید خارجی در جدول works_on از پایگاه داده‌ی sample را نشان می‌دهد.

نکته

قبل از اجرای مثال زیر، باید جدول works_on را حذف کنید.

مثال ۹)

```
USE sample;
CREATE TABLE works_on (emp_no INTEGER NOT NULL,
project_no CHAR(4) NOT NULL,
job CHAR (15) NULL,
enter_date DATE NULL,
CONSTRAINT prim_works PRIMARY KEY(emp_no, project_no),
CONSTRAINT foreign_works FOREIGN KEY(emp_no)
REFERENCES employee (emp_no));
```


جـدولworks_on در مثال ۹ با دو جامعیت تعیین شـده اسـت: prim_works و foreign_works. هر دو محدودیت در سـطح جدولی هسـتند.کلیدهای خارجی در یک جـدول، حداکثر ۶۳ عدد اسـت. یکی از مهم تریـن موارد مربوط به تعریـف کلید خارجی، جامعیت ارجاعی است.

۵-۱-۱۰ جامعیت ارجاعی

جامعیت ارجاعی، قواعد درج و ویرایش جدول‌ها را به همراه کلید خارجی و کلید اصلی مربوطه اعمال می‌کند. مثال‌های ۶ و ۹، دو محدودیت از این نوع را تعیین می‌کنند: prim_empl و foreign_works. عبارت REFERENCES در مثال ۹، جدول employee را به صورت جدول والد تعیین می‌کند.

مشکلات ممکن در ارتباط با جامعیت ارجاعی

چهار حالت در ویرایش مقادیر کلید خارجی یا کلید اولیه وجود دارد که می‌توانند سبب بروز مشکلات شوند. تمام این حالت‌ها با استفاده از پایگاه داده‌ی sample نشان داده خواهند شد. دو حالت اول روی تغییرات جدول ارجاع و دو حالت بعدی، روی جدول والد تأثیر می‌گذارند.

حالت اول

سطر جدیدی را در جدول works_on با شماره ی کارمندی ۱۱۱۱۱ درج کنید.

درج سطر جدید در جدول ارجاع works_on،که با شماره ی کارمندی در جدول employee مطابقت ندارد. اگر جامعیت ارجاعی برای هر دو جدول، به صورتی که در مثال‌های ۶ و ۹ انجام شد، تعیین شده باشد، موتور پایگاه داده از درج سطر جدید جلوگیری می‌کند.

حالت دوم

شماره ی کارمندی ۱۰۱۰۲ را در همه‌ی سطرهای جدول works_on به مقدار جدید ۱۱۱۱۱ تغییر دهید.

در حالــت ۲، مقدار موجــود کلید خارجی در جــدول works_on باید با مقدار جدید جای گزین شــود، که مقدار منطبقی در جدول والد employee ندارد. اگر جامعیت ارجاعی برای هر دو جدول تعیین شــده باشــد، سیســتم پایــگاه داده، تغییر ســطرها را در جدول works_on را قبول نمی‌کند.

حالت سوم

شــماره‌ی کارمندی ۱۰۱۰۲ در ســطر مربوطه از جدول employee را به مقدار جدید ۲۲۲۲۲ تغییر دهید.

در حالــت ۳، مقدار موجود کلید اصلی در جدول والد و کلید خارجی جدول ارجاع فقط در جدول والد ویرایش شــده است. مقادیر در جدول ارجاع تغییر نکرده‌اند. بنابراین سیستم، ویرایش سطری با شماره‌ی کارمندی ۱۰۱۰۲ در جدول employee را رد می‌کند. جامعیت ارجاع تعیین می‌کند که هیچ ســطری در جدول ارجاع نمی‌تواند وجود داشــته باشــد مگر این که ســطر مربوطه در جدول والد نیز وجود داشــته باشــد. در غیر این صورت، سطرهای جدول والد، "بدون والد" خواهند بود.

حالت چهارم

سطری از جدول employee را که شماره ی کارمندی ۱۰۱۰۲ دارد،حذف کنید.

حالت ۴ شــبیه حالت ۳ است. این عمل حذف، ســبب خواهد شد که سطرهای منطبق در جــدول ارجاع نیز حذف شــوند. مثال ۱۰،تعریف جدول‌های پایــگاه داده‌ی sample با کلید اولیه و کلید خارجی را نشــان می‌دهد(اگر جدول‌های ،employee، department project و works_on قبلاً ایجاد شــده‌اند، ابتدا آن ها را با اســتفاده از دستور DROP حذف کنید).

مثال ۱۰)

```
CREATE TABLE department(dept_no CHAR(4) NOT NULL,
dept_name CHAR(25) NOT NULL, location CHAR(30) NULL,
CONSTRAINT prim_dept PRIMARY KEY (dept_no));

CREATE TABLE employee (emp_no INTEGER NOT NULL,
emp_fname CHAR(20) NOT NULL, emp_lname CHAR(20) NOT NULL,
dept_no CHAR(4) NULL, CONSTRAINT prim_emp PRIMARY KEY (emp_no),
CONSTRAINT foreign_emp FOREIGN KEY(dept_no) REFERENCES
department(dept_no));

CREATE TABLE project (project_no CHAR(4) NOT NULL,
project_name CHAR(15) NOT NULL, budget FLOAT NULL,
CONSTRAINT prim_proj PRIMARY KEY (project_no));
```

```
CREATE TABLE works_on (emp_no INTEGER NOT NULL,
project_no CHAR(4) NOT NULL, job CHAR (15) NULL,
enter_date DATE NULL,
CONSTRAINT prim_works PRIMARY KEY(emp_no, project_no),
CONSTRAINT foreign1_works FOREIGN KEY(emp_no) REFERENCES
employee(emp_no),
CONSTRAINT foreign2_works FOREIGN KEY(project_no) REFERENCES
project(project_no));
```


۶-۱-۱۰ ایجاد سایر شیءهای پایگاه داده

یک پایگاه داده‌ی رابطه‌ای به جز جدول‌های پایه شامل دیدگاه‌ها (views) نیز هست که همان جدول‌های مجازی هستند. داده‌های یک جدول پایه به طور فیزیکی وجود دارند (روی دیسک ذخیره شده‌اند)، در حالی که یک دیدگاه از یک یا چند جدول پایه، مشتق می‌شود. دستور CREATE VIEW یک دیدگاه جدیدی را از یک یا چند جدول (یا دیدگاه) با

استفاده از دستور SELECT ایجاد می‌کند که یک بخش جدانشدنی از دستور CREATE VIEW است.

از آن جایی که ایجاد یک دیدگاه همیشه شامل یک پرس‌وجو است، دستور CREATE VIEW مربوط به زبان کارکردن با داده‌ها (DML) به جای زبان تعریف داده‌ها (DDL) است. به همین دلیل، ایجاد و حذف دیدگاه‌ها بعد از تمام دستورات ویرایش داده‌ها و در فصل‌های بعدی بیان می‌شود.

دستور CREATE INDEX شاخص جدیدی را برای جدول تعیین شده ایجاد می‌کند. شاخص‌ها دسترسی کارآمد به داده‌های ذخیره شده روی دیسک را فراهم می‌کنند. وجود یک شاخص می‌تواند دسترسی به داده‌ها را به طور چشم گیری بهبود بخشد. شاخص‌ها به همراه دستور CREATE INDEX در فصل‌های بعدی شرح داده می‌شوند.

یک روال ذخیره شده، شیء پایگاه داده‌ی دیگری است که می‌تواند با استفاده از دستور CREATE PROCEDURE ایجاد شود(روال ذخیره شده، دنباله‌ای از دستورات نوشته شده در Transact-SQL است).

trigger یک شیء پایگاه داده است که عملی را به منزله ی نتیجه‌ی یک عملیات تعیین می‌کند.به عبارت دیگر،موتور پایگاه داده ها زمان انجام عمل اصلاح داده‌ها (ویرایش، درج یا حذف) روی یک جدول خاص، یک یا چند عمل اضافی را انجام می‌دهد. دستور CREATE TRIGGER یک trigger جدید را ایجاد می‌کند.

synonym یک شیء پایگاه داده‌ی محلی است که پیوندی بین خود و شیء دیگری را به وسیله‌ی سرور پایگاه داده‌ی یک سان یا پیوندشده ارائه می‌کند. با استفاده از دستور CREATE SYNONYM می‌توان یک مترادف جدیدی را برای شیء موجود ایجاد کرد. مثال ۱۱،استفاده از این دستور را نشان می‌دهد.

مثال(۱۱)

```
USE sample;
CREATE SYNONYM prod
FOR AdventureWorks.Production.Product;
```

مثـال ۱۱،مترادفی را برای جـدول Product در Production Schema از پایگاه داده‌ی Adventureworks ایجاد می‌کند. این مترادف می‌تواند در دستورات DML مثل SELECT، INSERT، UPDATE و DELETE مورد استفاده قرار گیرد.

نکته

دلیل اصلی اســتفاده از مترادف‌ها، پرهیز از اســامی طولانی در دستورات DML اســت. همان طور که می‌دانید، نام یک شیء پایگاه داده می‌تواند شامل چهار بخش باشــد. معرفی یک مترادف (تک بخشی) برای یک شیء که دارای سه یا چهار بخش است، می‌تواند سبب صرفه‌جویی در زمان تایپ نام شود.

یک شما، شــیء پایگاه داده‌ای است که شامل دستوراتی برای ایجاد جدول‌ها، دیدگاه‌ها و مجوزهای کاربر اســت(می‌توان شما را به صورت ســاختاری تصور کرد که چندین جدول، دیدگاه‌های مربوطه و مجوزهای کاربری را با هم جمع کرده است).

۷-۱-۱۰ محدودیت‌های موجودیت و دامنه‌ها

یک دامنه، مجموعه‌ای از تمام مقادیر معتبر ممکن اســت که ســتون‌های جدول ممکن اســت در بــر بگیرند. تقریباً تمام انواع داده‌های اصلی مثــل CHAR، INT و DATE برای تعریف مجموعه‌ای از مقادیر ممکن ســتون، مورد اســتفاده قرار می‌گیرند. این روش اعمال

"جامعیت دامنه‌ای" ،همان طور که در مثال زیر مشاهده می‌کنید، کامل نیست.

جدول person دارای ستون zip است.این جدول کدپستی شهری را که شخص در آن زندگی می‌کند را تعیین می‌نماید. این ستون می‌تواند با استفاده از نوع داده‌ی SMALLINT یا CHAR(۵) تعریف شود. تعریف با نوع داده‌ی SMALLINT دقیق نیست، زیرا نوع داده‌ی SMALLINT شامل تمام مقادیر مثبت و منفی بین $-۲^{۱۵}$ و $۲^{۱۵}$ است. تعریف با استفاده از CHAR(۵) خیلی دقیق‌تر است، زیرا تمام کاراکترها و علایم خاص نیز می‌توانند در چنین حالتی مورد استفاده قرار گیرند. بنابراین، یک تعریف دقیق از کدپستی نیاز به تعریف بازه‌ای از اعداد صحیح مثبت بین ۰۰۶۰۱ و ۹۹۹۵۰ و انتساب آن به ستون zip دارد.

محدودیت CHECK می‌تواند جامعیت دامنه‌ای دقیق‌تری را اعمال کند، زیرا عبارات انعطاف بیش تری دارند و همیشه هنگام درج یا ویرایش ستون اعمال می‌شوند.

زبان Transact-SQL پشتیبانی از دامنه‌ها را با ایجاد انواع داده‌های مستعار به کمک دستور CREATE TYPE فراهم می‌کند. دو قسمت بعدی، انواع داده‌های مستعار و CLR را شرح می‌دهند.

پایگاه داده‌ی بیمارستان (Hospital) و جدول‌های آن را، که قبلاً مورد بررسی قرار گرفته است، به کمک دستورات Transact-SQL ایجاد و تمام قوانین جامعیت موردنیاز را نیز اعمال کنید.

۱۰-۲ ویرایش شیءهای پایگاه داده

از طریق ابزار Management Studio یا دستورات SQL می‌توان مشخصات یک پایگاه داده و اجزای آن را ویرایش کرد. زبان Transact-SQL از تغییر ساختار شیءهای زیر که اجزای پایگاه داده هستند، پشتیبانی می‌کند:

- Database
- Table
- Stored Procedure
- View
- Schema
- Trigger

دو بخش زیر، چگونگی ویرایش ساختار پایگاه داده و جدول را شرح می‌دهند.

۱۰-۲-۱ ویرایش پایگاه داده

دستور ALTER DATABASE ساختار فیزیکی یک پایگاه داده را تغییر می‌دهد. زبان Transact-SQL امکان تغییر مشخصات زیر از پایگاه داده را ارائه می‌کند:

تغییر نام پایگاه داده با استفاده از روال ذخیره شده‌ی sp_rename

افزودن یا حذف یک یا چند فایل پایگاه داده

افزودن یا حذف یک یا چند فایل Log

افزودن یا حذف گروه‌های فایل

تغییر مشخصات فایل یا گروه فایل

تعیین گزینه‌های پایگاه داده

در بخش‌های زیر این تغییرات شرح داده می‌شوند.

اضافه یا حذف کردن فایل‌های پایگاه داده، فایل‌های ثبت یا گروه‌های فایلی

دستور ALTER DATABASE امکان اضافه و حذف کردن فایل‌های پایگاه داده را فراهم می‌کند. عبارت‌های ADD FILE و REMOVE FILE، افزودن یک فایل جدید و حذف فایل موجود را به ترتیب ارائه می‌کنند. (به علاوه، یک فایل جدید با استفاده از گزینه‌ی TO FILE GROUP به یک گروه فایل جدید، منتسب می شود.)

مثال ۱۲، نشان می‌دهد که چگونه یک فایل پایگاه داده‌ی جدید به پایگاه داده‌ی projects اضافه می شود.

مثال ۱۲)

```
USE master;
GO
ALTER DATABASE projects
ADD FILE (NAME=projects_dat1,
FILENAME = 'C:\projects1.mdf', SIZE = 10,
MAXSIZE = 100, FILEGROWTH = 5);
```

دستورALTER DATABASE در این مثال، یک فایل جدید را به همراه نام منطقی projects_dat1 اضافه می‌کند. اندازه‌ی اولیه‌ی این فایل ۱۰مگابایت است و برحسب واحدهای ۵ مگابایتی افزایش اندازه پیدا می کند تا به ۱۰۰مگابایت برسد.

عبارت REMOVE FILE یک یا چند فایل را که به پایگاه داده‌ی موجود مربوط اند،حذف می‌کند. فایل می‌تواند یک فایل داده‌ای یا ثبت باشد. فایل حذف نمی‌شود مگر این که خالی باشد.

فایل‌های ثبت نیز به همان روش فایل‌های پایگاه داده اضافه می‌شوند. تنها تفاوت این است که از عبارت ADD LOG FILE به جای ADD FILE استفاده می‌شود.

عبـارت CREATE FILEGROUP یک گروه فایل جدیدی را ایجاد می‌کند، در حالی که DELETE FILEGROUP یک گروه فایل موجود را از سیستم حذف می‌کند. یک گروه فایل اگر خالی نباشد، حذف نمی‌شود.

تغییر مشخصات فایل یا گروه فایل

می‌توان از عبارت MODIFY FILE برای تغییر مشخصات فایلی زیر استفاده کرد:

تغییر نام منطقی فایل با استفاده از گزینه‌ی NEWNAME

افزایش مقدار مشخصه‌ی SIZE

تغییر مشخصه‌ی FILENAME، MAXSIZE یا FILEGROWTH

نشانه‌گذاری فایل به صورت OFFLINE

به طور مشـابه می‌توان از عبارت MODIFY FILEGROUP برای تغییر مشـخصات گروه فایل استفاده کرد:

تغییر نام گروه فایل با استفاده از گزینه‌ی NAME

نشـانه‌گذاری گـروه فایـل به صـورت گروه فایـل پیش فرض بـا اسـتفاده از گزینه‌ی DEFAULT

نشـانه‌گذاری گـروه فایل به صـورت فقط خواندنی یا خواندنی-نوشـتنی با اسـتفاده از گزینه‌های READONLY یا READWRITE

تنظیم گزینه‌های پایگاه داده

عبارت SET از دسـتور ALTER DATABASE برای تعیین گزینه‌های مختلف پایگاه داده مورد استفاده قرار می‌گیرد. بعضی از گزینه‌ها باید با ON یا OFF مقداردهی شوند ولی بعضی از آن ها دارای لیسـتی از مقادیر ممکن هسـتند. هر گزینه‌ی پایگاه داده دارای مقدار پیش فرض است که در پایگاه داده‌ی model مقداردهی شده است. بنابراین، می‌توان پایگاه

داده‌ی model را برای تغییر مقادیر پیش فرض گزینه‌های تعیین شده تغییر داد.

تمـــام گزینه‌هایی که می‌توانید تنظیم کنید، به چند گروه تقســـیم می‌شـــوند. مهم ترین گروه‌ها عبارت‌اند از:

- گزینه‌های state
- گزینه‌های Auto
- گزینه‌های SQL

گزینه‌های وضعیت، موارد زیر را کنترل می‌کنند:

دسترســـی کاربـــر بـــه پایـــگاه داده (گزینه‌هـــا عبارت‌انـــد از:،SINGLE_USER RESTRICTED_USER و MULTI_USER)

وضعیت پایگاه داده (گزینه‌ها عبارت‌اند از:OFFLIN،ONLINE و EMERGENCY)

وضعیت خواندن/نوشتن (گزینه‌ها عبارت‌اند از:READ_ONLY و READ_WRITE)

۱۰-۲-۲ ویرایش جدول

دســـتور ALTER TABLE شـــمای یک جدول را ویرایش می‌کنـــد. زبان Transact-SQL امکان تغییر انواع زیر را فراهم می‌کند:

افزودن یا حذف یک یا چند ستون؛

ویرایش مشخصات ستون؛

- اضافه یا حذف محدودیت‌های جامعیت؛
- فعال یا غیرفعال کردن محدودیت‌ها؛
- تغییر نام جدول‌ها و سایر شیءهای پایگاه داده.
- بخش‌های زیر، این انواع تغییرات را شرح می‌دهند.

افزودن یا حذف یک ستون جدید

می‌توان از عبارت ADD دستور ALTER TABLE برای اضافه کردن یک ستون جدید به جدول موجود اســتفاده کرد.در هر دســتور ALTER TABLE فقط می‌توان یک ستون اضافه کرد. مثال ۱۳ استفاده از عبارت ADD را نشان می‌دهد.

مثال ۱۳)

```
USE sample;
ALTER TABLE employee
ADD telephone_no CHAR(12) NULL;
```

دســتور ALTER TABLE در ایــن مثــال، ســتون telephone_no را بــه جدول employee اضافــه می‌کنــد. موتــور پایگاه داده، ســتون جدیــد را با NULL یــا مقادیر IDENTITY یا با پیش فرض تعیین شده مقدار دهی می‌کند. به همین دلیل، ستون جدید باید مقدار پوچ یا پیش فرض داشته باشد.

نکته

هیچ روشی برای درج یک ستون جدید در محل خاصی از جدول وجود ندارد. ستونی که با استفاده از عبارت ADD اضافه می‌شود، همیشه در پایان جدول درج می‌شود.

عبارت DROP COLUMN قابلیت حذف یک ستون موجود در جدول را، که در مثال ۱۴ نشان داده شده است، ارائه می‌کند.

مثال ۱۴)

```
USE sample;
ALTER TABLE employee
DROP COLUMN telephone_no;
```

دستور ALTER TABLE در این مثال، ستون telephone_no را حذف می‌کند که به جدول employee با دستور ALTER TABLE در مثال ۱۳ اضافه شده است.

تغییر مشخصات ستون

زبان Transact-SQL از عبارت ALTER COLUMN در ALTER TABLE پشتیبانی می‌کند تا مشخصات ستون موجود را ویرایش کنید. مشخصاتی از ستون که می‌توان ویرایش کرد، عبارت‌اند از:

نوع داده

قابلیت پوچ بودن

مثال ۱۵، استفاده از عبارت ALTER COLUMN را نشان می‌دهد.

مثال ۱۵)

```
USE sample;
ALTER TABLE department
ALTER COLUMN location CHAR(۲۵) NOT NULL;
```


دستور ALTER TABLE در این مثال، مشخصات قبلی (CHAR(30), NULL) ستون Location از جدول department را به مشخصات جدید (CHAR(25),NOT NULL) تغییر می‌دهد.

افزودن یا حذف محدودیت‌های جامعیت

یک جامعیت جدید می‌تواند با استفاده از دستور ALTER TABLE و گزینه‌ی آن به نام

ADD CONSTRAINT اضافه شـــود.مثال۱۶، نشان می‌دهد که چگونه می‌توان از عبارت ADD CONSTRAINT در ارتباط با محدودیت بررسی استفاده کرد.

مثال ۱۶)

```
USE sample;
CREATE TABLE sales
(order_no INTEGER NOT NULL,
order_date DATE NOT NULL,
ship_date DATE NOT NULL);
ALTER TABLE sales
ADD CONSTRAINT order_check CHECK(order_date <= ship_date);
```

دســـتور CREATE TABLE در مثال فوق، جدول sales را با دو ســـتون از نوع داده‌ی DATE ایجـــاد می‌کند که عبارت‌اند از: order_date و ship_date. دســـتور ALTER TABLE یک جامعیت به نام order_check را تعریف و هر دو مقدار را مقایسه می‌کند و در صورتی که تاریخ خرید قبل از تاریخ سفارش باشد، پیغام خطایی را نمایش می‌دهد.

مثال ۱۷ نشـــان می‌دهد که چگونه می‌توان از دســـتور ALTER TABLE برای تعریف کلید اصلی جدول استفاده کرد.

مثال ۱۷)

```
USE sample;
ALTER TABLE sales
ADD CONSTRAINT primaryk_sales PRIMARY KEY(order_no);
```

دستور ALTER TABLE در مثال فوق، کلید اولیه‌ی جدول sales را تعریف می‌کند.

هــر جامعیتی را می‌توان با اســتفاده از عبارت DROP CONSTRAINT در دســتور ALTER TABLE حذف کرد (مثال ۱۸ را مشاهده کنید).

مثال ۱۸)

```
USE sample;
ALTER TABLE sales
DROP CONSTRAINT order_check;
```

دستور ALTER TABLE در مثال فوق، محدودیت CHECK به نام order_check را،که در مثال ۱۶ تعیین شده بود،حذف می‌کند.

نکته

از دستور ALTER TABLE نمی‌توان برای تغییر تعریف یک جامعیت استفاده کرد. در این حالت، محدودیت باید دوباره ایجاد شود.

فعال یا غیرفعال کردن محدودیت‌ها

همان طور که قبلاً نیز بیان شد، یک جامعیت دارای نامی است که می‌تواند به طور واضح با استفاده از گزینه‌ی CONSTRAINT یا به وسیله‌ی سیستم تعریف شود.

چگونه می‌توان اسامی محدودیت‌های اعمال شده را مشاهده کرد؟

پاسخ: نام تمام محدودیت‌های تعریف شده‌ی جدول را می‌توان با استفاده از روال سیستمی sp_helpconstraint مشاهده کرد.

یک محدودیت به طور پیش فرض در طول عملیات درج و ویرایش بعدی، اعمال می‌شود. به علاوه، مقادیر موجود در ستون(ها) نیز در مقابل محدودیت بررسی می‌شوند. در غیر این صورت، یک محدودیت که با گزینه‌ی WITH NOCHECK ایجاد شده است، در حالت دوم غیر فعال می‌شود. به عبارت دیگر، اگر از گزینه‌ی WITH NOCHECK استفاده کنید، محدودیت فقط به درج و ویرایش بعدی اعمال خواهد شد(هر دو گزینه فقط با محدودیت‌های CHECK و FOREIGN KEY می‌توانند اعمال شوند).

مثال ۱۹، نشان می‌دهد که چگونه می‌توان تمام محدودیت‌های موجود برای یک جدول را غیرفعال کرد.

مثال ۱۹)

```
USE sample;
ALTER TABLE sales
NOCHECK CONSTRAINT ALL;
```

در این مثال، کلیدواژه‌ی ALL برای غیرفعال کردن تمام محدودیت‌های جدول sales مورد استفاده قرار گرفته است.

نکته

استفاده از گزینه‌ی NOCHECK توصیه نمی‌شود. هر اختلال محدودیتی که متوقف شود ممکن است سبب خرابی ویرایش‌های بعدی شود.

تغییر نام شیء‌های پایگاه داده

روال سیستمی sp_rename نام یک جدول موجود (و هر شیء پایگاه داده مثل پایگاه داده، دیدگاه یا روال ذخیره شده) را تغییر می‌دهد. مثال‌های ۲۰ و ۲۱ استفاده از این روال سیستمی را نشان می‌دهند.

مثال ۲۰)

```
USE sample;
EXEC sp_rename @objname = department, @newname = subdivision
```

این مثال، جدول department را به subdivision تغییر نام می‌دهد.

مثال ۲۱)

```
USE sample;
EXEC sp_rename@objname ='sales.order_no', @newname =ordernumber
```

این مثال، ستون order_no در جدول sales را تغییر نام می‌دهد. اگر شیء تغییر نام یافته، یک ستون در جدول است، مشخصات باید به شکل table_name.column_name باشند.

نکته

از روال سیستمی sp_rename استفاده نکنید، زیرا تغییرنام شیء می‌تواند سبب بروز خطا در سایر شیء‌های پایگاه داده (که به آن رجوع می‌کنند)،شود.شیء را حذف و دوباره با نام جدید ایجاد کنید.

۳-۱۰ حذف شیءهای پایگاه داده

تمام دستورات Transact_SQL که برای حذف یک شیء پایگاه داده مورد استفاده قرار می‌گیرند، دارای شکل کلی زیر هستند:

```
DROP  object_type   object_name
```

هر دستور CREATE object دارای دستور DROP object مربوطه است.

دستور DROP DATABASE database1 {,...} یک یا چند پایگاه داده را حذف می‌کند.به این معنا که تمام اجزای پایگاه داده از سیستم پایگاه داده حذف می‌شوند.

یک یا چند جدول می‌توانند از پایگاه داده با دستور زیر حذف شوند:

```
DROP TABLE  table_name1{,...}
```

تمام داده‌ها، شاخص‌ها و triggerهای مربوط به جدول نیز حذف می‌شوند. فقط کاربری با مجوز مربوطه می‌تواند جدول را حذف کند.

علاوه بر DATABASE و TABLE در دستور DROP به جای object می‌توان موارد زیر را به کار برد:

TYPE

SYNONYM

PROCEDURE

INDEX

VIEW

TRIGGER

SCHEMA

٤-١٠ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شـده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی آن را درک کند.

واژه	معنی	واژه	معنی
master	اصلی - مسلط	initialize	مقداردهی اولیه
back up	پشتیبان گیری	metedata	دادگان
create	ایجاد کردن	rest	بقیه - ادامه
modify	اصلاح کردن - ویرایش	empty	خالی
drop	حذف کردن - رها کردن	except	به جز -استثنای
statement	عبارت	internal	داخلی - درونی
autocommit	پذیرش انجام تمام دستورات به صورت خودکار	statement	عبارت
management	مدیریت	specify	تعیین‌کردن- مشخص نمودن
allow	ارائه کردن	instance	نمونه
explicit	صریح - واضح - آشکار	owner	مالک
implicit	مجازی - مطلق - بی شرط	perform	اجرا کردن
store	ذخیره کردن	activity	فعالیت
implement	پیاده‌سازی	special	خاص - ویژه

The master database should be backed up whenever a user database is created, modified, or dropped.

The CREATE DATABASE statement must run in autocommit mode (the default transaction management mode) and is not allowed in an explicit or implicit transaction.

You can use one CREATE DATABASE statement to create a database and the files that store the database. SQL Server implements the CREATE

DATABASE statement by using the following steps:

The SQL Server uses a copy of the model database to initialize the database and its metadata.

A service broker GUID is assigned to the database.

The Database Engine then fills the rest of the database with empty pages, except for pages that have internal data that records how the space is used in the database.

A maximum of ۳۲,۷۶۷ databases can be specified on an instance of SQL Server.

Each database has an owner that can perform special activities in the database. The owner is the user that creates the database. The database owner can be changed by using sp_changedbowner.

زبـان Transact_SQL از چندین دسـتور تعریف داده‌ها که شـیء‌های پایگاه داده را ایجاد، تغییر و حذف می‌کنند، پشتیبانی می‌کند. شیء‌های پایگاه داده را می‌توان با استفاده از دسـتور CREATE object و DROP object ایجاد و حذف کرد. این شیء‌ها عبارت‌اند از:

- پایگاه داده
- جدول
- شما
- دیدگاه
- Trigger
- روال‌های ذخیره شده
- شاخص

شیء‌های پایگاه داده و جدول، در این واحدکار مورد بررسی قرار گرفتند.

سـاختار تمام شـیء‌های پایگاه داده را که در لیست فوق ذکر شده‌اند می‌توان با استفاده از دسـتور ALTER object تغییر داد. توجه داشـته باشید که دستور ALTER TABLE تنها دسـتور استاندارد شده از این لیست است. تمام دستورات ALTER object از الحاقات Transact-SQL به استاندارد SQL هستند.

برای تایپ دستورات SQL، روی دکمه‌ی New Query در پنجره‌ی Management Studio کلیک کنید.هم چنین، بعد از وارد کردن دسـتورات، روی دکمه Execute کلیک کنید.

خودآزمایی

۱- با استفاده از دستور CREATE DATABASE، یک پایگاه داده‌ی جدید به نام test_db ایجاد کنید. فایل پایگاه داده را با نام منطقی test_db_dat ،که در فایل C:\tmp\test_db.mdf و اندازه‌ی اولیه‌ی ۵ مگابایت، حداکثر اندازه‌ی نامحدود و رشد ۸ درصدی، ایجاد کنید. فایل ثبت را با نام test_db_log در فایل C:\tmp\test_db_log.ldf و اندازه‌ی اولیه‌ی ۲ مگابایت، حداکثر اندازه‌ی ۱۰ مگابایت و رشد ۵۰۰ کیلوبایتی ایجاد کنید.

۲- با استفاده از ALTER DATABASE، یک فایل ثبت جدید به پایگاه داده‌ی test_db اضافه کنید. فایل ثبت را در C:\tmp\emp_Log.ldf و اندازه‌ی اولیه‌ی ۲مگابایت را با رشد ۲مگابایتی و حداکثر اندازه‌ی نامحدود ایجاد کنید.

۳- با استفاده از دستور ALTER DATABASE، اندازه‌ی فایل پایگاه داده‌ی test_db را به ۱۰مگابایت تغییر دهید.

۴- در مثال ۳، بعضی از ستون‌های چهار جدول ایجاد شده با مشخصه‌ی NOT NULL تعریف شده است. برای کدام ستون، این مشخصه نیاز است و برای کدام ستون نیاز نیست؟

۵- چرا ستون‌های dept_no و project_no در مثال ۳ به صورت مقادیر CHAR تعریف شده‌اند (و مقادیر عددی نیستند)؟

۶- جدول‌های customers و orders را با ستون‌های زیر ایجاد کنید. (کلیدهای اصلی و خارجی را تعریف نکنید.)

orders	customers
ordered integer not null	not null (۵)customerid char
not null (۵)customerid char	not null (۴)companyname varchar
orderdate date null	null (۳۰)contactname char
shippeddate date null	null (۶۰)address varchar
freight money null	null (۱۵)city char
null (۴۰)shipname varchar	null (۲۴)phone char
shipaddress varchar	null (۲۴)fax char
quantity integer null	

۷- با اســتفاده از دستور ALTER TABLE، ستون جدیدی به نام shipregion را به جــدول orders اضافه کنید. فیلــد باید قابلیت پوچ بودن را داشــته و از نوع اعداد صحیح باشد.

۸- با اســتفاده از دســتور ALTER TABLE، نوع داده‌ی ســتون shipregion را از INTEGER به CHARACTER با طول هشــت، تغییر دهید. فیلد ممکن اســت شــامل مقادیر NULL باشد.

۹- ستون shipregion را حذف کنید.

۱۰- به طور دقیق شــرح دهید که اگر جدول با دســتور DROP TABLE حذف شود، چه اتفاقی رخ می‌دهد؟

۱۱- دوباره جدول‌های customer و orders را همراه با تعریف تمام محدودیت‌های کلید اولیه و خارجی ایجاد کنید.

۱۲- با اســتفاده از SQL Server Management Studio ســعی کنید که ســطر جدیدی را با مقادیر زیر در جدول orders درج کنید:

(۱ ,'windstar', 'ocean' ,۱۰۰,۰ ,( )getdate( ), getdate ,'ordo۱',۱۰)

چرا این کار انجام نمی‌شود؟

۱۳- با اســتفاده از دستور ALTER TABLE، تاریخ و زمان جاری سیستم را به عنوان مقدار پیش‌فرض به ستون orderdate از جدول orders اضافه کنید.

۱۴- با اســتفاده از دستور ALTER TABLE، یک محدودیت جامعیت ایجاد کنید که مقادیر ممکن ستون quantity در جدول orders را به مقادیر بین ۱و۳۰ محدود کند.

۱۵- تمام محدودیت‌های جامعیت برای جدول orders را نمایش دهید.

۱۶- کلید اولیه‌ی جدول customers را حذف کنید. چرا این کار انجام نمی‌شود؟

۱۷- محدودیت جامعیت به نام prim_empl را،که در مثال ۶ تعریف شده است،حذف کنید.

۱۸- ســتون city از جدول customers را تغییر نام دهیــد. نام جدید را town قرار دهید.

پرس‌وجوها

هدف های رفتاری

پس از پایان این فصل، هنرجو قادر خواهد بود:

- شکل پایه ی دستور SELECT را شرح دهد و به کار گیرد؛
- پرس‌وجوهای فرعی را ایجاد کند و بنویسد؛
- سایر عبارات و توابع دستور SELECT را به کار گیرد؛
- جدول‌های موقتی را ایجاد کند؛
- عملگرهای Join را به کار گیرد؛
- پرس‌وجوهای فرعی وابسته را ایجاد کند؛
- پرس وجوهای فرعی وابسته را بنویسد؛
- عبارات جدولی را به کار گیرد.

۱-۱۱ ورود داده‌ها به جدول

پس از ایجاد جدول‌های بانک اطلاعاتی، باید بتوان داده‌ها را در آن ها وارد نمود.هم‌چنین ،داده‌هــا را ویرایــش و احتمالاً داده‌های غیر ضروری را حذف کــرد. برای ایجاد هماهنگی و سهولت درک مطالب، برای هر کدام از جدول‌های دو پایگاه داده‌ای که از ابتدای کتاب مورد بررسی قرار گرفته‌اند، داده‌هایی را در نظر گرفته‌ایم که آنها را وارد جدول‌ها خواهید کرد.

دو روش برای وارد نمودن داده‌ها در جدول وجود دارد:

اســتفاده از برنامه‌ی Microsoft SQL Server Management Studio که در این جا بیان می‌شود.

استفاده از دستور INSERT که در واحدکار ۱۳ شرح داده می‌شود.

۱-۱-۱۱ ورود داده‌ها با استفاده از SQL Server Management Studio

مراحل زیر، چگونگی ورود داده‌ها به جدول را با ذکر مثال شرح می‌دهند:

۱. SQL Server Management Studi را اجرا کنید.

۲. اتصال به سرور را انجام دهید.

۳. ابتدا مطمئن شوید که بانک اطلاعاتی sample (شرکت پیمانکاری) ایجاد شده باشد. در صورتی که ایجاد نشده بود، بانک اطلاعاتی و جدول‌های مربوطه را ایجاد کنید.

۴. در Object Explorer و بانک اطلاعاتی sample روی جدول Department کلیک راســت نمائیــد و گزینــه‌ی Edit Top ۲۰۰ Rows را انتخاب کنید. اکنــون پنجره‌ی ورود داده‌های مربوط به جدول ظاهر می‌شود (شکل ۱-۱۱).

۵. محتویات جدول را با توجه به داده‌های نمونه زیر، پر کنید (شکل ۲-۱۱).

dept_no	dept_name	Location
d۱	پژوهش	تهران
d۲	حسابداری	کرج
d۳	بازاریابی	تهران

۶. پنجره را ببندید.

شکل(۱-۱۱)

SQORE92.Samp...bo.Department

	dept_no	dept_name	Location
	d1	پژوهش	تهران
	d2	حسابداری	کرج
	d3	بازاریابی	تهران
▶*	NULL	NULL	NULL

شکل (۲-۱۱)

مراحـل ۴ تا ۶ را برای هر کدام از جدول‌ها، با داده‌های نمونه ی زیر، انجام دهید:

جدول Employee

emp_no	emp_fname	emp_lname	dept_no
۲۵۳۴۸	احمد	همت یار	d۲
۱۰۱۰۲	نگین	حمیدی	d۱
۱۸۳۱۶	حسین	علوی	d۳
۲۹۳۴۶	کامران	حیدری	d۲
۹۰۳۱	مهناز	خسروی	d۱
۲۵۸۱	سمیه	شیرازی	d۳
۲۸۵۵۹	علی	جهرمی	d۲

جدول Project

project_no	project_name	Budget
p۱	میل لنگ	۱۲۰۰۰۰۰۰
p۲	سیستم ترمز	۹۵۰۰۰۰۰
p۳	فرمان	۱۸۶۵۰۰۰۰

جدول works_on

emp_no	project_no	job	enter_date
۱۰۱۰۲	p۱	تحلیلگر	۱۳۸۵/۱۰/۱
۱۰۱۰۲	p۳	مدیر	۱۳۸۷/۱/۱
۲۵۳۴۸	p۲	منشی	۱۳۸۶/۲/۱۵
۱۸۳۱۶	p۲	NULL	۱۳۸۶/۶/۱
۲۹۳۴۶	p۲	NULL	۱۳۸۵/۱۲/۱۵
۲۵۸۱	p۳	تحلیلگر	۱۳۸۶/۱۲/۱۵
۹۰۳۱	p۱	مدیر	۱۳۸۶/۴/۱۵
۲۸۵۵۹	p۱	NULL	۱۳۸۷/۸/۱
۲۸۵۵۹	p۲	منشی	۱۳۸۸/۲/۱
۹۰۳۱	p۳	منشی	۱۳۸۵/۱۱/۱۵
۲۹۳۴۶	p۱	منشی	۱۳۸۷/۱/۴

بانک اطلاعاتی بیمارستان و جدول‌های آن را ایجاد کنید و رکوردهای نمونه‌ای را به دل‌خواه در هر جدول وارد کنید.

۲-۱۱ دستور SELECT: شکل پایه و عبارت WHERE

زبان Transact_SQL برای بازیابی اطلاعات از پایگاه داده یک دستور اصلی دارد که عبارت است از دستور SELECT. با این دستور، امکان انجام پرس‌وجوی اطلاعات از یک یا چند جدول پایگاه داده ممکن است (یا حتی از چندین پایگاه داده). حاصل دستور SELECT، جدول دیگری است که result set (مجموعه‌ی حاصل) نامیده می‌شود.

ساده‌ترین شکل دستور SELECT به صورت زیر است:

```
SELECT [ ALL |DISTINCT] column_list
FROM {table2 [tab_alias2] } ,...
```

table2 نام جدولی است که اطلاعات از آن بازیابی می‌شوند. tab_alias1 نام مستعاری را بـرای جدول مربوطه ارائه می‌کند. نام مسـتعار، نام دیگری برای جدول مربوطه اسـت و می‌توانـد میانبری برای رجوع به جدول یا روشـی برای رجوع بـه دو نمونه‌ی منطقی همان جدول فیزیکی باشـد و مورد اسـتفاده قرار گیرد (نگران نباشـید، این دسـتور بعد از انجام مثال‌ها ساده خواهد شد).

column_list شامل یک یا چند مشخصه‌ی زیر است:

علامـت سـتاره(*) ،که تمام سـتون‌های جدول یـا جدول‌های تعیین شـده در عبارت FROM را مشخص می‌کند.

تعیین دقیق نام ستون‌هایی که می‌خواهیم بازیابی شوند.

تعیین column_name [AS] column_heading ،که روشـی برای جای‌گزینی نام ستون یا انتساب یک نام جدید به عبارت است.

- یک عبارت
- یک تابع سیستمی یا تجمعی

دستور SELECT می‌تواند هم ستون‌ها و هم سطرها را از جدول بازیابی کند.

نکته

به عبارت دیگر، دستور select جای گزین دو عملگر زیر است:

۱. گزینش (select) یا محدودیت (restrict): سطرهایی از جدول را ارائه می‌دهد. در این عملگر، تمام ستون‌های جدول ظاهر می‌شوند و سطرها می‌توانند براساس شرطی، انتخاب (فیلتر) شوند.

۲. پرتو (project): ستون‌هایی از جدول را انتخاب می‌کند و برای آن هیچ شرطی اعمال نمی‌شود.

ترکیب هر دو عملیات در دستور SELECT ممکن است.

مثال ۱، ساده‌ترین شکل بازیابی با دستور SELECT را نشان می‌دهد.

مثال ۱) جزئیات کامل تمام بخش‌ها را به دست آورید:

```
USE sample;
SELECT dept_no, dept_name, Location
FROM Department;
```


نتیجه عبارت است از:

	dept_no	dept_name	Location
1	d1	پژوهش	تهران
2	d2	حسابداری	کرج
3	d3	بازاریابی	تهران

دستور SELECT در مثال فوق، تمام سطرها و ستون‌های جدول Department را

بازیابی می‌کند. در صورتی که تمام ستون‌های جدول را در لیست SELECT قرار دهید (مانند مثال ۱)، می‌توانید از علامت * به جای آن استفاده کنید، ولی این روش توصیه نمی‌شود. اسامی ستون‌ها، تیترهای ستون در خروجی نتیجه را ارائه می‌کنند.

مشخصات کامل بیماران بانک اطلاعاتی بیمارستان را نمایش دهید.

مشخصات کارمندان شرکت پیمانکاری را نمایش دهید.

۱-۲-۱۱ عبارت WHERE

ساده‌ترین شکل دستور SELECT، که در بخش قبل شرح داده شد، برای پرس‌وجوها خیلی مفید نیست. در عمل، همیشه چندین عبارت در دستور SELECT وجود دارد که در مثال ۱ نشان داده نشده است. در زیر، شکل کلی دستور SELECT ارائه شده است:

```
SELECT select_list
[INTO new_table_]
FROM table
```

[WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression [ASC | DESC]];

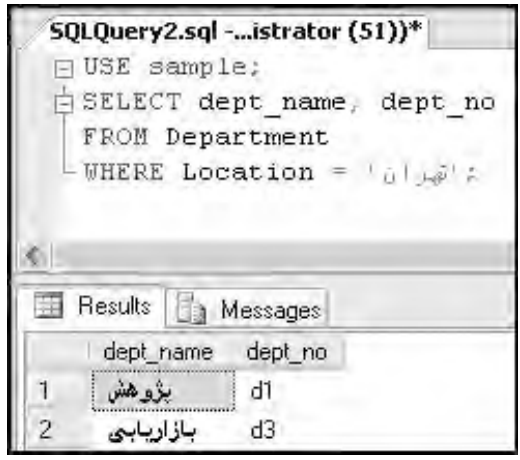
این بخش با تعریف عبارت WHERE شـروع می‌شـود. اغلب ضروری است یک یا چند شرط تعریف کنید که سطرهای انتخاب شده را محدود کنند. عبارت WHERE یک عبارت منطقی را تعیین می‌کند که برای برگرداندن هر سـطری بررسـی می‌شـود. در صورتی که عبارت درست باشد، سطر برگردانده می‌شود.

مثال ۲، استفاده از عبارت WHERE را نشان می‌دهد.

مثال ۲) نام و شماره‌ی تمام بخش‌هایی را که در تهران قرار گرفته‌اند، برمی‌گرداند:

```
USE sample;
SELECT dept_name, dept_no
FROM Department
WHERE Location = تهران ;
```

نتیجه به صورت زیر است:



علاوه بر علامت مســاوی، عبارت WHERE می‌تواند شامل عملگرهای مقایسه‌ای دیگر نیز باشد که عبارت‌اند:

نامساوی	< > (یا =!)
کوچک تر از	<
بزرگ تر از	>
بزرگ تر یا مساوی	>=
کوچک تر یا مساوی	<=

مثال ۳، استفاده از عملگر مقایسه در عبارت WHERE را نشان می‌دهد.

مثال ۳) نام و نام خانوادگی تمام کارمندان با شماره ی کارمندی بزرگ تر یا مساوی ۱۵۰۰۰ را به دست آورید:

```
USE sample;
SELECT emp_lname, emp_fname
FROM Employee
WHERE emp_no >= 15000;
```

نتیجه به صورت زیر خواهد بود:

	emp_lname	emp_fname
1	علوی	حسین
2	همت یار	احمد
3	جهرمی	علی
4	حیدری	کامران

در عبارت WHERE یک گزاره می‌تواند بخشی از شرط باشد که در مثال ۴ نشان داده شده است.

مثال ۴) اســامی پروژه‌هایی با بودجه‌ی بیش از ۱۰۰۰ دلار ارائه کنید. نرخ فعلی تبدیل ریال به دلار، ۰/۰۰۰۱ است.

```
USE sample;
SELECT project_name
FROM project
WHERE budget*0.0001 > 1000;
```

نتیجه به صورت زیر است:

مقایسه ی رشته‌ها(مقادیری از نوع داده‌های CHAR، VARCHAR، NCHAR یا NVARCHAR) با دقت انجام می‌شود. اگر دو رشته با استفاده از کد اسکی (ASCII) (یا هرکد دیگری)مقایسه شوند،تمام کاراکترها (اولی، دومی، سومی و...) با هم مقایسه خواهند شد.

یک کاراکتر در صورتی از کاراکتر دیگر کوچک تر است که در جدول کد، قبل از آن باشد. دو رشته با طول متفاوت بعد از این که رشته ی کوتاه تر در سمت راست با فضای خالی پر شد و دو رشته هم اندازه شدند، مقایسه می‌شوند. اعداد نیز به صورت الفبایی مقایسه می‌شوند. مقادیر انواع داده‌های موقتی (مثل DATE، TIME و DATETIME)،به ترتیب تاریخی و زمانی، مقایسه می‌شوند.

نکته

ستون‌هایی با انواع داده‌های TEXT یا IMAGE نمی‌توانند در عبارت WHERE مورد استفاده قرار گیرند(به جز با عملگرهای LIKE و IS NULL).

تمرین:

مشخصات پزشکانی را نمایش دهید که تخصص آن ها عمومی است.

۲-۲-۱۱ عملگرهای منطقی

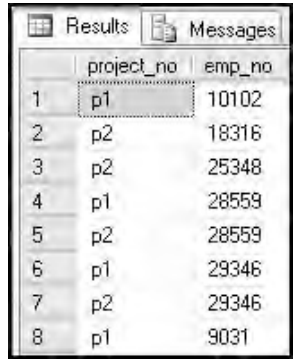
شرط‌های عبارت WHERE می‌توانند ساده باشد یا چندین شرط را شامل شوند. شرط‌های چندگانه می‌توانند با استفاده از عملگرهای AND، OR و NOT ایجاد شوند. رفتار این عملگرها در فصل‌های قبلی شرح داده شد.

مثال ۵) شماره‌ی کارمندی تمام کارمندانی را، که روی پروژه ی P۱ یا P۲ (یا هر دو) کار می‌کنند، به دست آورید:

```
USE sample;
SELECT project_no, emp_no
FROM works_on
```

```
WHERE project_no = 'p1'
OR project_no = 'p2';
```

نتیجه به صورت جدول زیراست:



Results | Messages

	project_no	emp_no
1	p1	10102
2	p2	18316
3	p2	25348
4	p1	28559
5	p2	28559
6	p1	29346
7	p2	29346
8	p1	9031

نتیجه‌ی مثال ۵ شـــامل چند مقدار تکراری در ســـتون emp_no است. اگر می‌خواهید این اطلاعات اضافی حذف شـــوند، گزینه‌ی DISTINCT به صورت زیر مورد اســـتفاده قرار گیرد:

```
USE sample;
SELECT DISTINCT emp_no
FROM works_on
WHERE project_no = 'p1'
OR project_no = 'p2';
```


در این حالت نتیجه به صورت زیر است:

Results | Messages

	emp_no
1	10102
2	18316
3	25348
4	28559
5	29346
6	9031

نکته

ستون‌هایی با نوع داده‌های TEXT یا IMAGE با گزینه‌ی DISTINCT نمی‌توانند بازیابی شوند.

توجه داشته باشـــید که گزینه‌ی DISTINCT فقط با لیست SELECT می‌تواند مورد اســـتفاده قرار گیرد و این گزینه باید قبل از اســـامی ستون‌ها در لیست آورده شود. بنابراین، مثال ۶ اشتباه است.

مثال ۶) مثالی از یک دستور اشتباه

```
USE sample;
SELECT emp_fname, DISTINCT emp_no
FROM Employee
WHERE emp_lname = 'حیدری';
```

نتیجه به صورت زیر است:

```
Server: Msg 156, Level 15, State 1, Line 1
Incorrect syntax near the keyword 'DISTINCT'.
```

نکته

هنگامی که بیش از یک ستون درلیست SELECT وجود داشته باشد، عبارت DISTINCT سطرهایی را که ترکیب ستون‌های آن‌ها متمایز هستند،نمایش می‌دهد.

تمرین:

مشخصات بیمارهایی را نمایش دهید که جنسیت آن‌ها مرد و تاریخ تولدشان بین ۱۳۵۵/۰۱/۰۱ و ۱۳۶۵/۰۱/۰۱ است.

عبارت WHERE ممکن است شامل هر تعدادی از عملیات منطقی یک‌سان یا متفاوت باشد. در یک عبارت منطقی، عملگرهای منطقی NOT، AND و OR به ترتیب دارای اولویت هستند (NOT بالاترین و OR پایین‌ترین تقدم). اگر به این موضوع توجه نداشته باشید، نتایج عبارت‌های منطقی، دور از انتظار خواهد بود (مثال ۷).

مثال ۷)

```
USE sample;
SELECT emp_no, emp_fname, emp_lname
FROM Employee
```

```
WHERE emp_no = 25348 AND emp_fname = 'احمد'
OR emp_lname = 'همت یار' AND dept_no = 'd1';
SELECT emp_no, emp_fname, emp_lname
FROM Employee
WHERE ((emp_no = 25348 AND emp_fname = 'احمد')
OR emp_lname ='همت یار') AND dept_no = 'd1';
```

نتیجه به صورت زیر خواهد بود:

همان طور که نتایج مثال ۷ نشـــان می‌دهد، دو دستور SELECT دو نتیجه‌ی متفاوت را نشان می‌دهند. در اولین دستور SELECT، ابتدا هر دو عملگر AND و سپس OR ارزیابی می‌شوند. در دستور SELECT دوم، استفاده از پرانتزها سبب می شود اجرای عملیات تغییر کندو ابتدا داخل پرانتز از چپ به راست ارزیابی شود. همان طوری که مشاهده کردید، اولین دستور، یک سطر برمی‌گرداند در حالی که دستور دوم، هیچ سطری را برنمی‌گرداند.

وجـــود چندین عمل منطقی در عبارت WHERE را می‌توان با پرانتزها گروه‌بندی کرده

و به نتیجه‌ی مطلوب رسید. برای مثال، می‌توان مثال ۷ را به صورت زیر اصلاح کرد:

```
USE sample;
SELECT emp_no, emp_fname, emp_lname
FROM Employee
WHERE (emp_no = 25348 AND emp_lname = ' همت یار')
OR (emp_fname = 'احمد' AND dept_no = 'd1');
```

در مثـال ۸، کاربرد عملگر NOT،که نقیض یک عبارت منطقی را بر می‌گرداند، نشـان داده شده است.

مثال ۸) شـماره ی کارمنـدی و نام تمام کارمندانی را که مربوط به بخش d2 نیسـتند به دست آورید:

```
USE sample
SELECT emp_no, emp_lname
FROM Employee
WHERE NOT dept_no = 'd2';
```

نتیجه به صورت زیر است:

Results | Messages

	emp_no	emp_lname
1	10102	حمیدی
2	18316	علوی
3	2581	شیرازی
4	9031	خسروی

در این حالت، عملگر NOT را می‌توان به عملگر مقایسه‌ای < > (نامساوی) تغییر داد.

نکته

در ایــن کتــاب از عملگر < > (به جای = !) برای حفظ اســتاندارد ANSI SQL استفاده می‌شود.

تمرین:

نام و نام خانوادگی پزشــکانی را نمایش دهید که تخصص آن ها عمومی نیست.

۳-۲-۱۱ عملگرهای IN و BETWEEN

یــک عملگر IN امــکان تعیین دو یا چند مقــدار را، که می‌توانند برای جســت‌وجو در پرس‌وجو مورد اســتفاده قرار گیرند، فراهم می‌کند. اگر مقدار ســتون مربوطه یکی از مقادیر تعیین شده در IN باشد، نتیجه‌ی شرط، درست خواهد بود.

مثال ۹) مشــخصات کارمندانی را که شــماره‌ی کارمندی آن ها یکی از مقادیر ۲۸۵۵۹، ۲۹۳۴۶ یا ۲۵۳۴۸ است به دست آورید:

نتیجه به صورت زیر است:

Results | Messages

	emp_no	emp_lname	emp_fname
1	25348	احمد	هنیار
2	28559	علی	جهرمی
3	29346	گامران	حیدری

عملگر IN معادل یک سری از شرط‌هاست که با یک یا چند عملگر OR به هم مرتبط هستند(تعداد عملگرهای OR معادل تعداد مقادیر IN منهای یک است).

عملگر IN می‌تواند به همراه عملگرهای منطقی NOT نیز به کار رود (مثال ۱۰). در این حالت، پرس‌وجو سطرهایی را بازیابی می‌کند که شامل هیچ کدام از مقادیر لیست شده نیست.

مثال ۱۰) تمام ستون‌های مربوط به کارمندی را که شماره‌ی کارمندی وی هیچ کدام از مقادیر ۱۰۱۰۲ و ۹۰۳۱ نیست ارائه کنید:

```
USE sample;
SELECT emp_no, emp_fname, emp_lname, dept_no
FROM Employee
WHERE emp_no NOT IN (10102, 9031);
```

نتیجه به صورت زیر است:

عملگر BETWEEN محدوده‌ای از مقادیر را تعیین می‌کند که در این عملگر، کران پایین و بالای مقادیر، مشخص می‌شوند. مثال ۱۱ را مشاهده کنید.

مثال ۱۱) اسامی و بودجه‌ی تمام پروژه‌هایی را که بودجه‌ی آن‌ها بین ۹۵۰۰۰۰۰ و ۱۲۰۰۰۰۰۰ ریال است ارائه کنید:

```
USE sample;
SELECT project_name, Budget
```

```
FROM Project
WHERE Budget BETWEEN 9500000 AND 12000000;
```

نتیجه عبارت است از:

Results | Messages

	project_name	Budget
1	میل لنگ	12000000.00
2	سیستم ترمز	9500000.00

عملگر BETWEEN تمام مقادیر موجود در محدوده‌ی تعیین شده را جست‌وجو می‌کند. این عملگر به طور منطقی معادل دو مقایسه‌ی خاص است که با عملگر AND ترکیب شده‌اند. مثال ۱۲ معادل مثال ۱۱ است.

مثال ۱۲)

```
USE sample;
SELECT project_name, budget
FROM project
WHERE budget >= 9500000 AND budget <= 12000000;
```

شبیه عملگر BETWEEN، عملگر NOT BETWEEN نیز می‌تواند برای جست‌وجوی مقادیر ستون‌هایی که مقدار آنها در محدوده‌ی تعیین شده نیستند، مورد استفاده قرار گیرد. عملگر BETWEEN می‌تواند به ستون‌هایی با مقادیر کاراکتری و تاریخی نیز اعمال شود.

دو دستور SELECT در مثال ۱۳، پرس‌وجویی را نشان می‌دهد که می‌تواند به دو روش متفاوت ولی یک‌سان نوشته شود.

مثال ۱۳) نام تمام پروژه‌هایی را که بودجه‌ی آن‌ها کم‌تر از ۱۰۰۰۰۰۰۰ ریال و بیش‌تر از ۱۵۰۰۰۰۰۰ ریال است،نمایش دهید:

```
USE sample;
SELECT project_name
FROM Project
WHERE Budget NOT BETWEEN 10000000 AND 15000000;
```

نتیجه بدین صورت است:

با استفاده از عملگرهای مقایسه‌ای، پرس‌وجو به صورت زیر تبدیل می‌شود:

```
USE sample;
SELECT project_name
FROM Project
WHERE Budget < 10000000 OR Budget > 15000000;
```

تمرین:

نام و نام خانوادگی پزشکانی را نمایش دهید که تخصص آن ها یکی از موارد «چشـــم، گوش و اعصاب» است.

٤-۲-۱۱ پرس‌وجوهای دربرگیرنده‌ی مقادیر پوچ

یک مقدار NULL در دستور CREATE TABLE تعیین می‌کند که مقادیر خاصی به نام NULL در ستون ارائه شوند. این مقادیر با تمام مقادیر پایگاه داده متفاوت هستند. عبارت WHERE از دســتور SELECT سبب می‌شود ســطرهایی برگردانده شوند که دارای شرط

هســـتند. تمام مقایسه‌ها با مقادیر NULL، مقدار false را برمی‌گرداند (حتی هنگامی که با NOT بیایند). برای بازیابی ســـطرها با مقادیر NULL در ستون، Transact-SQL شامل ویژگی عملگر IS NULL اســـت. این مشـــخصه در عبارت WHERE از دستور SELECT دارای شکل کلی زیر است:

```
Column IS [NOT] NULL
```

مثال ۱۴، کاربرد عملگر IS NULL را نشان می‌دهد.

مثال ۱۴) شماره‌ی کارمندی و شماره‌ی پروژه‌ی مربوط برای کارمندانی را که شغل آن ها مشخص نیست و روی پروژه‌ی p۲ کار می‌کنند را ارائه دهید:

```
USE sample;
SELECT emp_no, project_no
FROM works_on
WHERE project_no = 'p2'
AND job IS NULL;
```

نتیجه به صورت زیر است:

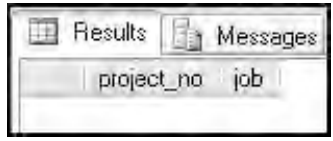
Results | Messages

	emp_no	project_no
1	18316	p2
2	29346	p2

به دلیل این که تمام مقایسه‌ها با مقادیر NULL، مقدار false را برمی‌گرداند، مثال ۱۵ نشـــان می‌دهد که کاربرد NULL به طور تحلیلی صحیح است ولی به طور منطقی نادرست است.

مثال ۱۵)

نتیجه عبارت است از:



شرط Column IS NOT NULL معادل شرط (Column IS NULL) NOT است.

تابع سیستمی ISNULL امکان نمایش مقدار تعیین شده به صورت جای گزین NULL را فراهم می‌کند (مثال ۱۶).

مثال ۱۶)

```
USE sample;
SELECT emp_no, ISNULL(job, 'Job unknown') AS task
FROM works_on
WHERE project_no = 'p1';
```

نتیجه به صورت زیر است:

Results | Messages

	emp_no	task
1	10102	تحلیلگر
2	28559	Job unknow
3	29346	منشی
4	9031	مدیر

مثال ۱۶ از عنوان task برای ستون job استفاده می‌کند.

۵-۲-۱۱ عملگر LIKE

LIKE عملگری است که از آن برای مطابقت الگو استفاده می‌شود. این عملگر، مقدار

ستون را با الگوی تعیین شده مقایسه می‌کند. نوع داده‌ی ستون می‌تواند کاراکتری یا تاریخی باشد. شکل کلی عملگر LIKE به صورت زیر است:

```
Column [NOT] LIKE 'pattern'
```

Pattern ممکن است یک رشته، ثابت یا عبارت تاریخی (شامل ستون‌های جدول) باشد و باید با نوع داده‌ی ستون مربوطه سازگار باشد. اگر مقدار ستون با الگو مطابقت داشته باشد، برای ستون تعیین شده، مقایسه ی بین مقدار موجود در سطر و الگو، درست است.

کاراکترهای جای گزین داخل الگو (کاراکترهای عمومی نامیده می‌شوند) دارای تفسیر خاصی است. دو مورد از این کاراکترها عبارت‌اند:

ش (علامت درصد)- جای گزین هیچ یا چند کاراکتر

_ (زیرخط)- جای گزین یک کاراکتر

مثال ۱۷، کاربرد کاراکترهای عمومی ش و _ را نشان می‌دهد.

مثال ۱۷) نام، نام خانوادگی و شماره‌ی کارمندانی را که نام آن ها شامل حرف «م» به عنوان سومین کاراکتر است نمایش دهید:

```
USE sample;
SELECT emp_fname, emp_lname, emp_no
FROM Employee
WHERE emp_fname LIKE '__م%';
```


نتیجه به صورت زیر است:

Results | Messages

	emp_fname	emp_lname	emp_no
1	احمد	همتیار	25348
2	کامران	حیدری	29346

Transact-SQL ،علاوه بر علامت درصد و زیرخط، از سایر کاراکترهایی که هنگام

اســتفاده از عملگر LIKE دارای معنی خاصی هستند، نیز پشتیبانی می‌کند. این کاراکترها ([]،[و ^) در مثال‌های ۱۸ و ۱۹ شرح داده شده‌اند.

مثال ۱۸) جزئیات کامل تمام بخش‌هایی که محل آن ها با کاراکتری در محدوده‌ی C تا F شروع می‌شود را نمایش دهید:

```
USE sample;
SELECT *
FROM Department
WHERE Location LIKE '[س-ت]%';
```

نتیجه به صورت زیر است:

همان طور که در مثال ۱۸ نشــان داده شده است، زوج کروشه‌ها محدوده‌ای یا لیستی از کاراکترها را در برمی‌گیرند.

نام و نام خانوادگی بیمارهایی را نمایش دهید که حرف دوم نام خانوادگی آن ها ک باشد.

کاراکتــر ^، نقیض محدوده (لیســتی از کاراکترها) را تعییــن می‌کند. این کاراکتر، فقط داخل زوج کروشه‌ها دارای معنی است (شکل ۱۹).

مثال ۱۹) شماره، نام و نام خانوادگی تمام کارمندانی را که نام خانوادگی آن ها با حروف د، ذ، ر، ز، ژ یا س شـــروع نمی‌شـــود و نام آن ها با حرف ج یا ح شـــروع نمی‌شـــود، نمایش دهید:

```
USE sample;
SELECT emp_no, emp_fname, emp_lname
FROM Employee
WHERE emp_lname LIKE '[^س-د]%'
AND emp_fname LIKE '[^ج ح]%';
```

نتیجه به صورت زیر است:

Results | Messages

	emp_no	emp_fname	emp_lname	dept_no
1	18316	حسین	علوی	d3
2	25348	احمد	هنرپیار	d2
3	9031	مهناز	خسروی	d1

شـــرط Column NOT LIKE 'pattern' معـــادل شـــرطNOT (Column LIKE pattern'') است.

مثال ۲۰، کاربرد عملگر LIKE (همراه با NOT) را نشان می‌دهد.

مثال ۲۰) جزئیات کامل تمام کارمندانی را که نام آن ها به کاراکتر ن شـــروع نمی‌شـــود ارائه دهید:


```
USE sample;
SELECT emp_no, emp_fname, emp_lname
FROM Employee
WHERE emp_fname NOT LIKE 'ن%';
```

نتیجه عبارت است از:

Results | Messages

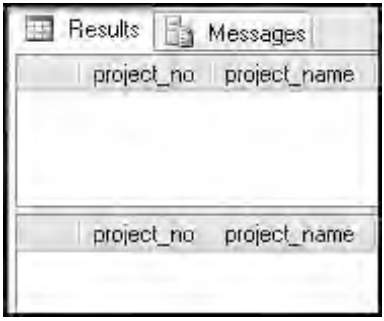
	emp_no	emp_fname	emp_lname
1	18316	حسین	علوی
2	25348	احمد	همتیار
3	2581	سمیه	شیرازی
4	28559	علی	جهرمی
5	29346	کامران	حیدری
6	9031	مهناز	خسروی

هــر کاراکتــر عمومی (د ،_ ،] ، [یا ۸) که داخل زوج کروشــه قــرار گیرد، جای گزینی برای خودش است و کاراکتر عمومی (جانشین) محسوب نمی‌شود. ویژگی معادل این حالت، گزینه‌ی ESCAPE اســت. بنابراین، هر دو دســتور SELECT در مثال ۲۱ معنی یک سانی دارند.

مثال ۲۱)

```
USE sample;
SELECT project_no, project_name
FROM Project
WHERE project_name LIKE '%[_]%';
SELECT project_no, project_name
FROM Project
WHERE project_name LIKE '%!_%' ESCAPE '!';
```

نتیجه به صورت زیر است:



هر دو دستور SELECT، زیرخط را به عنوان کاراکتر واقعی در ستون project_name جست‌وجو می‌کنند. در اولین دستور SELECT، این جست‌وجو با قرار دادن علامت زیرخط در زوج کروشه‌ها انجام می‌شود. دومین دستور SELECT از یک کاراکتر (در مثال ۲۱، کاراکتر !) به عنوان کاراکتر گریز استفاده می‌کند.

نکته

SQL استاندارد فقط از ٪_ و عملگر ESCAPE پشتیبانی می‌کند.

تمرین:

مشخصات پزشک‌هایی را نمایش دهید که حرف اول نام خانوادگی آن ها بین ج تا د نیست.

۳-۱۱ پرس‌وجوهای فرعی

تمام مثال‌های قبلی این واحدکار شـامل مقایسـه‌ی مقادیر ستون با یک عبارت، ثابت یا مجموعه‌ای از ثابت‌ها بودند. زبان Transact-SQL قابلیت مقایسه ی مقادیر ستون با نتایج دسـتور SELECT را ارائه می‌کند. چنین سـاختاری را، که یک یا چند دسـتور SELECT در عبارت WHERE دسـتور SELECT دیگر نوشته شـوند را پرس‌وجوی فرعی می‌نامند. اولین دسـتور SELECT در این نحوه ی نوشتن پرس‌وجوها را پرس‌وجوی بیرونی می‌نامند (در مقایسـه با پرس‌وجوی درونی که دسـتور SELECT آن در حال انجام مقایسـه است). پرس‌وجوی درونی، ابتدا ارزیابی می شـود و پرس‌وجوی بیرونی مقادیر پرس‌وجوی درونی را دریافت و ارزیابی می‌کند.

نکته

یک پرس‌وجوی درونی می‌تواند در داخل خود، دستورات ،INSERT UPDATE یا DELETE داشـته باشـید که در واحدکار بعدی شرح داده می‌شود.

دو نوع پرس‌وجوی فرعی وجود دارد:

- خودشمول(Self-contained)
- وابسته(Correlated)

در یـک پرس‌وجوی فرعی خودشـمول، پرس‌وجوی درونی به طـور منطقی دقیقاً یکبار

ارزیابی می‌شود. یک پرس‌وجوی فرعی وابسته متفاوت از خودشمول است و مقدار آن وابسته به متغیری است که در پرس‌وجوی بیرونی است. بنابراین، پرس‌وجوی درونی یک پرس‌وجوی فرعی وابسته است که به طور منطقی هر بار که سیستم سطر جدیدی را از پرس‌وجوی بیرونی بازیابی می‌کند، ارزیابی می‌شود.

این بخش، مثال‌هایی از پرس‌وجوهای فرعی خودشمول را نشان می‌دهد. پرس‌وجوهای فرعی وابسته در ادامه همراه با عملیات Join در این واحدکار شرح داده می‌شود.یک پرس‌وجوی فرعی خودشمول می‌تواند با عملگرهای زیر مورد استفاده قرار گیرد:

- عملگرهای مقایسه‌ای
- عملگر IN
- عملگر ANY یا ALL

۱-۳-۱۱ پرس‌وجوهای فرعی و عملگرهای مقایسه‌ای

مثال ۲۲، پرس‌وجوی فرعی خودشمولی را نشان می‌دهد که از عملگر = استفاده کرده است.

مثال ۲۲) نام و نام خانوادگی کارمندانی را که در بخش پژوهش کار می‌کنند ارائه کنید:

```
USE sample;
SELECT emp_fname, emp_lname
FROM Employee
WHERE dept_no =
(SELECT dept_no
FROM Department
WHERE dept_name = 'پژوهش');
```

نتیجه به صورت زیر است:

Results | Messages

	emp_fname	emp_lname
1	نگین	حمیدی
2	مهناز	خسروی

پرس‌وجوی درونی مثال ۲۲ به طور منطقی ابتدا ارزیابی می‌شود. پرس‌وجو شماره‌ی بخش تحقیقات (d1) را برمی‌گرداند. بنابراین، بعد از ارزیابی پرس‌وجوی درونی، پرس‌وجوی فرعی مثال ۲۲ می‌تواند با پرس‌وجوی معادل زیر، نمایش داده شود:

```
USE sample
SELECT emp_fname, emp_lname
FROM Employee
WHERE dept_no = ‘d1’;
```

یک پرس‌وجو می‌تواند با سایر عملگرهای مقایسه‌ای نیز مورد استفاده قرار گیرد. هر عملگر مقایسه‌ای می‌تواند مورد استفاده قرار گیرد و پرس‌وجوی درونی یک سطر را برمی‌گرداند. این واضح است، زیرا مقایسه ی بین مقادیر ستون خاص از پرس‌وجوی بیرونی و مجموعه‌ای از مقادیر (نتیجه‌ی پرس‌وجوی درونی)، ممکن نیست. قسمت بعدی نشان می‌دهد که چگونه می‌توان حالتی را،که نتیجه‌ی یک پرس‌وجوی درونی شامل مجموعه‌ای از مقادیر است، مدیریت کرد.

نام و نام خانوادگی پزشکانی را نمایش دهید که حداقل یک روز وقت آزاد دارند.

۲-۳-۱۱ پرس‌وجوهای فرعی و عملگر IN

عملگـر IN امـکان تعیین مجموعـه‌ای از عبارت‌ها (یا ثابت‌هـا) را فراهم می‌کند که به ترتیب برای جست‌وجوی پرس‌وجو مورد استفاده قرار می‌گیرند. این عملگر می‌تواند به یک پرس‌وجوی فرعی نیز به همین دلیل (نتایج پرس‌وجوی درونی شامل مجموعه‌ای از مقادیر)، اعمال شود.

مثال ۲۳، استفاده از عملگر IN را در پرس‌وجوی فرعی نشان می‌دهد.

مثال ۲۳) مشخصات کارمندانی را که بخش آن ها در تهران است ارائه کنید:

```
USE sample;
SELECT *
FROM Employee
WHERE dept_no IN
(SELECT dept_no
FROM Department
WHERE Location = 'تهران');
```

نتیجه به صورت زیر است:

Results | Messages

	emp_no	emp_fname	emp_lname	dept_no
1	10102	نگین	حمیدی	d1
2	18316	حسین	علوی	d3
3	2581	سمیه	شیرازی	d3
4	9031	مهناز	خسروی	d1

هـر پرس‌وجوی درونی ممکن اسـت شـامل پرس‌وجوهای بعدی نیز باشـد. این نوع را پرس‌وجوی فرعی با چندین سـطح تداخل می‌نامند. حداکثر تعداد پرس‌وجوهای درونی در

یک پرس‌وجوی فرعی به میزان حافظه‌ی Database Engine برای هر دستور SELECT بستگی دارد. در وضعیت پرس‌وجوهای فرعی با چندین سطح تودرتو، ابتدا سیستم، داخلی‌ترین پرس‌وجو را اجرا می کند و نتایج را به پرس‌وجوهای سطح بالاتر برمی‌گرداند و الی آخر. در پایان، بیرونی‌ترین پرس‌وجو اجرا می شود و خروجی نهایی ارائه می‌گردد.

نام و نام خانوادگی بیمارهایی را نمایش دهید که حداقل یک بار ویزیت شده‌اند.

مثال ۲۴، پرس‌وجویی با چندین سطح تودرتو را نشان می‌دهد.

مثال ۲۴) نام خانوادگی تمام کارمندانی را که روی پروژه‌ی فرمان کار می‌کنند به‌دست آورید:

```
USE sample;
SELECT emp_lname
FROM Employee
WHERE emp_no IN
(SELECT emp_no
FROM works_on
WHERE project_no IN
(SELECT project_no
FROM Project
WHERE project_name = 'فرمان'));
```

نتیجه به صورت زیر است:

درونی‌ترین پرس‌وجو در مثال ۲۴، مقدار p۳ را برای project_no بدست می‌آورد. پرس‌وجوی درونی وسطی، این مقدار را با تمام مقادیر ستون project_no در جدول works_on مقایسه می‌کند. نتیجه‌ی این پرس‌وجو، مجموعه‌ای از شماره‌های کارمندی (۱۰۱۰۲، ۲۵۸۱ و ۹۰۳۱) است. در پایان بیرونی‌ترین پرس‌وجو، نام‌خانوادگی مربوط به شماره‌های کارمندی انتخاب شده را نمایش می‌دهد.

نام و نام خانوادگی بیمارهایی را نمایش دهید که فقط به وسیله ی پزشک عمومی ویزیت شده‌اند.

۳-۳-۱۱ پرس‌وجوهای فرعی و عملگرهای ANY و ALL

عملگرهای ANY و ALL همیشه در ترکیب با یکی از عملگرهای مقایسه‌ای مورد استفاده قرار می گیرند. شکل کلی هر دو عملگر، به صورت زیر است:

Column_name operator [ANY | ALL] query

که operator یکی از عملگرهای مقایسه‌ای و query یک پرس‌وجوی درونی است.

عملگـر ANY در صورتی درسـت (True) خواهد بود که نتیجـه‌ی پرس‌وجوی درونی مربوطه شـامل حداقل یک سطر باشد که شرط مقایسه را برآورده کند. کلیدواژه‌ی SOME مترادفی برای ANY است. مثال ۲۵، استفاده از عملگر ANY را نشان می‌دهد.

مثال ۲۵) شـماره‌ی کارمندی، کد پروژه و شغل کارمندانی را که در یک پروژه کمترین وقت را صرف کرده‌اند، نمایش دهید:

```
USE sample;
SELECT DISTINCT emp_no, project_no, job
FROM works_on
WHERE enter_date > ANY
(SELECT enter_date
FROM works_on);
```

نتیجه به صورت زیر است:

Results | Messages

	emp_no	project_no	job
1	10102	p3	مدیر
2	18316	p2	NULL
3	25348	p2	منشی
4	2581	p3	تحلیلگر
5	28559	p1	NULL
6	28559	p2	منشی
7	29346	p1	منشی
8	29346	p2	NULL
9	9031	p1	مدیر
10	9031	p3	منشی

هر مقداری از سـتون enter_date در مثال ۲۵ با تمام مقادیر این ستون مقایسه شده اسـت. برای تمام تاریخ‌های سـتون، به جز قدیمی‌ترین آن ها، مقایسـه ی حداقل یک بار با

True برابر می‌شود. سطری با قدیمی‌ترین تاریخ متعلق به نتیجه نیست، زیرا مقایسه در هر حالتی با True برابر نخواهد بود. به عبارت دیگر، عبارت:

" «enter_date>ANY (SELECT enter_date FROM works_on)

درست است اگر یک یا چند (any) سطر در جدول works_on وجود داشته باشد که مقدار ستون enter_date آن از مقدار enter_date سطر جاری کم‌تر باشد.

اگر ارزیابی ستون جدول در پرس‌وجوی درونی تمام مقادیر آن ستون را برگرداند، عملگر ALL با True برابر خواهد بود.

نکته

از عملگرهای ANY و ALL اســتفاده نکنید. هر پرس‌وجویی که از ANY و ALL اســتفاده می‌کند، با تابع EXISTS بهتر خواهد بود (این تابع در ادامه شــرح داده می‌شود). به علاوه، معنی معنایی عملگر ANY به سادگی می‌تواند با معنی معنایی عملگر ALL اشتباه گرفته شود و برعکس.

تمرین:

نام و نام خانوادگی پزشکانی را نمایش دهید که که کمترین وقت آزاد را دارند.

۴-۱۱ دستور SELECT : سایر عبارات و توابع

قسمت‌های زیر، ادامه‌ی عبارت‌هایی را که می‌توانند در پرس‌وجو به صورت توابع تجمعی و مجموعه ی عملگرها مورد استفاده قرار گیرند، شرح می‌دهند. ابتدا عبارت GROUP BY را شـــرح می‌دهیم و چندین مثال را ذکر می‌کنیم. بعد از آن HAVING و ORDER BY معرفی می‌شـــوند. هم چنین این قسمت، مشخصه‌ی IDENTITY و عملگرهای مجموعه‌ای موجود (EXCEPT، INTERSECT، UNION) را شرح می‌دهد.

۱-۴-۱۱ عبارت GROUP BY

این عبارت، یک یا چند ستون را به منزله گروه تعریف می‌کند،به طوری که تمام سطرهای درون گروه برای آن ســـتون‌ها مقادیر یک ســـانی دارند. مثال ۲۶، کاربرد ســـاده‌ای از عبارت GROUP BY را نشان می‌دهد.

مثال ۲۶) شغل تمام کارمندان را در پروژه‌های مختلف به دست آورید:

```
USE sample;
SELECT job
FROM works_on
GROUP BY job;
```

نتیجه عبارت است از:

در مثال۲۶،عبـارت GROUP BY گروه‌هـای متفاوتـی را بـرای تمـام مقادیر ممکن (هم چنین NULL) ستون Job ایجاد می‌کند.

یک جدول می‌تواند براساس هر ترکیبی از ستون‌ها گروه‌بندی شود. مثال ۲۷، گروه‌بندی سطرهای جدول works_onرا، با استفاده از دو ستون، نشان می‌دهد.

مثال ۲۷) تمام کارمندان را با استفاده از کد پروژه و شغل‌هایشان گروه‌بندی کنید:

```
USE sample;
SELECT project_no, job
FROM works_on
GROUP BY project_no, job;
```

نتیجه عبارت است از:

Results | Messages

	project_no	job
1	p1	NULL
2	p1	تحلیلگر
3	p1	مدیر
4	p1	منشی
5	p2	NULL
6	p2	منشی
7	p3	تحلیلگر
8	p3	مدیر
9	p3	منشی

نتیجه‌ی مثال ۲۷ نشان می‌دهد که ۹ گروه با ترکیب‌های متفاوت از کد پروژه و شغل‌ها وجود دارد. فقط دو گروه است که شامل بیش از یک سطر هستند:

p۲	**منشی**	**۲۸۵۵۹ ،۲۵۳۴۸**
p۲	**Null**	**۲۹۳۴۶ ،۱۸۳۱۶**

ترتیب اسامی ستون‌ها در GROUP BY نیاز نیست که با ترتیب اسامی در لیست SELECT یک سان باشند.

نکته

ستون‌هایی با نوع داده‌ی TEXT (یا IMAGE) نمی‌توانند در عبارت GROUP BY مورد استفاده قرار گیرند.

۲-٤-۱۱ توابع تجمعی

توابع تجمعی، توابعی هستند که برای به دست آوردن مقادیر خلاصه مورد استفاده قرار می‌گیرند. تمام توابع تجمعی می‌توانند به چندین گروه تقسیم شوند:

توابع تجمعی متداول

توابع تجمعی آماری

توابع تجمعی تعریف شده به وسیله‌ی کاربر

توابع تجمعی تحلیلی

نوع اول در قسمت‌های زیر شرح داده می‌شود، ولی شرح بقیه توابع از حوصله ی این کتاب خارج است و علاقه‌مندان می‌توانند به کتاب‌های مرجع SQL Server رجوع کنند.

توابع تجمعی متداول

زبان Transact-SQL از شش تابع تجمعی پشتیبانی می‌کند:

MIN

MAX

SUM

AVG
COUNT
COUNT_BIG

تمام توابع تجمعی دارای یک آرگومان هستند که می‌تواند یک ستون یا عبارت باشد (تنها استثنا شکل دوم توابع COUNT و COUNT_BIG است: COUNT(*) و COUNT_BIG(*)). نتیجه‌ی هر تابع تجمعی، یک مقدار ثابت است که در ستون مجزایی نمایش داده می‌شود.

توابع تجمعی که در لیست SELECT ظاهر می‌شوند، می‌توانند شامل یک عبارت GROUP BY باشند. اگر عبارت GROUP BY در دستور SELECT وجود نداشته باشد و لیست SELECT شامل حداقل یک تابع تجمعی باشد، نام هیچ ستونی در لیست SELECT قرار نمی گیرد. بنابراین، مثال ۲۸ اشتباه است.

مثال ۲۸) (مثالی از یک دستور اشتباه)

```
USE sample;
SELECT emp_lname, MIN(emp_no)
FROM Employee;
```

ستون emp_lname از جدول Employee باید در لیست SELECT مثال فوق، قرار نگیرد، زیرا آرگومانی از یک تابع تجمعی نیست. به عبارت دیگر، تمام اسامی ستون‌هایی که آرگومانی از یک تابع تجمعی نیستند، در صورتی می‌توانند در لیست SELECT ظاهر شوند که برای گروه‌بندی مورد استفاده قرار گیرند.

یکی از کلیدواژه‌های زیر می‌تواند قبل از تابع تجمعی قرار گیرد:

ALL- تعیین می‌کند که تمام مقادیر ستون در نظر گرفته شوند (ALL مقدار پیش فرض است).

DISTINCT- مقادیر تکراری ستون را قبل از اعمال تابع تجمعی، حذف می‌کند.

توابع تجمعی MIN و MAX- این توابع، به ترتیب مقدار کمینه و بیشـــینه‌ی ستون را محاسبه می‌کنند. اگر یک عبارت WHERE وجود دارد، توابع MIN و MAX مقدار کمینه و بیشینه را از بین سطرهای انتخاب شده برمی‌گردانند. مثال ۲۹، کاربرد تابع تجمعی MIN را نشان می‌دهد.

مثال ۲۹) کوچک ترین شماره ی کارمندی را به دست آورید:

```
USE sample;
SELECT MIN(emp_no) AS min_employee_no
FROM Employee;
```

نتیجه به صورت زیر است:

	min_employee_no
1	10102

نتیجـــه‌ی مثال ۲۹، کاربرپســـند نیســـت. برای مثال، نـــام کارمندی با حداقل شـــماره، ناشـــناخته اســـت. همان طور که قبلاً نشان داده شـــد، تعیین واضح ستون emp_name در لیســـت SELECT ممکن نیســـت. برای بازیابی نام کارمندی با حداقل شـــماره‌ی کارمندی، از یک پرس‌وجوی فرعی اســـتفاده کنید (مثال ۳۰)،زیرا پرس‌وجوی درونی شـــامل دســـتور SELECT مثال قبل است.

مثـــال ۳۰) شـــماره و نام‌خانوادگی کارمندی را که دارنده ی کم ترین شـــماره کارمندی است، نشان دهید:

```
USE sample;
SELECT emp_no, emp_lname
FROM employee
```

```
WHERE emp_no =
(SELECT MIN(emp_no)
FROM employee);
```

نتیجه عبارت است از:

	emp_no	emp_lname
1	10102	حمیدی

مثال ۳۱، به کارگیری تابع MAX را نشان می‌دهد.

مثال ۳۱) شماره‌ی کارمندی مدیری را که دیرتر از بقیه شروع به کار کرده است را از جدول works_on به دست آورید:

```
USE sample;
SELECT emp_no
FROM works_on
WHERE enter_date =
(SELECT MAX(enter_date)
FROM works_on
WHERE job like  'مد%');
```

نتیجه عبارت است از:

	emp_no	enter_date	job
1	10102	1387-01-01	مدیر

آرگومان توابع MIN و MAX می‌تواند یک مقدار رشته‌ای یا تاریخی باشد.

گزینه‌ی DISTINCT نمی‌تواند با توابع MIN و MAX مورد استفاده قرار گیرد. تمام مقادیر NULL ستونی که آرگومانی از توابع MIN و MAX هستند، همیشه قبل از اجرای تابع، حذف می‌شوند.

تابع تجمعی SUM- این تابع، مجموع مقادیر ستون را محاسبه می‌کند. آرگومان تابع SUM باید عددی باشد. مثال ۳۲، کاربرد این تابع را نشان می‌دهد.

مثال ۳۲) مجموع کل بودجه‌ی تمام پروژه‌ها را محاسبه کنید.

```
USE sample;
SELECT SUM (Budget) sum_of_budgets
FROM Project;
```

نتیجه به صورت زیر است:

	sum_of_budgets
1	40150000.00

تابع تجمعی مثال ۳۲ تمام مقادیر بودجه‌های پروژه‌ها را گروه‌بندی و مجموع کل آن ها را تعیین می‌کند. به همین دلیل، پرس‌وجوی مثال ۳۲ ،به طور غیر آشکار شامل تابع گروه‌بندی است، که آن را به صورت مثال ۳۳ نیز می‌توان نوشت.

مثال ۳۳)

```
SELECT SUM (Budget) sum_of_budgets
FROM project
GROUP BY();
```

استفاده از شکل جدید عبارت GROUP BY توصیه می‌شود، زیرا گروه‌بندی را به طور واضح تعریف می‌کند.

به کارگیری گزینه‌ی DISTINCT تمام مقادیر تکراری ستون را قبل از اجرای تابع SUM، حذف می‌کند. به طور مشابه، تمام مقادیر NULL نیز همیشه قبل از انجام تابع SUM، حذف می‌شوند.

تابع AVG- این تابع، میانگین مقادیر ستون را محاسبه می‌کند. آرگومان تابع AVG

باید عددی باشد. قبل از اجرای تابع AVG، تمام مقادیر NULL حذف می‌شوند. مثال ۳۴، کاربرد این تابع را نشان می‌دهد.

مثال ۳۴) میانگین تمام بودجه‌هایی را که مبلغ بیش از ۱۰۰۰۰۰۰۰ ریال دارند محاسبه کنید.

```
USE sample;
SELECT AVG(Budget) avg_budget
FROM Project
WHERE Budget > 10000000;
```

نتیجه عبارت است از:

	avg_budget
1	15325000.00

توابع تجمعی COUNT و COUNT_BIG

تابع COUNT دارای دو شکل متفاوت است:

```
COUNT ([DISTINCT] col_name)
COUNT (*)
```

اولین شکل،تعداد مقادیر ستونcol_nameرا محاسبه می‌کند.هنگامی که از کلیدواژه‌ی DISTINCT استفاده شود، تمام مقادیر تکراری قبل از اجرای COUNT، حذف می‌شوند. این شکل از تابع، مقادیر NULL ستون را نمی‌شمارد.

مثال ۳۵، کاربرد شکل اول تابع تجمعی COUNT را نشان می‌دهد.

مثال ۳۵) تمام شغل‌های مختلف را در هر پروژه شمارش کنید:

```
USE sample;
SELECT project_no, COUNT(DISTINCT job) job_count
```

```
FROM works_on
GROUP BY project_no;
```

نتیجه به صورت زیر است:

	project_no	job_count
1	p1	3
2	p2	1
3	p3	3

همان طـور که می‌توان از نتیجه‌ی مثال ۳۵ مشـاهده کرد، تمام مقادیر NULL قبل از اجرای تابع (COUNT(DISTINCT job حذف می‌شـوند (مجموع تمام مقادیر ستون به جای ۱۱، عدد ۷ است، زیرا سه سطر دارای مقدار NULL و یک سطر نیز مقدار تکراری دارد).

دومین شکل تابع که تعداد سطرهای جدول را می‌شمارد به صورت COUNT(*) است. یا هم چنین،اگر عبارت WHERE در دسـتور SELECT موجود باشـد، تعداد سـطرهایی برگردانده می‌شـود که شرط WHERE را دارا هستند. در مقایسه با اولین شکل تابع، شکل دوم مقادیـر NULL را حـذف نمی‌کند، زیـرا این تابع روی سـطرها کار می‌کند و نه روی ستون‌ها. مثال ۳۶،کاربرد COUNT(*) را نشان می‌دهد.

مثال ۳۶) تعداد هر شغل را در هر پروژه به دست آورید:

```
USE sample;
SELECT job, COUNT(*) job_count
FROM works_on
GROUP BY job;
```

نتیجه عبارت است از:

Results | Messages

	job
1	NULL
2	تحلیلگر
3	مدیر
4	منشی

تابع COUNT_BIG مشـابه تابع COUNT است. تنها تفاوت بین آن ها در ارتباط با مقادیری است که برمی‌گردانند:

COUNT_BIG همیشه مقداری از نوع داده‌ی BIGINT را برمی‌گرداند، در حالی که تابع COUNT همیشه مقداری از نوع داده‌ی INTEGER را بر می‌گرداند.

میانگین پذیرش بیمار در بیمارستان را نمایش دهید.

۳-۴-۱۱ عبارت HAVING

این عبارت، شرطی را تعریف می‌کند که به گروهی از سطرها اعمال می‌شود. این عبارت همان معنی را برای گروهی از سـطرها دارد که عبارت WHERE برای جدول دارد. شـکل کلی عبارت HAVING به صورت زیر است:

HAVING condition

که condition شامل توابع تجمعی یا ثابت‌هاست.

مثال ۳۷،کاربرد عبارت HAVING با تابع تجمعی COUNT(*) را نشان می‌دهد.

مثال ۳۷) شماره ی پروژه‌هایی را که تعداد کارمندنشان کم تر از چهار نفر است به دست آورید:

```
USE sample;
SELECT project_no
FROM works_on
GROUP BY project_no
HAVING COUNT(*) < 4;
```

نتیجه به صورت زیر است:

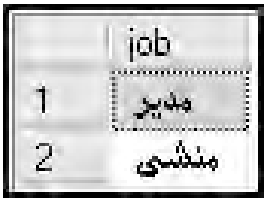
	project_no
1	p3

در مثال ۳۷، سیستم از عبارت GROUP BY برای گروه‌بندی تمام سطرها مطابق با مقادیر موجود در ستون project_no استفاده می‌کند. بعد از آن، تعداد سطرهای هر گروه شمارش شده و گروه‌هایی که سه سطر یا کم تر دارند، انتخاب می‌شوند. عبارت HAVING بدون توابع تجمعی نیز مورد استفاده قرار می گیرد (مثال ۳۸).

مثال ۳۸) سطرهایی از جدول works_on را براساس شغل گروه‌بندی کنید و شغل‌هایی را که با حرف م شروع نمی‌شود حذف کنید:

```
USE sample;
SELECT job
FROM works_on
GROUP BY job
HAVING job LIKE 'م%';
```

نتیجه عبارت است از:



	job
1	مدیر
2	منشی

هم چنیــن عبارت HAVING بدون GROUP BY نیز می‌تواند به کار رود (اگر چه در عمل، متداول نیست). در این حالت، تمام سطرهای جدول یک گروه تلقی می‌شوند.

تمرین:

نام و نام خانوادگی پزشکانی را نمایش دهید که تعداد پذیرش آن ها از میانگین بیمارستان کم تر است.

نکته

ستون‌هایی از نوع داده‌ی TEXT (یا IMAGE) نمی‌توانند با عبارت HAVING مورد استفاده قرار گیرند.

٤-٤-١١ عبارت ORDER BY

این عبارت، ترتیب خاصی از ســطرها را در خروجی پرس‌وجو تعریف می‌کند. این عبارت

دارای شکل کلی زیر است:

ORDER BY {[col_name | col_number [ASC | DESC]]} , ...

col_name نام ستونی است که مقادیر آن مرتب می‌شوند. col_number جای گزین نام ستونی است که موقعیت ستون در لیست SELECT است (یک برای اولین ستون، دو برای دومین ستون،الی آخر). ASC سبب مرتب سازی صعودی و DESC نزولی می‌شود. پیش فرض، ASC است.

نکته

ستون‌های موجود در عبارت OEDER BY نیازی به ظاهر شدن در لیست SELECT ندارند. ولی اگر SELECT DISTINCT تعیین شده باشد، ستون‌های ORDER BY باید در لیست SELECT ظاهر شوند. هم چنین، این عبارت ممکن نیست به ستون‌هایی که جدول آن ها در FROM ذکر نشده، رجوع کند.

همان طور که شکل کلی عبارت ORDER BY نشان می‌دهد، ORDER ممکن است شامل بیش از یک ستون باشد (مثال ۳۹).

مثال ۳۹) شماره‌ی بخش، نام و نام‌خانوادگی کارمندانی را بدست آورید که شماره‌ی کارمندی آن ها از ۲۰۰۰۰ کوچک تر است.سپس آن ها را براساس نام‌خانوادگی و نام،به صورت صعودی،مرتب کنید.

USE sample;

```
SELECT emp_fname, emp_lname, dept_no
FROM employee
WHERE emp_no < 20000
ORDER BY emp_lname, emp_fname;
```

نتیجه به صورت زیر است:

	emp_fname	emp_lname	dept_no
1	نگین	حمیدی	d1
2	مهناز	خسروی	d1
3	سمیه	شیرازی	d3
4	حسین	علوی	d3

هم چنین معرفی ستون‌ها براساس ترتیبی که در لیست SELECT دارند، امکان پذیر است. عبارت ORDER BY مثال ۳۹ را می‌توان به صورت زیر نوشت:

```
ORDER BY 2,1
```

استفاده از شماره ستون‌ها به جای اسامی ستون‌ها،در صورتی که از توابع تجمعی استفاده شود، راه حل مناسبی است. با این وجود، به کارگیری اسامی ستون‌ها(به دلیل حذف و اضافه شدن ستون‌ها به لیست SELECT)توصیه می‌شود. مثال ۴۰، کاربرد شماره ی ستون‌ها را نشان می‌دهد.

مثال ۴۰) برای هر پروژه، کد پروژه و تعداد کارمندان آن را،که براساس شماره‌ی کارمندی به صورت نزولی مرتب شده‌اند، به دست آورید:


```
USE sample;
SELECT project_no, COUNT(*) emp_quantity
FROM works_on
GROUP BY project_no
ORDER BY 2 DESC;
```

نتیجه عبارت است از:

	project_no	emp_quantity
1	p1	4
2	p2	4
3	p3	3

اگر مرتب‌سازی صعودی باشد، مقادیر NULL در بالای تمام مقادیر و اگر نزولی باشد در پایین تمام مقادیر قرار می‌گیرند.

نکته

ستون‌هایی با نوع داده‌ی TEXT (یا IMAGE) در عبارت ORDER BY نمی‌توانند مورد استفاده قرار گیرند.

۵-٤-۱۱ عملگرهای مجموعه‌ای

علاوه بر عملگرهای شـرح داده شده در بخش‌های قبلی، سـه عملگر مجموعه‌ای وجود دارند که زبان Transact-SQL از آن ها پشتیبانی می‌کند:

UNION

ITERSECT

EXCEPT

عملگرهای مجموعه‌ای UNION

نتیجه‌ی اجتماع دو مجموعه، مجموعه‌ای از تمام عناصری است که در هر دو وجود دارند.

بر همین اساس، اجتماع دو جدول، جدول جدیدی است که شامل تمام سطرهای هر دو جدول است.

شکل کلی عملگر UNION به صورت زیر است:

select_1 UNION [ALL] select_2 {[UNION [ALL] select_3]}...

Select_1، Select_2 و... دستورات SELECT هستند که اجتماع را تشکیل می‌دهند. اگر ازگزینه‌ی ALL استفاده شود، تمام سطرهای حاصل (حتی تکرارها) نمایش داده می‌شوند. گزینه‌ی ALL همان معنی را دارد که در لیست SELECT داشته است. فقط یک تفاوت وجود دارد: گزینه‌ی ALL در لیست SELECT پیش فرض است،ولی در UNION باید تعیین شود تا تمام سطرها حتی تکراری را نمایش دهد.

پایگاه داده‌ی sample در شکل اصلی آن برای تشریح عملگر UNION مناسب نیست. بنابراین، در این بخش جدول جدیدی به نام employee_enh معرفی می‌شود که با جدول Employee یک سان و دارای ستون اضافی domicile است. این ستون شامل محل اقامت هر کارمندی است.

جدول جدید employee_enh دارای شکل زیر است:

emp_no	emp_fname	emp_lname	dept_no	domicile
10102	نگین	حمیدی	d1	شهرود
10316	حسین	علوی	d3	هشتگرد
25348	احمد	هنریار	d2	مهرشهر
2581	سعید	شیرازی	d3	رودهن
28559	علی	مهرمی	d2	دماوند
29346	کامران	حیدری	d2	پردیس
9031	مهناز	خسروی	d1	شهریار

برای ایجاد جدول employee_enh از عبارت INTO از دستور SELECT می‌توان استفاده کرد. SELECT INTO دارای دو بخش مختلف است: اولاً یک جدول جدید

با ستون‌های مربوط به ستون‌های لیست شده در لیست SELECT ایجاد می‌کند، ثانیاً سطرهای موجود از جدول اصلی را در جدول جدید درج می‌کند. (نام جدول جدید همراه عبارت INTO و نام جدول اصلی در عبارت FROM از دستور SELECT ظاهر می‌شود).

مثال ۴۱)

```
USE sample;
SELECT emp_no, emp_fname, emp_lname, dept_no
INTO employee_enh
FROM employee;
ALTER TABLE employee_enh
ADD domicile CHAR(25) NULL;
```

در مثال ۴۱، دستور SELECT INTO جدول employee_enh را تولید و تمام سطرهای جدول اولیه (employee) را در آن درج می‌کند. در پایان، دستور ALTER TABLE ستون domicile را به جدول جدید، اضافه می‌کند.

بعد از اجرای مثال ۴۱، ستون domicile شامل هیچ مقداری نیست. مقادیر را می‌توان با استفاده از SQL Server Management Studio (واحدکار هفتم) اضافه کرد:

مثال ۴۲، اجتماع جدول‌های employee_enh و department را نشان می‌دهد.

مثال ۴۲)


```
USE sample;
SELECT domicile
FROM employee_enh
UNION
SELECT Location
FROM Department;
```

نتیجه به صورت زیر است:

	domicile
1	پردیس
2	تهران
3	دماوند
4	رودهن
5	شهرری
6	شهریار
7	مهرشهر
8	هشتگرد
9	کرج

اگر دو جدول با همدیگر سازگار باشند، می‌توانند با عملگر UNION به هم متصل شوند. به این معنی که هر دو لیست SELECT باید تعداد ستون‌های یک سانی داشته باشند و ستون‌های مربوطه باید نوع داده‌ی سازگاری داشته باشند(برای مثال، INT و SMALLINT نوع داده‌های سازگاری هستند).

مرتب‌سازی نتیجه‌ی اجتماع صرفاً در صورتی می‌تواند انجام شود که از عبارت ORDER BY به همراه آخرین دستور SELECT استفاده شود(مثال ۴۳). عبارت های GROUP BY و HAVING می‌توانند با دستورهای SELECT خاصی مورد استفاده قرار گیرند،ولی همراه با اجتماع نیستند.

مثال ۴۳) شماره‌ی کارمندیِ کارمندانی را که در بخش d1 بوده‌اند یا تاریخ شروع به کارشان در پروژه‌ی قبل از ۱۳۸۶/۱/۱ بوده است، به ترتیب صعودی،نمایش دهید.


```
USE sample;
SELECT emp_no
FROM employee
WHERE dept_no = 'd1'
UNION
```

```
SELECT emp_no
FROM works_on
WHERE enter_date < '01.01،1386'
ORDER BY 1;
```

نتیجه به صورت زیر است:

	emp_no
1	10102
2	18316
3	25348
4	2581
5	29346
6	9031

نکته

عملگـر UNION از گزینه‌ی ALL پشـتیبانی می‌کند. UNION به همراه ALL به این معناست که تکرارها از نتیجه حذف نمی‌شوند.

اگـر تمام دسـتورات SELECT،که به وسـیله‌ی یک یا چنـد UNION به هم متصل می‌شوند، به جدول یک سانی رجوع کنند عملگر OR می‌تواند به جای عملگر UNION مورد استفاده قرار گیرد. در این حالت، مجموعه دستورات SELECT با یک دستور SELECT،به همراه مجموعه‌ای از عملگرهای OR، جا به جا می‌شود.

عملگرهای مجموعه‌ای INTERSECT و EXCEPT

دو عملگر مجموعه‌ای دیگر، عبارت‌اند از INTERSECT، که اشتراک را تعیین می‌نماید

و EXCEPT،که عملگر اختلاف (تفاضل) را تعریف می‌کند. اشتراک دو جدول، مجموعه‌ای از سطرهاست که در هر دو جدول وجود دارند. تفاضل دو جدول، مجموعه‌ای از سطرهایی است که در جدول اول هستند،ولی در جدول دوم نیستند. مثال ۴۴، کاربرد عملگر INTERSECT را نشان می‌دهد.

مثال ۴۴) شماره کارمندانی را نمایش دهید که در بخش d۱ کار می‌کنند و تاریخ شروع به کار آن‌ها قبل از تاریخ ۱۳۸۸/۱/۱ بوده است.

```
USE sample;
SELECT emp_no
FROM Employee
WHERE dept_no = 'd1'
INTERSECT
SELECT emp_no
FROM works_on
WHERE enter_date < '01.01.1388';
```

نتیجه عبارت است از:

	emp_no
1	10102
2	9031

نکته

Transact-SQL از عملگر INTERSECT و EXCEPT به همراه ALL پشتیبانی نمی‌کند.

مثال ۴۵، کاربرد عملگر مجموعه‌ای EXCEPT را نشان می‌دهد.

مثال ۴۵) شماره ی کارمندانی را نمایش دهید که در بخش d۳ کار می‌کنند،به جز آن هایی که تاریخ شروع به کارشان بعد از تاریخ ۱۳۸۸/۱/۱ بوده است.

```
USE sample;
SELECT emp_no
FROM Employee
WHERE dept_no = 'd3'
EXCEPT
SELECT emp_no
FROM works_on
WHERE enter_date > '01.01.1388';
```

نتیجه به صورت زیر است:

	emp_no
1	18316
2	2581

نکته

اولویت عملگرهای مجموعه‌ای به ترتیب عبارت‌اند از INTERSECT، EXCEPT و UNION.

فرض کنید که اسامی بازیکنان دو تیم فوتبال و بسکتبال هنرستان در جدول‌هایی با همین نام ذخیره شده است:

Select name From football;

name
احمدی
محمدی
علوی
شمس
کاظمی
کربلایی
صمدزاده

SELECT name FROM basketball;

name
حبیبی
حسن زاده
قربانی
شمس
جاریانی
نظیری
صمدزاده

مطلوب است:

الف) اسامی کل بازیکنان هر دو تیم (با تکرار و بدون تکرار).

ب) اسامی بازیکنانی که در هر دو تیم عضویت دارند.

ج) اسامی بازیکنانی که در تیم فوتبال هستند ولی در تیم بسکتبال نیستند.

۶-٤-۱۱ عبارات CASE

در برنامه‌نویسی کاربردی پایگاه داده، بعضی مواقع تغییر نمایش داده‌ها ضروری است. برای مثال، جنسیت یک شخص می‌تواند با استفاده از مقادیر ۱،۲و۳ (به ترتیب برای زن، مرد و بچه) کدگذاری شود. چنین تکنیک برنامه‌نویسی می‌تواند زمان پیاده‌سازی برنامه را کاهش دهد. عبارت CASE در زبان Transact-SQL چنین کدگذاری ساده‌ای را پیاده‌سازی می‌کند.

عبارت CASE دارای دو شکل مختلف است:

CASE ساده

CASE جست وجو

شکل کلی عبارت CASE ساده به صورت زیر است:

```
CASE expression_1
  {WHEN expression_2 THEN result_1} ...
  [ELSE result_n]
END
```


دستور Transact-SQL به همراه عبارت CASE ساده، اولین عبارت WHEN را که با expression_1 مطابقت دارد پیدا و عبارت THEN آن را اجرا می‌کند. اگر نمونه‌ی منطبقی پیدا نشود، عبارت بعد از EISE اجرا می‌شود.

شکل کلی عبارت CASE جست وجو به صورت زیر است:

```
CASE
  {WHEN condition_1 THEN result_1} ...
  [ELSE result_n]
END
```

دستور Transact-SQL با عبارت CASE جست وجو، اولین عبارتی را که با true برابر باشد جست وجو می‌کند. اگر هیچ کدام از شرط‌های WHEN درست نباشند، مقدار عبارت ELSE برگردانده می‌شود. مثال ۴۶، کاربرد CASE جست وجو را نشان می‌دهد.

مثال ۴۶)

```
USE sample;
SELECT project_name,
CASE
   WHEN Budget > 0 AND Budget < 10000000 THEN 1
   WHEN Budget >= 10000000 AND Budget < 20000000 THEN 2
   WHEN Budget >= 20000000 AND Budget < 30000000 THEN 3
   ELSE 4
END budget_weight
FROM Project;
```

نتیجه عبارت است از:

	project_name	budget_weight
1	میل لنگ	2
2	سیستم ترمز	1
3	فرمان	2

در مثال فوق، بودجه‌ی تمام پروژه‌ها بررسی و محاسبه می‌شود.

مثال ۴۷)

```
USE sample;
SELECT project_name,
  CASE
WHEN p1.Budget < (SELECT AVG(p2.Budget) FROM Project p2)
THEN ‘below average’
WHEN p1.Budget = (SELECT AVG(p2.Budget) FROM Project p2)
THEN ‘on average’
WHEN p1.Budget > (SELECT AVG(p2.Budget) FROM Project p2)
THEN ‘above average’
END budget_category
FROM Project p1;
```

نتیجه عبارت است از:

	project_name	budget_category
1	پل لنگ	below average
2	سیستم ترمز	below average
3	ترمان	above average

پرس‌وجویی بنویسید که نام خانوادگی پزشکان را با این توضیح که تعداد پذیرش روزانه ی آن ها از میانگین بیمارستان کم تر، مساوی یا بیش تر است، نمایش دهد.

۷-٤-۱۱ عبارت COMPUTE

این عبارت از توابع تجمعی (AVG، SUM، MAX و COUNT) برای محاسبه‌ی

مقادیر استفاده می‌کند که در نتیجه‌ی یک پرس‌وجو به صورت سطرهای اضافی ظاهر می‌شوند. توابعی از آن که با عبارت COMPUTE استفاده می‌شود توابع سطری می‌نامند.

توابع تجمعی معمولاً به سطرهای یک جدول اعمال می‌شوند تا یک مقدار را محاسبه کنند.سپس، در نتیجه‌ی پرس‌وجو به صورت یک سطر اضافی ظاهر می‌شود (مثال ۴۸ را مشاهده کنید).

عبارت COMPUTE دارای یک گزینه‌ی اختیاری BY است. BY شکل گروه‌بندی نتیجه را تعریف می‌کند. اگر BY نوشته نشود، تابع به تمام سطرهای پرس‌وجوی حاصله اعمال می‌شود. گزینه‌یBY column_name تعیین می‌کند که مقادیر این ستون برای گروه‌بندی مورد استفاده قرار می‌گیرد. عبارت ORDER BY، در صورت استفاده از BY،مورد نیاز است.

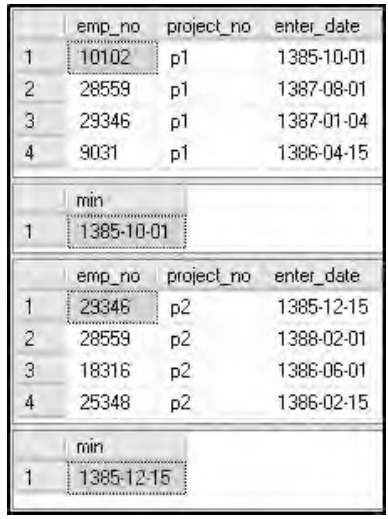
مثال ۴۸، کاربرد عبارت COMPUTE به همراه BY را نشان می‌دهد.

مثال ۴۸)

```
USE sample;
SELECT emp_no, project_no, enter_date
  FROM works_on
  WHERE project_no = ‘p1’ OR project_no = ‘p2’
```

```
ORDER BY project_no
COMPUTE MIN(enter_date) BY project_no;
```

نتیجه عبارت است از:



	emp_no	project_no	enter_date
1	10102	p1	1385-10-01
2	28559	p1	1387-08-01
3	29346	p1	1387-01-04
4	9031	p1	1386-04-15

	min
1	1385-10-01

	emp_no	project_no	enter_date
1	29346	p2	1385-12-15
2	28559	p2	1388-02-01
3	18316	p2	1386-06-01
4	25348	p2	1386-02-15

	min
1	1385-12-15

عبارت COMPUTE می‌تواند در دستور SELECT چندین کاربرد داشته باشد. بنابراین، مثال ۴۸ می‌تواند با استفاده از دستور SELECT و چندین COMPUTE به صورت زیر نوشته شود:

```
COMPUTE MIN(enter_date) BY project_no
COMPUTE MIN(enter_date)
```

چندین محدودیت در ارتباط با عبارت COMPUTE وجود دارد:

SELECT INTO ممکن نیست (زیرا نتیجه‌ی COMPUTE، یک جدول نیست).

تمام ستون‌های عبارت COMPUTE باید در لیست SELECT ظاهر شوند.

نام هر ستون در عبارت COMPUTE BY باید در عبارت ORDER BY ظاهر شود.

ترتیب ستون‌ها در عبارت‌های COMPUTE BY و ORDER BY باید یک سان باشند.

۵-۱۱ جدول‌های موقتی

یک جدول موقتی، شیئی از پایگاه داده است که به طور موقتی ذخیره شده و به وسیله‌ی سیستم پایگاه داده مدیریت می‌شود. جدول‌های موقتی می‌توانند محلی یا عمومی باشند. جدول‌های موقتی محلی دارای نمایش فیزیکی هستند (در پایگاه داده‌ی سیستمی tempdb ذخیره می‌شوند). این نوع جدول‌ها با پیشوند # مشخص می‌شوند (برای مثال، # table_name).

جدول موقتی محلی متعلق به جلسه ی کاری است که در آن ایجاد شده و فقط برای آن جلسه‌ی کاری قابل رؤیت است. چنین جدولی، هنگامی که جلسه‌ی کاری خاتمه یابد، به طور خودکار حذف می‌شود. (اگر جدول موقتی محلی را درون روال ذخیره شده تعریف کنید، هنگام خاتمه‌ی روال، آن نیز از بین خواهد رفت).

جدول‌های موقتی عمومی برای هر کاربری و هر اتصالی بعد از ایجاد، قابل رؤیت است و هنگامی که تمام کاربرانی که به آن ارجاع دارند، از سرور پایگاه داده قطع اتصال کنند، حذف می‌شود. مشابه جدول‌های موقتی محلی، جدول‌های موقتی عمومی نیز با پیشوند # مشخص می‌شوند.

مثال‌های ۴۹ و ۵۰ نشان می‌دهند که چگونه جدول project_temp می‌تواند با استفاده از دو دستور متفاوت Transact-SQL ایجاد شود.

مثال ۴۹)

```
USE sample;
CREATE TABLE #project_temp
(project_no CHAR(۴) NOT NULL,
project_name CHAR(۲۵) NOT NULL);
```

مثال ۵۰)

```
USE sample;
SELECT project_no, project_name
INTO #project_temp
FROM project;
```

مثال‌های فوق شـــبیه هم‌اند. آن ها از دو دســـتور Transact_SQL متفاوت برای ایجاد جـــدول موقتی محلی project_temp # اســـتفاده می‌کنند. مثال ۴۹ جـــدول موقتی را با داده‌های جدول project پر می‌کند،در حالی که مثال ۵۰ آن را خالی رها می‌کند.

۶-۱۱ عملگر Join

بخش‌های قبلی این فصل، کاربرد دستور SELECT را برای پرس‌وجوی سطرها (از جدول پایگاه داده)را شـــرح دادند. اگر زبان Transact-SQL فقط از چنین دســـتورات SELECT ساده پشتیبانی کند، اتصال دو یا چند جدول برای بازیابی داده‌ها غیرممکن خواهد بود.

بر این اســـاس،مجبوریم تمـــام داده‌های پایگاه داده را که در یـــک جدول ذخیره کنیم. اگرچه ذخیره‌ی تمام داده‌های پایگاه داده درون یک جدول، امکان پذیر است، ولی بر کنار از یک عیب عمده نیست (داده‌های ذخیره شده دارای افزونگی فراوان‌اند و نرمال نیستند).

Transact-SQL عملگـــر پیوندی (Join) را ارائه می‌کند کـــه داده‌ها را از چند جدول بازیابی می‌کند. این عملگر احتمالاً برای سیستم‌های پایگاه داده‌ای رابطه‌ای مهم ترین عملگر اســـت، زیـــرا امکان توزیع داده‌ها را روی چندین جدول فراهـــم می‌کند،ضمن این که به یک ویژگی خوب سیستم‌های پایگاه داده دسترسی پیدا می‌کنیم (عدم افزونگی داده‌ها).

نکته

عملگر UNION نیز دو یا چند جدول را به هم متصل می‌کند. با این وجود، عملگر UNION همیشه دو یا چند دستور SELECT را متصل می‌کند، در حالی که عملگر پیوند، دو یا چند جدول را با استفاده از فقط یک SELECT پیوند می‌دهد. عملگر UNION سطرهای جدول‌ها را متصل می‌کند، در حالی که عملگر پیوند، ستون‌های جدول‌ها را نیز پیوند می‌دهد.

عملگر Join به جدول‌های پایه و دیدگاه‌ها اعمال می‌شود. در این واحدکار، پیوند بین جدول‌های پایه شرح داده می‌شود.

عملگر پیوند چندین شکل مختلف دارد. این بخش، انواع اساسی زیر را شرح می‌دهد:

- پیوند طبیعی؛
- ضرب دکارتی؛
- فرا پیوند؛
- پیوند شرطی، خود پیوند و نیم پیوند.

قبل از تشریح شکل‌های مختلف پیوند، این قسمت شکل کلی عملگرهای پیوند را نشان می‌دهد.

۱-۶-۱۱ دو شکل کلی پیاده‌سازی پیوندها

برای پیوند جدول‌ها می‌توان از دو شکل مختلف استفاده کرد:

- شکل پیوندی واضح
- شکل پیوندی تلویحی (پیوند شیوه‌ی قدیمی)

شکل اول پیوند در استاندارد SQL۹۲ معرفی می شود و آشکارا عملیات پیوند را تعریف می‌کند. کلیدواژه‌های مربوط به تعریف واضح پیوند عبارت‌اند از:

- CROSS JOIN
- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN

CROSS JOIN ضرب دکارتی دو جدول را تعیین می‌کند. INNER JOIN پیوند طبیعی دو جدول است، در حالی که LEFT OUTER JOIN و RIGHT OUTER JOIN عملیات پیوند را مطابق با نام خودشان مشخص می‌کنند. در پایان، FULL OUTER JOIN اجتماع راست و چپ فراپیوندها را مشخص می‌کند(تمام این پیوندها در قسمت‌های زیر شرح داده می‌شوند).

شکل کلی پیوند تلویحی، همان شکل کلی «شیوه‌ی قدیمی" است که هر عملی به طور مجازی از طریق عبارت WHERE تعریف می‌شود(مثال ۵۱).

۲-۶-۱۱ پیوند طبیعی

پیوند طبیعی را با ذکر مثال، بهتر می‌توان توضیح داد، بنابراین، مثال ۵۱ را مرور کنید.

مثال ۵۱) مشخصات کامل هر کارمند (شماره ی کارمندی، نام و نام‌خانوادگی) و شماره‌ی بخش، نام بخش و محل آن را برای هر کارمندی که در آن مشغول کار است، نمایش دهید:

```
USE sample;
SELECT Employee.*, Department.*
FROM Employee INNER JOIN Department
ON Employee.dept_no = Department.dept_no;
```

لیست SELECT در مثال فوق، شامل تمام ستون‌های جدول‌های employee و department است. عبارت FROM در دستور SELECT، جدول‌هایی را که با هم پیوند داده می‌شوند تعیین می‌کند. عبارت ON نیز بخشی از عبارت FROM است که ستون‌های پیوند برای هر دو جدول را تعیین می‌کند. شرط Employee.dept_no = Department. dept_no در مثال ۵۱ یک شرط پیوند را مشخص می‌کند.به هر دو ستون ستون‌های پیوندی می‌گویند.

راه حل معادل به صورت زیر است:

شکل پیوندی به شیوه‌ی قدیمی:

```
USE sample;
SELECT Employee.*, Department.*
FROM Employee, Department
WHERE Employee.dept_no = Department.dept_no;
```


شکل پیوندی شیوه‌ی قدیمی دارای دو تفاوت آشکار است: عبارت FROM پرس‌وجو شامل لیستی از جدول‌هاست که پیوند می‌شوند و شرط پیوندی مربوطه، با استفاده از ستون‌های پیوندی در عبارت WHERE، تعیین می‌شود.

نتیجه هر دو پرس‌وجو به صورت زیر است:

	emp_no	emp_fname	emp_lname	dept_no	dept_no	dept_name	Location
1	10102	نگین	حمیدی	d1	d1	پژوهش	تهران
2	18316	حسین	علوی	d3	d3	بازاریابی	تهران
3	25348	احمد	هنتیار	d2	d2	حسابداری	کرج
4	2581	صمیه	شیرازی	d3	d3	بازاریابی	تهران
5	20559	علی	جهرمی	d2	d2	حسابداری	کرج
6	29346	کامران	حیدری	d2	d2	حسابداری	کرج
7	9031	مهناز	خسروی	d1	d1	پژوهش	تهران

نکته

قویاً توصیه می‌کنیم که از علامت * در لیست SELECT استفاده نکنید. فقط هنگامی که می‌خواهید با SQL محاوره داشته باشد، از آن استفاده کنید و در برنامه های کاربردی خود از آن پرهیز نمائید.

مثـال ۵۱ می‌تواند برای نشـان دادن چگونگی عملیات پیوند، مورد اسـتفاده قرار گیرد. توجه داشـته باشـید که این فقط شرحی است برای این که شـما تصویری از فرآیند پیوند داشـته باشـد. موتور پایگاه داده به طور واقعی دارای چندین راه کار است که می‌توان آن را برای پیاده‌سازی عملگر پیوند انتخاب کرد. تصور کنید که هر سطری از جدول Employee با هر سـطری از جدول Department ترکیب می‌شـود. نتیجه‌ی این ترکیب، جدولی با ۷ ستون (۴ ستون از جدول Employee و ۳ ستون از جدول Department) و ۲۱سطر است (جدول ۱ را مشاهده کنید).

	emp_no	emp_lname	emp_lname	dept_no	dept_no	dept_name	Location
1	10102	نگین	حمیدی	d1	d1	پژوهش	تهران
2	18316	حسین	علوی	d3	d1	پژوهش	تهران
3	25348	احمد	هشیار	d2	d1	پژوهش	تهران
4	2581	سمیه	شیرازی	d3	d1	پژوهش	تهران
5	28559	علی	جهرمی	d2	d1	پژوهش	تهران
6	29346	گلاره	حیدری	d2	d1	پژوهش	تهران
7	9031	مهناز	خسروی	d1	d1	پژوهش	تهران
8	10102	نگین	حمیدی	d1	d2	حسابداری	کرج
9	18316	حسین	علوی	d3	d2	حسابداری	کرج
10	25348	احمد	هشیار	d2	d2	حسابداری	کرج
11	2581	سمیه	شیرازی	d3	d2	حسابداری	کرج
12	28559	علی	جهرمی	d2	d2	حسابداری	کرج
13	29346	گلاره	حیدری	d2	d2	حسابداری	کرج
14	9031	مهناز	خسروی	d1	d2	حسابداری	کرج
15	10102	نگین	حمیدی	d1	d3	بازاریابی	تهران
16	18316	حسین	علوی	d3	d3	بازاریابی	تهران
17	25348	احمد	هشیار	d2	d3	بازاریابی	تهران
18	2581	سمیه	شیرازی	d3	d3	بازاریابی	تهران
19	28559	علی	جهرمی	d2	d3	بازاریابی	تهران
20	29346	گلاره	حیدری	d2	d3	بازاریابی	تهران
21	9031	مهناز	خسروی	d1	d3	بازاریابی	تهران

جدول ۱) نتیجه‌ی ضرب دکارتی[1] جدول‌های Employee و Department

در مرحله‌ی دوم،تمام سطرهای جدول۱که شرط پیوند Employee.dept_no=Department.dept_no را برآورده نمی‌کنند، حذف می‌شوند. این سطرها در جدول ۱ با علامت * مشخص شده‌اند. بقیه‌ی سطرها نتیجه‌ی مثال ۵۱ را ارائه می‌کنند.

تحلیل ستون‌های پیوندی مربوطه باید یک سان باشند.به این معنی که هر دو ستون باید دارای معنای منطقی یک سانی باشند. نیاز نیست که ستون‌های پیوندی هم نام باشند (یا حتی از یک نوع باشند)، اگرچه اغلب این حالت رخ می‌دهد.

پایگاه داده‌ی sample شامل سه جفت است از ستون‌هایی که هر کدام از جفت خود معنای منطقی یک سانی دارند(و آن ها دارای اسامی مشابهی نیز هستند).جدول‌های Employee و Department می‌توانند با استفاده از ستون‌های Employee.dept_no و

۱- در ادامه شرح داده می‌شود.

Department.dept_no به هم پیوند داشته باشند.

ستون‌های پیوندی جدول‌های Employee و works_on عبارت‌اند از .Employee emp_no وworks_on.emp_no. در پایان، جدول‌های Project و works_on می‌توانند با استفاده از ستون‌هایProject.project_noوworks_on.project_no به هم پیوند داشته باشند.

اسامی ستون‌ها در دستور SELECT می‌توانند توصیفی باشند. "توصیف کردن" نام ستون به این معناست که برای جلوگیری از هر اشتباه ممکنی درباره‌ی جدولی که ستون به آن متعلق است، می‌توان قبل از نام ستون، نام جدول یا نام مستعار جدول را ذکر کرد که با یک نقطه از هم جدا می‌شوند:

table_name.column_name

در اغلب دستورات SELECT نام ستون نیازی به هیچ توصیفی ندارد.اگرچه استفاده از اسامی توصیفی برای قابلیت خوانایی،توصیه می‌شود.اگر اسامی ستون‌های دستورSELECT مبهم هستند (مثل ستون‌های Employee.dept_no و Department.dept_no در مثال ۵۱)، برای ستون‌ها باید از اسامی توصیفی استفاده شود.

در دستور SELECT به همراه پیوند، عبارت WHERE می‌تواند شرایط دیگری غیر از شرط پیوند داشته باشد،که در مثال ۵۲ نشان داده شده است.

مثال ۵۲) جزئیات کامل تمام کارمندانی را که روی پروژه‌ی فرمان کار می‌کنند ارائه کنید.

پیوند واضح به صورت زیر است:

```
USE sample;
SELECT emp_no, Project.project_no, job, enter_date, project_name,
Budget
```

```
FROM works_on JOIN Project
ON Project.project_no = works_on.project_no
WHERE project_name = 'فرمان';
```

شکل پیوندی قدیمی:

```
USE sample;
SELECT emp_no, Project.project_no, job, enter_date, project_name,
Budget
FROM works_on, Project
WHERE Project.project_no = works_on.project_no
AND project_name = 'فرمان';
```

نتیجه عبارت است از:

	emp_no	project_no	job	enter_date	project_name	Budget
1	10102	p3	مدیر	1387-01-01	فرمان	18650000.00
2	2581	p3	تحلیلگر	1386-12-15	فرمان	18650000.00
3	9031	p3	منشی	1385-11-15	فرمان	18650000.00

از این جا به بعد، تمام مثال‌ها فقط با شکل پیوندی واضح پیاده‌سازی می‌شوند.

مثال ۵۳، کاربرد دیگری از پیوند را نشان می‌دهد.

مثال ۵۳) شـــماره‌ی بخش تمام کارمندانی را که در تاریخ ۱۵ اسفند ماه سال ۱۳۸۶ در پروژه‌هایشان مشغول به کار شده‌اند به دست آورید:


```
USE sample;
SELECT dept_no
FROM employee JOIN works_on
ON employee.emp_no = works_on.emp_no
WHERE enter_date = '12.15.1386';
```

نتیجه به صورت زیر است:

	dept_no
1	d3

نام و نام خانوادگی پزشکانی را نمایش دهید که در تاریخ ۱۳۸۹/۰۱/۱۵ وقت آزاد داشته‌اند.

پیوند بیش از دو جدول

به طور نظری، در تعداد جدول‌هایی که می‌توانند با اســتفاده از دستور SELECT با هم پیوند داشته باشند، محدودیتی وجود ندارد(یک شرط پیوندی، همیشه دو جدول را ترکیب می‌کنــد!) با این وجود، موتور پایگاه داده محدودیت پیاده‌ســازی دارد تعداد جدول‌هایی که می توانند در دستور SELECT با هم پیوند داشته باشند،حداکثر ۶۴ مورد است.

نکته

معمولاً در یک دســتورSELECTحداکثر ۸ تا۱۰جدول پیوند داده می‌شــوند.اگردر یک دســتورSELECT به پیوند بیش از ۱۰جدول نیاز دارید،طراحی پایگاه داده نرمال به نظر نمی رسد.

مثال ۵۴) نام و نام‌خانوادگی تمام تحلیل گرانی را که بخش آن ها در تهران است به دست آورید:

```
USE sample;
SELECT emp_fname, emp_lname
 FROM works_on JOIN Employee ON works_on.emp_no=Employee.
emp_no
  JOIN Department ON Employee.dept_no=Department.dept_no
 AND Location = 'تهران'
 AND job = 'تحلیل‌گر';
```

نتیجه به صورت زیر است:

	emp_fname	emp_lname
1	نگین	حمیدی
2	سمیه	شیرازی

نتیجه‌ی مثال ۵۴ می‌تواند فقط با پیوند سه جدول به دست آید: works_on، Employee و Department. این جدول‌ها با دو جفت ستون پیوندی می‌توانند پیوند داشته باشند:

(works_on.emp_no, Employee.emp_no)

(Employee.dept_no, Department.dept_no)

مثال ۵۵ از ۴ جدول پایگاه داده‌ی sample برای به دست آوردن نتیجه، استفاده می‌کند.

مثال ۵۵) اسامی پروژه‌هایی را که (با حذف افزونگی) به وسیله‌ی کارمندان بخش حسابداری روی آن ها کار شده است ارائه کنید:

```
USE sample;
SELECT DISTINCT project_name
```

```
FROM Project JOIN works_on
ON Project.project_no = works_on.project_no
JOIN Employee ON works_on.emp_no = Employee.emp_no
JOIN Department ON Employee.dept_no = Department.dept_no
WHERE dept_name = 'حسابداری';
```

نتیجه به صورت زیر است:

	project_name
1	سیستم نرم‌افزار
2	میل لنگ

توجه داشته باشید که برای پیوند سه جدول، از دو شرط پیوندی و برای پیوند چهار جدول، از سه شرط پیوندی در پیوند طبیعی استفاده شده است. به طور کلی، اگر n جدول را پیوند دهید، برای جلوگیری از ضرب دکارتی به n-۱ شرط پیوندی نیاز خواهد داشت.

۳-۶-۱۰ ضرب دکارتی

قسمت قبلی، یک روش ممکن از تولید پیوند طبیعی را شرح داد. در اولین مرحله‌ی این فرآیند، هر سطر از جدول Employee با هر سطر از جدول Department ترکیب می‌شود. این نتیجه‌ی واسط به وسیله‌ی عملیاتی به نام ضرب دکارتی ایجاد می‌شود. مثال ۵۶، ضرب دکارتی جدول‌های Employee و Department را نشان می‌دهد.

مثال ۵۶)

```
USE sample;
SELECT Employee.*, Department.*
FROM Employee CROSS JOIN Department;
```

نتیجه‌ی مثال ۵۶ در جدول ۱ نشان داده شده است. ضرب دکارتی، هر سطر از جدول اول را با هر سطر از جدول دوم، ترکیب می‌کند. به طور کلی، ضرب دکارتی دو جدول که

اولی دارای n سطر و دومی دارای m سطر است، جدولی را تولید خواهد کرد که دارای n*m سطر است. بنابراین، حاصل مثال ۵۶ شامل ۷×۳=۲۱ سطر است.

ضرب دکارتی، در عمل خیلی متداول نیست. بعضی مواقع، کاربران ضرب دکارتی دو جدول را،هنگامی که شرط پیوند در عبارت WHERE شیوه‌ی قدیمی را فراموش کنند، تولید می‌نمایند. در این حالت، خروجی منطبق با انتظار نیست، زیرا شامل سطرهای بسیار زیاد خواهد شد.

٤-۶-۱۱ فراپیوند

در مثال‌های قبلی (پیوند طبیعی)، نتیجه شامل فقط سطرهایی از یک جدول است که با سطرهای جدول دیگر مرتبط است. بعضی مواقع ضروری است که علاوه بر سطرهای منطبق، سطرهای غیرمنطبق(از یک یا هر دو جدول)، بازیابی شوند. چنین عملیاتی را فراپیوند می‌نامند.

مثال‌های ۵۷ و ۵۸ تفاوت بین پیوند طبیعی و فراپیوند مربوطه را نشان می‌دهند(تمام مثال‌های این بخش، از جدول employee_enh استفاده می‌کنند).

مثال ۵۷) جزئیات کامل تمام کارمندانی (شامل محل بخش آن ها)را که محل کار و زندگی شان یکی است به دست آورید:

```
USE sample;
SELECT employee_enh.*, Department.Location
FROM employee_enh JOIN Department
ON domicile = Location;
```

نتیجه به صورت زیر است:

	emp_no	emp_fname	emp_lname	dept_no	domicile	Location
1	9031	مهناز	خسروی	d1	تهران	تهران

مثـال ۵۷،بـرای نمایش مجموعه‌ای از سـطرها از پیونـد طبیعی اسـتفاده می‌کند.اگر می‌خواهید محل زندگی سایر کارمندان را بدانید، باید از فراپیوند چپ استفاده کنید. این را فراپیوند چپ می‌نامند، زیرا تمام سطرها از جدول سمت چپ عملگر برگردانده می‌شوند. این سطرها ممکن است در جدول سمت راست سطر منطبق داشته باشند یا نداشته باشند.

به عبارت دیگر، اگر سـطرهای منطبقی در جدول سـمت راست نباشـد، فراپیوند هنوز هم سـطری از جدول سـمت چپ را برمی‌گرداند که مقادیر سـتون‌های جدول دیگر برابر با NULL هسـتند (مثال ۵۷). موتـور پایگاه داده از عملگـر LEFT OUTER JOIN برای تعیین فراپیوند چپ استفاده می‌کند.

فراپیوند راست نیز مشابه همین است، ولی تمام سطرهای جدول سمت راست عملگر را برمی‌گردانـد. موتور پایگاه داده از عملگـر RIGHT OUTER JOIN برای تعیین فراپیوند راست استفاده می‌کند.

مثال ۵۸) جزئیات کامل تمام کارمندان و محل بخش آن ها را برای تمام شـهرهایی که محل زندگی یا هم محل زندگی و هم محل کار کارمندان است ارائه کنید.

```
USE sample;
SELECT employee_enh.*, Department.Location
    FROM employee_enh LEFT OUTER JOIN Department
        ON domicile = Location;
```


نتیجه به صورت زیر است:

	emp_no	emp_fname	emp_lname	dept_no	domicile	location
1	10102	نگین	حمیدی	d1	شهرری	NULL
2	18316	حسین	علوی	d3	هشتگرد	NULL
3	25348	احمد	همتیار	d2	مهرشهر	NULL
4	2581	سمیه	شیرازی	d3	رودهن	NULL
5	28559	علی	جهرمی	d2	دماوند	NULL
6	29346	کامران	حیدری	d2	پردیس	NULL
7	9031	مهناز	خسروی	d1	تهران	تهران

همان طور که می‌توانید مشـــاهده کنید، هنگامی که هیچ سطر منطبقی در جدول سمت راســـت وجود ندارد (در این حالـــت، Department)، فراپیوند چپ هنوز هم ســـطرها را از جدول ســـمت چپ (employee_enh) برمی‌گرداند و ســـتون‌های جدول دیگر با مقادیر NULL مقداردهی می‌شوند. مثال ۵۹، کاربرد عملیات فراپیوند راست را نشان می‌دهد.

مثال ۵۹) جزئیات کامل تمام بخش‌ها به همراه محل زندگی کارمندانشـــان را برای تمام شهرهایی که محل بخش‌ها یا محل کار و زندگی کارمند یکی است به دست آورید.

```
USE sample;
SELECT employee_enh.domicile, Department.*
    FROM employee_enh RIGHT OUTER JOIN Department
        ON domicile =Location;
```

نتیجه به صورت زیر است:

	domicile	dept_no	dept_name	location
1	تهران	d1	پژوهش	تهران
2	NULL	d2	حسابداری	کرج
3	NULL	d3	بازاریابی	تهران

علاوه بر فراپیوندهای چپ و راست، فراپیوند کامل نیز وجود دارد که اجتماع دو فراپیوند چپ و راســـت اســـت. به عبارت دیگر، تمام ســـطرهای هر دو جدول به صـــورت نتیجه ارائه می‌شـــوند. اگر ســـطر مربوطه در یکی از جدول‌هـــا وجود ندارد، ســـتون‌های آن ها با مقادیر NULL برگردانده می‌شـــوند. این عملیات با اســـتفاده از عملگـــر FULL OUTER JOIN مشخص می‌شوند. هر عمل فراپیوندی می‌تواند با استفاده از عملگر UNION،به علاوه تابع NOT EXISTS ،پیاده‌سازی شود. مثال ۶۰ معادل مثال ۵۸ است.

مثال ۶۰)

```
USE sample;
SELECT employee_enh.*, Department.location
   FROM employee_enh JOIN Department
   ON domicile = location
UNION
SELECT employee_enh.*, 'NULL'
  FROM employee_enh
  WHERE NOT EXISTS
  (SELECT *
     FROM Department
      WHERE Location = domicile);
```

اولیــن دســتورSELECT در اجتماع،پیوند طبیعی جدول‌هــای employee_enh و Department را با پیوند ستون‌های domicile و Location تعیین می‌کند. این دستور SELECT تمام شــهرهایی را که به طور هم‌زمان محل زندگی و محل کار هر کارمند است برمی‌گرداند. دستور SELECT دوم، تمام سطرهایی از جدول employee_enh را بازیابی می‌کند که با شرط پیوند طبیعی مطابقت ندارند.

۵-۶-۱۱ سایر فرم‌های عملیات پیوند

قســمت‌های قبلی، مهم‌ترین شکل‌های پیوند را شــرح دادند. این بخش سه شکل دیگر پیوند را نشان می‌دهد:

پیوند شرطی

خود پیوند

نیم پیوند

پیوند شرطی

مقایسه ی ستون‌های پیوندی، با استفاده از علامت مساوی، نیاز نیست. عمل پیوندیِ با استفاده از شرط پیوند کلی (از عملگر مقایسه‌ای غیر از تساوی استفاده می‌کند) را پیوند شرطی می‌نامند. مثال ۶۱، که از جدول employee_enh استفاده می‌کند، عمل پیوند شرطی را نشان می‌دهد.

مثال ۶۱) تمام ترکیبات اطلاعات کارمند و اطلاعات بخش را به دست آورید، به طوری که شهر محل زندگی کارمند از نظر الفبایی از محل بخشی که کارمند در آن کار می‌کند، کوچک تر باشد.

```
USE sample;
SELECT emp_fname, emp_lname, domicile, Location
FROM employee_enh JOIN Department
ON domicile < Location;
```

نتیجه عبارت است از:

	emp_fname	emp_lname	domicile	location
1	علی	جهرمی	دماوند	گرج
2	کامران	حیدری	پردیس	گرج

در مثال ۶۱، مقادیر مربوط به ستون‌های domicile و Location مقایسه می‌شوند. در هر سطر به دست آمده، مقدار ستون domicile قبل از مقدار ستون Location مرتب می‌شود.

خود پیوند یا پیوند یک جدول با خودش

علاوه بر پیوند دو یا چند جدول متفاوت، یک پیوند طبیعی می‌تواند به یک جدول نیز اعمال شود. در این حالت، جدول به خودش پیوند می‌شود که یک ستون از جدول با خودش

مقایسه می‌شود.

مقایسه ی یک ستون با خودش به این معنی است که نام جدول دو بار در عبارت FROM دستور SELECT ظاهر می‌شود. بنابراین،ضروریتی ندارد که به نام یک جدول دو بار رجوع کنید. این کار می‌تواند با استفاده از حداقل یک نام مستعار انجام شود. برای تشخیص هر دو نام ستون در شرط پیوندی، از اسامی توصیفی استفاده کنید. مثال ۶۲، جدول Depatment را به خودش پیوند می‌دهد.

مثال ۶۲) جزئیات کامل بخش‌هایی را که در محلی قرار گرفته‌اند و حداقل یک بخش دیگر در آن جا وجود دارد به دست آورید:

```
USE sample;
SELECT t1.dept_no, t1.dept_name, t1.Location
FROM Department t1 JOIN Department t2
ON t1.Location = t2.Location
WHERE t1.dept_no <> t2.dept_no;
```

نتیجه عبارت است از:

	dept_no	dept_name	Location
1	d3	بازاریابی	تهران
2	d1	پژوهش	تهران

عبارت FROM در مثال ۶۲ شامل دو نام مستعار برای جدول Department است: t1 و t2. اولین شرط مقایسه در مقابل ON پیوند ستون‌ها را تعیین می‌کند در حالی که شرط دوم بعد از WHERE ،تکرارهای غیر ضروری را،که بر اثر مقایسه بخش‌ها به‌وجود آمده‌اند،حذف می‌کند.

نیم پیوند

نیم پیوند شبیه پیوند طبیعی است ولی نتیجه‌ی نیم پیوند فقط سطرهایی از یک جدول است که یک یا چند مطابقت در جدول دوم دارند. مثال ۶۳ ،عمل نیم پیوند را نشان می‌دهد.

مثال ۶۳)

```
USE sample;
SELECT emp_no, emp_lname, e.dept_no
FROM Employee e JOIN Department d
ON e.dept_no = d.dept_no
WHERE Location = 'تهران';
```

نتیجه عبارت است از:

	emp_no	emp_lname	dept_no
1	10102	حمیدی	d1
2	18316	علوی	d3
3	2581	شیرازی	d3
4	9031	خسروی	d1

همان طور که می‌توان در مثال ۶۳ مشاهده کرد، لیست SELECT نیم پیوند شامل فقط ستون‌هایی از جدول Employee است. این دقیقاً چیزی است که عمل نیم پیوند را مشخص می‌کند.از این عمل معمولاً در پردازش پرس‌وجوی توزیعی استفاده می‌شود تا انتقال داده‌ها را به حداقل برساند. موتور پایگاه داده برای پیاده‌سازی ویژه‌ای به نام پیوند ستاره‌ای از عمل نیم پیوند استفاده می‌کند.

مشخصات بیمارهایی را نمایش دهید که در تاریخ ۱۳۸۹/۰۲/۱۵ از پزشک متخصص چشم وقت گرفته‌اند.

۷-۱۱ پرس‌وجوهای فرعی وابسته

پرس‌وجوئی فرعی است که در آن، پرس‌وجوی درونی به پرس‌وجوی بیرونی وابسته باشد.

مثال ۶۴، یک پرس‌وجوی فرعی وابسته را نشان می‌دهد.

مثال ۶۴) نام و نام‌خانوادگی تمام کارمندانی را که روی پروژه‌ی p۳ کار می‌کنند به دست آورید:

```
USE sample;
SELECT emp_lname
  FROM Employee
  WHERE ‘p3’ IN
  (SELECT project_no
     FROM works_on
      WHERE works_on.emp_no = Employee.emp_no);
```


نتیجه عبارت است از:

	emp_lname
1	حمیدی
2	شیرازی
3	خسروی

پرس‌وجوی درونی، در مثال ۶۴، باید به طور منطقی چندین بار ارزیابی شود، زیرا شامل ستون emp_no است که متعلق به جدول Employee در پرس‌وجوی بیرونی است و مقدار ستون emp_no ،هر بار که موتور پایگاه داده سطر متفاوتی از جدول Employee را در پرس‌وجوی بیرونی بررسی می‌کند، تغییر می‌یابد.

بررسی کنید که سیستم، چگونه پرس‌وجوی مثال ۶۴ را پردازش می‌کند؟

ابتدا سیستم، اولین سطر از جدول Employee را بازیابی می‌کند (برای پرس‌وجوی بیرونی) و شماره‌ی کارمندی آن (۱۰۱۰۲) را با مقادیر ستون works_on.emp_no در پرس‌وجوی درونی مقایسه می‌کند. این کارمند دارای دو سطر در جدول works_on با مقادیر p1 و p3 در project_no است، در این صورت، نتیجه‌ی پرس‌وجوی درونی (p3,p1) است. در نتیجه یکی از عناصر معادل با مقدار ثابت p3 است، بنابراین شرط با true برابر می‌شود و مقدار مربوطه از ستون emp_lname در اولین سطر (حمیدی) نمایش داده می‌شود. همین پردازش به تمام سطرهای جدول Employee اعمال شده و نتیجه‌ی نهایی دارای سه سطر است.

اغلب مثال‌های پرس‌وجوهای فرعی وابسته در قسمت بعدی نشان داده شده‌اند.

۱-۷-۱۱ پرس‌وجوهای فرعی و تابع EXISTS

تابع EXISTS یک پرس‌وجوی درونی را به عنوان آرگومان پذیرفته است و در صورتی که پرس‌وجو، یک یا چند سطر را برگرداند، مقدار true و در صورتی که هیچ سطری را برنگرداند،مقدار false را برمی‌گرداند. این تابع، با استفاده از مثال‌های زیر، تشریح خواهد شد:

مثـال ۶۵) نام‌خانوادگی تمـام کارمندانی را که روی پـروژه‌ی p1 کار می‌کنند نمایش دهید:

```
USE sample;
SELECT emp_lname
FROM Employee
WHERE EXISTS
(SELECT *
FROM works_on
WHERE Employee.emp_no = works_on.emp_no
AND project_no = 'p1');
```

نتیجه عبارت است از:

	emp_lname
1	حمیدی
2	جهره‌ی
3	حیدری
4	خسروی

پرس‌وجوی درونی تابع EXISTS ،تقریباً همیشـه به متغیری از یک پرس‌وجوی بیرونی وابسـته اسـت. بنابراین، تابع EXISTS معمولاً یک پرس‌وجوی فرعی وابسـته را مشـخص می‌کند.

بررسی کنید که چگونه پرس‌وجوی مثال ۶۵ به وسیله ی موتور پایگاه داده پردازش می‌شود؟

ابتدا، پرس‌وجوی بیرونی، اولین سطر از جدول Employee را در نظر می‌گیرد (حمیدی). سـپس، تابع EXISTS اجرا می‌شود. برای تعیین این که آیا سطری در جدول works_on وجود دارد که شماره‌ی کارمندی آن مطابق با سطر جاری در پرس‌وجوی بیرونی و project_no

آن برابر p۱است. با استفاده از این فرآیند، تمام سطرهای جدول employee آزمایش می‌شوند و نتیجه نمایش داده می‌شود.

مثال ۶۶ ،کاربرد تابع NOT EXISTS را نشان می‌دهد.

مثال ۶۶) نام‌خانوادگی تمام کارمندانی را نمایش دهید که برای بخش‌هایی کار می‌کنند که در تهران نیستند:

```
USE sample;
SELECT emp_lname
FROM Employee
WHERE NOT EXISTS
(SELECT *
FROM Department
WHERE Employee.dept_no = Department.dept_no
AND Location = 'تهران');
```

نتیجه به صورت زیر است:

	emp_lname
1	همتیار
2	جهرمی
3	حیدری

لیســت SELECT پرس‌وجوی بیرونی، که شامل تابع EXISTS است، نیاز نیست که به شکل SELECT * باشد. شکل SELECT column_list یک شکل جای گزین است. هر دو شــکل معادل هم هستند، زیرا تابع EXISTS فقط بودن (یا نبودن) سطرها در نتیجه را بررسی می‌کند. به همین دلیل، استفاده از SELECT * در این حالت، مطمئن است.

۲-۷-۱۱ دلیل استفاده از پیوند یا پرس‌وجوی فرعی

می‌توان تقریباً تمام دستورات SELECT را که جدول‌ها را پیوند می‌دهند و از عملگر Join استفاده می‌کنند به صورت پرس‌وجوی فرعی دوباره‌نویسی کرد و برعکس. نوشتن دستور SELECT با استفاده از عملگر پیوند، خوانایی و درک آن را ساده‌تر می‌کند و هم‌چنین ،به موتور پایگاه داده کمک می‌کند تا راه کار کارآمدتری برای بازیابی داده‌های مناسب پیدا کند. در عین حال، بعضی از مسائل را می‌توان با استفاده از پرس‌وجوهای فرعی و برخی دیگر را با پیوندها ساده‌تر حل کرد، که تشخیص آن به تجربه و تمرین بیش‌تر نیاز دارد.

مزایای پرس‌وجوی فرعی

پرس‌وجوهای فرعی نسبت به پیوندها دارای مزیت هستند،از جمله هنگامی که برای محاسبه‌ی یک مقدار تجمیع مجبور باشد و از آن‌ها در پرس‌وجوی بیرونی برای مقایسه استفاده شود. مثال ۶۷، این موضوع را نشان می‌دهد.

مثال ۶۷) شماره‌ی کارمندی و تاریخ شروع به کار کارمندانی که تاریخ شروع به کارشان با قدیمی‌ترین تاریخ شروع به کار برابر است نمایش دهید:

```
USE sample
SELECT emp_no, enter_date
 FROM works_on
 WHERE enter_date = (SELECT min(enter_date)
FROM works_on)
```


	emp_no	enter_date
1	10102	1385-10-01

این مسئله را نمی‌توان به سادگی با پیوند حل کرد، زیرا مجبور خواهید بود تا یک تابع تجمعی در عبارت WHERE بنویسید(می‌توان مسئله را با استفاده از دو پرس‌وجوی جداگانه و مرتبط با جدول works_on حل کرد).

مزایای پیوند

اگر لیست SELECT در یک پرس‌وجو شامل ستون‌هایی از چندین جدول باشد، پیوندها نسبت به پرس‌وجوهای فرعی دارای مزیت هستند. مثال ۶۸، این مطلب را نشان می‌دهد.

مثال ۶۸) شماره‌ی کارمندی، نام‌خانوادگی و شغل تمام کارمندانی را که در تاریخ ۱۵ اسفند سال ۱۳۸۶ در پروژه‌هایشان مشغول به کار شده‌اند ارائه کنید:

```
USE sample;
SELECT Employee.emp_no, emp_lname, job
   FROM Employee, works_on
   WHERE Employee.emp_no = works_on.emp_no
    AND enter_date = '12.15.1386';
```

	emp_no	emp_lname	job
1	2581	شیرازی	تحلیلگر

لیست SELECT پرس‌وجوی مثال ۶۸ ،شامل ستون‌های emp_no و emp_lname از جدول Employee و ستون Job از جدول works_on است.به همین دلیل،راه حل معادل با پرس‌وجوهای فرعی، خطایی را نمایش خواهد داد، زیرا پرس‌وجوهای فرعی می‌توانند اطلاعات را فقط از جدول بیرونی نمایش دهند.

برای استفاده از SQL چرا به یادگیری تئوری بانک های اطلاعاتی رابطه‌ای نیاز داریم ؟

SQL بـرای سـرویس دهی بانک‌های اطلاعاتـی رابطـه‌ای ایجـاد شـده اسـت. بدون داشـتن حداقل اطلاعات درباره تئوری بانک اطلاعاتی رابطه ، قادر به استفاده مؤثر نخواهیم بود.

۱۱-۸ عبارات جدولی

عبارات جدولی، پرس‌وجوهای فرعی‌ای هستند که در محلی که انتظار جدولی می‌رود به کار می‌روند. دو نوع عبارت جدولی وجود دارند:

جدول‌های مشتق شده[1]

عبارات جدولی متداول

قسمت‌های زیر، این دو شکل عبارات جدولی را شرح می‌دهند.

۱۱-۸-۱ جدول‌های مشتق شده

جدول مشـتق شـده، یک عبارت جدولی اسـت که در عبارت FROM پرس‌وجو ظاهر می‌شـود. می‌توان جدول‌های مشتق شده را، هنگامی که استفاده از اسامی مستعار ستون‌ها غیرممکن اسـت، اعمال کرد. زیرا قبل از این که اسم مستعار شناسایی شود، عبارت دیگری بوسیله‌ی مترجم SQL پردازش می‌شود. مثال ۶۹ ،این موضوع را نشان می‌دهد.

مثال ۶۹ (**مثالی از یک دستور اشتباه**)

تمام گروه‌های موجود از ماه‌های سـتون enter_date از جدول works_on را نمایش دهید:

```
USE sample;
SELECT MONTH(enter_date) as enter_month
FROM works_on
GROUP BY enter_month;
```


1- Derived Table

نتیجه به صورت زیر است:

```
Msg 207, Level 16, State 1, Line 4
Invalid column name 'enter_month'.
```

دلیل پیغام خطا این اســت که عبارت GROUP BY قبل از لیست SELECT مربوطه پردازش می‌شود و نام مستعار enter_month در زمان گروه‌بندی، ناشناخته است.

با استفاده از یک جدول مشتق شده که شامل یک پرس‌وجوست (بدون عبارت GROUP BY)، می‌توان این مســئله را حل کرد، زیرا عبــارت FROM قبل از عبارت GROUP BY اجرا می‌شود (مثال ۷۰).

مثال ۷۰

```
USE sample;
SELECT enter_month
 FROM (SELECT MONTH(enter_date) as enter_month
FROM works_on) AS m
GROUP BY enter_month;
```

نتیجه به صورت زیر است:

	enter_month
1	1
2	2
3	4
4	6
5	8
6	10
7	11
8	12

به طور کلی، امکان نوشـــتن یک عبارت جدولی در هر جایی از دســـتور SELECT ،که جدول می‌تواند ظاهر شـــود، وجود دارد(نتیجه‌ی عبارت جدولی، همیشـــه یک جدول یا در حالت خاص، یک عبارت خاص است) .مثال ۷۱ ،کاربرد عبارت جدولی در لیست SELECT را نشان می‌دهد.

مثال ۷۱

```
USE sample;
SELECT w.job, (SELECT e.emp_lname
                FROM employee e WHERE e.emp_no = w.emp_no)
AS name
    FROM works_on w
    WHERE w.job IN ('مدیر', 'تحلیل گر');
```

نتیجه عبارت است از:

	job	name
1	تحلیلگر	حمیدی
2	مدیر	حمیدی
3	تحلیلگر	شیرازی
4	مدیر	خسروی

مشخصات پزشکانی را نمایش دهید که در تاریخ ۱۳۸۹/۰۲/۱۵ از پزشک متخصص چشم وقت گرفته‌اند.

۲-۸-۱۱ عبارات جدولی متداول

یک عبارت جدولی متداول[1] (CTE)، عبارت جدولی نام‌گذاری شده‌ای است که به وسیله‌ی Transact-SQL پشتیبانی می‌شود. دو نوع پرس‌وجو وجود دارد که از CTE استفاده می‌کنند:

پرس‌وجوهای غیربازگشتی

پرس‌وجوهای بازگشتی

قسمت‌های زیر، هر دو نوع پرس‌وجو را شرح می‌دهند.

CTEها و پرس‌وجوهای غیربازگشتی

شکل غیربازگشتی یک CTE می‌تواند به عنوان جایگزینی برای جدول‌های مشتق شده و دیدگاه‌ها مورد استفاده قرار گیرد. به طور کلی، یک CTE با استفاده از دستور WITH و یک پرس‌وجوی اضافی تعریف می‌شود که به نام مورد استفاده در WITH رجوع می‌کند (مثال ۷۳).

نکته

کلیدواژه‌ی WITH در زبان Transact-SQL مبهم است. برای پرهیز از ابهام، باید از یک نقطه ویرگول (;) استفاده کنید تا دستور قبل از دستور WITH را خاتمه دهید.

مثال‌های ۷۲ و ۷۳ از پایگاه داده‌ی AdventureWorks استفاده می‌کنند تا نشان دهند که

1- **Common Table Expression**

چگونه می‌توان از CTEها در پرس‌وجوهای غیربازگشتی استفاده کرد. مثال ۷۲ از ویژگی های متداول استفاده می‌کند،درحالی که مثال ۷۳ این مسئله را با استفاده از پرس‌وجوی غیربازگشتی حل می‌کند.

نکته

اگـر پایـگاه داده‌ی AdventureWorks را روی سیسـتم ندارید، می‌توانیـد آن را از سـایت /**www.codeplex.com** **MSFTDBProdsamples** دانلود کنید.

مثال ۷۲)

```
USE AdventureWorks;
SELECT SalesOrderID
FROM Sales.SalesOrderHeader
WHERE TotalDue > (SELECT AVG(TotalDue)
 FROM Sales.SalesOrderHeader
 WHERE YEAR(OrderDate) = '2002')
AND Freight > (SELECT AVG(TotalDue)
FROM Sales.SalesOrderHeader
WHERE YEAR(OrderDate) = '2002')/2.5;
```


پرس‌وجـوی مثـال ۷۲، جمع کل بدهـی افرادی را کـه مبلغ آن ها بیـش از میانگین کل بدهی‌هاسـت وهم چنین افرادی را کـه کرایه ی آن ها بیـش از ۴۰درصد میانگین کل

بدهی‌هاست، به دست می‌آورد.

ویژگی اصلی این پرس‌وجو این اســت که در مصرف حافظــه، صرفه جویی می‌کند، زیرا یک پرس‌وجوی درونی باید دوباره نوشته شود. مثال ۷۳ کاربرد CTE غیربازگشتی را نشان می‌دهد،که تعریف کوتاهی از پرس‌وجوی مثال ۷۲ است.

مثال ۷۳)

```
USE AdventureWorks;
WITH price_calc(year_2002) AS
(SELECT AVG(TotalDue)
        FROM Sales.SalesOrderHeader
        WHERE YEAR(OrderDate) = '2002')
SELECT SalesOrderID
    FROM Sales.SalesOrderHeader
    WHERE TotalDue > (SELECT year_2002 FROM price_calc)
AND Freight > (SELECT year_2002 FROM price_calc)/2.5;
```

شکل کلی عبارت WITH در پرس‌وجوهای غیربازگشتی به صورت زیر است:

```
WITH cte_name (column_list) AS
    ( inner_query)
outer_query
```


cte_name نام CTE اســت که جدول حاصله را مشــخص می‌کند. لیست ستون‌هایی که متعلق به عبارت جدولی هســتند در داخل کروشه‌ها نوشته می‌شوند(CTE در مثال ۷۳، price_calc نامیده شده و دارای ستون year_2002 است).

inner_query در شــکل کلی CTE، دستور SELECT را تعریف می‌کند که نتیجه‌ی عبــارت جدولی مربوطه را مشــخص می‌نماید.بعد از آن،می‌توانیــد از عبارت جدولی تعریف

شـــده در یک پرس‌وجوی بیرونی استفاده کنید(پرس‌وجوی بیرونی در مثال ۷۳ از CTE به نام price_calc و ســـتون year_2002 استفاده می‌کند تا پرس‌وجوی درونی را که دوبار ظاهر می‌شود ساده کند).

CTEها و پرس‌وجوهای بازگشتی

می‌توان از CTEها برای پیاده‌سازی بازگشت‌پذیری استفاده کرد، زیرا ارجاع به خودشان را نیز شـــامل می شـــوند. شـــکل کلی یک CTE برای پرس‌وجوهای بازگشتی به صورت زیر است:

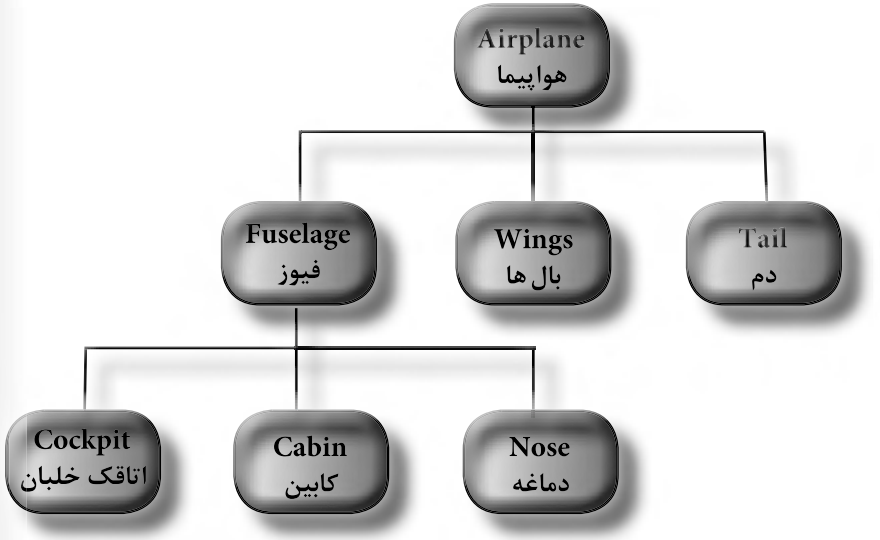
```
WITH cte_name (column_list) AS
   (anchor_member
   UNION ALL
   recursive_member)
outer_query
```

cte_name و column_list دارای همان معنی پرس‌وجوهای غیربازگشـــتی هســـتند. بدنه‌ی عبارت WITH شـــامل دو پرس‌وجویی اســـت که با عملگر UNION ALL به هم متصل هســـتند. اولین پرس‌وجو فقط یک بار اجرا می‌شود و نتیجه‌ی بازگشت را جمع آوری می‌کنـــد. اولین عملوند UNION ALL ارجاعی به CTE ندارد(مثال ۷۴). این پرس‌وجو را پرس‌وجوی لنگر می‌نامند.

دومین پرس‌وجو شـــامل ارجاعی به CTE و ارائه ی بخش بازگشـــتی آن است، به همین دلیل، **عضو بازگشتی** نامیده می‌شود. از جدول تعریف شده در مثال ۷۴ برای تشریح شکل بازگشتی CTEها استفاده خواهد شد.

مثال ۷۴

```
USE sample;
CREATE TABLE airplane
   (containing_assembly VARCHAR(10),
   contained_assembly VARCHAR(10),
   quantity_contained INT,
   unit_cost DECIMAL (6,2));
insert into airplane values ( ‘Airplane’, ‘Fuselage’,1, 10);
insert into airplane values ( ‘Airplane’, ‘Wings’, 1, 11);
insert into airplane values ( ‘Airplane’, ‘Tail’,1, 12);
insert into airplane values ( ‘Fuselage’, ‘Cockpit’, 1, 13);
insert into airplane values ( ‘Fuselage’, ‘Cabin’, 1, 14);
insert into airplane values ( ‘Fuselage’, ‘Nose’,1, 15);
insert into airplane values ( ‘Cockpit’, NULL, 1,13);
insert into airplane values ( ‘Cabin’, NULL, 1, 14);
insert into airplane values ( ‘Nose’, NULL, 1, 15);
insert into airplane values ( ‘Wings’, NULL,2, 11);
insert into airplane values ( ‘Tail’, NULL, 1, 12);
```



شکل ۱) نمایش یک هواپیما و بخش‌های آن

جدول airplane شـامل ۴ ستون است. سـتون containing_assembly یک بخش سرهم شـده را مشخص می‌کند، در حالی که contained_assembly شامل قطعات (یک به یک) اسـت که یک بخش را می‌سازند(شـکل ۱ به طور گرافیکی نشان می‌دهد که چگونه یک هواپیما و قطعات آن می‌توانند در نظر گرفته شوند).

تصور کنید که جدول هواپیما شـامل ۱۱سـطر اسـت که در جدول ۲ نشـان داده شده است(دستورات INSERT[۱] در مثال ۷۴ این سطرها را در جدول airplane درج می‌کنند).

مثـال ۷۵، کاربرد عبارت WITH را برای تعریـف پرس‌وجویی که کل قیمت هر بخش هواپیما را محاسبه می‌کند، نشان می‌دهد.

۱- دستور **Insert** در واحدکار بعدی آموزش داده خواهد شد.

containing_assembly	contained_assembly	quantity_contained	unit_cost
Airplane	Fuselage	1	10.00
Airplane	Wings	1	11.00
Airplane	Tail	1	12.00
Fuselage	Cockpit	1	13.00
Fuselage	Cabin	1	14.00
Fuselage	Nose	1	15.00
Cockpit	*NULL*	1	13.00
Cabin	*NULL*	1	14.00
Nose	*NULL*	1	15.00
Wings	*NULL*	2	11.00
Tail	*NULL*	1	12.00

جدول ۲) محتوای جدول **airplane**

مثال ۷۵)

```
USE sample;
WITH list_of_parts (assembly1, quantity, cost) AS
  (SELECT containing_assembly, quantity_contained, unit_cost
     FROM airplane
     WHERE contained_assembly IS NULL
  UNION ALL
  SELECT a.containing_assembly, a.quantity_contained,
     CAST (l.quantity*l.cost AS DECIMAL(6,2))
     FROM list_of_parts l, airplane a
     WHERE l.assembly1 = a.contained_assembly)
SELECT * FROM list_of_parts;
```


عبـارت WITH، یک CTE به نام List_of_parts را تعریف می‌کند که شـامل سـه ستون اسـت: assembly، quantity و cost. اولین دستور SELECT در مثال ۷۵ فقط

یک بار اجرا می‌شـود تا نتایج اولین مرحله در فرآیند بازگشـتی را جمع آوری کند. دسـتور SELECT در آخرین سطر مثال ۷۵، نتایج زیر را نمایش می‌دهد:

	assembly1	quantity	cost
1	Cockpit	1	13.00
2	Cabin	1	14.00
3	Nose	1	15.00
4	Wings	2	11.00
5	Tail	1	12.00
6	Airplane	1	12.00
7	Airplane	1	22.00
8	Fuselage	1	15.00
9	Airplane	1	15.00
10	Fuselage	1	14.00
11	Airplane	1	14.00
12	Fuselage	1	13.00
13	Airplane	1	13.00

پنج سطر اول در خروجی فوق، نتیجه‌ی اولین اجرای عضو لنگری پرس‌وجوی مثال ۷۵ را نشان می‌دهد. تمام سطرهای دیگر از عضو بازگشتی (بخش دوم) پرس‌وجو در همان مثال، به دست می‌آیند. عضو بازگشتی پرس‌وجو دو بار اجرا خواهد شد: اولین بار برای قطعه‌ی فیوز و بار دوم برای خود هواپیما.

مثـال ۷۶) نام قطعه، تعداد اجـزا و مجموع قیمت هر قطعه ی هواپیما (به همراه اجزای آن) را نمایش دهید.


```
USE sample;
WITH list_of_parts(assembly, quantity, cost) AS
 (SELECT containing_assembly, quantity_contained, unit_cost
    FROM airplane
    WHERE contained_assembly IS NULL
```

```
UNION ALL
SELECT a.containing_assembly, a.quantity_contained,
   CAST(l.quantity*l.cost AS DECIMAL(6,2))
   FROM list_of_parts l,airplane a
  WHERE l.assembly = a.contained_assembly )
SELECT assembly, SUM(quantity) parts, SUM(cost) sum_cost
  FROM list_of_parts
  GROUP BY assembly;
```

خروجی پرس‌وجو به صورت زیر است:

	assembly	parts	sum_cost
1	Airplane	5	76.00
2	Cabin	1	14.00
3	Cockpit	1	13.00
4	Fuselage	3	42.00
5	Nose	1	15.00
6	Tail	1	12.00
7	Wings	2	11.00

چندین محدودیت برای یک CTE در یک پرس‌وجوی بازگشتی وجود دارد:

تعریف CTE باید شامل حداقل دو دستور SELECT باشد (یک عضو لنگری و یک عضو بازگشتی) که با عملگر UNION ALL ترکیب می‌شوند.

تعداد ستون‌ها در اعضای لنگری و بازگشتی باید یک‌سان باشند (این ویژگی عملگر UNION ALL است).

نوع داده‌ی ستون در عضو بازگشتی باید با ستون مربوطه در عضو لنگر، یک‌سان باشد.

عبارت FROM از عضو بازگشتی باید فقط یک‌بار به نام CTE رجوع کند.

گزینه‌های زیر در تعریف بخشی از یک عضو بازگشتی به کار گرفته نشوند:

SELECT DISTINCT، GROUP BY، HAVING، توابع تجمعی، TOP و

پرس‌وجوهای فرعی.

۹-۱۱ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شـــده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی آن را درک کند.

واژه	معنی	واژه	معنی
join	پیوند	operator	عملگر - اپراتور
Base	اصول - پایه - اساس	compare	مقایسه کردن
retrieve	بازیابی کردن	inner join	پیونددرونی-پیوند طبیعی
logical	منطقی	outer join	پیوند بیرونی - فراپیوند
relationship	ارتباط	clause	عبارت
indicate	مشخص کردن - نشان دادن	combine	ترکیب کردن
condition	شرط	reference	رجوع - ارجاع دادن
define	تعریف کردن	separate	جداکردن- مجزا کردن
relate	مرتبط ساختن	recommend	توصیه کردن
specify	تعیین کردن مشخص کردن	method	روش
typical	نمونه	simplify	ساده‌سازی
foreign	خارجی - بیگانه	syntax	نحو - شکل دستور - قاعده
associate	وابسته کردن - وابسته	kind	نوع
cross join	ضرب دکارتی	predicate	گزاره - مبتنی کردن
evaluate	ارزیابی - بررسی	pair	زوج - جفت

Join Fundamentals

By using joins, you can retrieve data from two or more tables based on logical relationships between the tables. Joins indicate how Microsoft SQL Server should use data from one table to select the rows in another

table.

A join condition defines the way two tables are related in a query by:

Specifying the column from each table to be used for the join. A typical join condition specifies a foreign key from one table and its associated key in the other table.

Specifying a logical operator (for example, = or <>,) to be used in comparing values from the columns.

Inner joins can be specified in either the FROM or WHERE clauses. Outer joins can be specified in the FROM clause only. The join conditions combine with the WHERE and HAVING search conditions to control the rows that are selected from the base tables referenced in the FROM clause.

Specifying the join conditions in the FROM clause helps separate them from any other search conditions that may be specified in a WHERE clause, and is the recommended method for specifying joins. A simplified ISO FROM clause join syntax is:

FROM *first_table join_type second_table* [ON (*join_condition*)]

join_type specifies what kind of join is performed: an inner, outer, or cross join. *join_condition* defines the predicate to be evaluated for each pair of joined rows. The following is an example of a FROM clause join specification:


```
FROM Purchasing.ProductVendor JOIN Purchasing.Vendor
    ON (ProductVendor.VendorID = Vendor.VendorID)
```

ایــن فصل تمام ویژگی های دســتور SELECT را درباره‌ی بازیابی داده‌ها(از یک یا چند جدول)پوشــش داد. هر دســتور SELECT که داده‌ها را از یک جدول بازیابی می‌کند، باید شامل حداقل یک لیست SELECT، و عبارت FROM باشد. عبارت FROM تعیین می‌کند که از کدام جدول(ها) داده‌ها بازیابی شود.

مهم ترین عبارت انتخابی، WHERE اســت، شــامل یک یا چند شرط، که با استفاده از عمل گرهای منطقی AND، OR و NOT ترکیب می شــوند. شرط‌ها در عبارت WHERE محدودیتی را روی ســطر انتخاب شده قرار می‌دهند. با دستور SELECT می‌توان ستون‌ها را نیز محدود کرد.

با استفاده از پرس‌وجوها می‌توان عملیات زیر را انجام داد:

۱- داده‌ها را بازیابی کرد.

۲- اطلاعات را به وسیله‌ی چند فیلد خاص مرتب یا فیلتر کرد.

۳- جدول‌هایی ایجاد کرد.

۴- رکوردهای تکراری را یافت و حذف کرد.

۵- عملیات محاســباتی مانند مجموع، میانگین، بزرگ ترین مقدار، کوچک ترین مقدار و غیر آن ها را در جدول انجام داد.

۶- داده‌ها را گروه‌بندی کرد.

بــرای بازیابــی اطلاعات از دو یا چنــد جدول، باید آن ها را با هم پیونــد داد. انواع پیوند

جدول‌ها عبارت اند از:

۱- پیوند طبیعی

۲- فراپیوند

۳- ضرب دکارتی

۴- خودپیوند

۵- نیم‌پیوند

۶- پیوند شرطی

پرس‌وجوهای فرعی مانند پیوند جدول‌ها عمل می‌کنند ولی کارایی الحاق از پرس‌وجوهای فرعی بیش تر است.

عبارت‌های جدولی، پرس‌وجوهای فرعی ای هستند که حاصل آن ها یک جدول است. دو نوع عبارت جدولی وجود دارد که عبارت اند از:

۱- جدول‌های مشتق شده

۲- عبارت‌های جدولی متداول

خودآزمایی

۱- تمام سطرهای جدول works_on را به دست آورید.

۲- شماره کارمندی تمام منشی‌ها را به دست آورید.

۳- شماره‌ی کارمندی کارمندانی را که روی پروژه‌ی p۲ کار می‌کنند و شماره‌ی کارمندی شان کوچک تر از ۱۰۰۰۰ است ارائه دهید. این مسئله را با دستور SELECT حل کنید.

۴- شماره‌ی کارمندی کارمندانی را که در سال ۱۳۸۶ در پروژه‌هایشان شروع به کار نکرده‌اند نمایش دهید.

۵- شماره‌ی کارمندی تمام کارمندانی را که دارای شغل مدیریتی (مثل تحلیل گر یا مدیر) در پروژه‌ی P۱ هستند نمایش دهید.

۶- تاریخ شروع به کار تمام کارمندان پروژه‌ی P۲ را، که شغل آن ها هنوز تعیین نشده است نمایش دهید.

۷- شماره‌ی کارمندی و نام‌خانوادگی تمام کارمندانی را که نامشان شامل دو حرف ت و س است نمایش دهید.

۸- شماره‌ی کارمندی و نام تمام کارمندانی را که دومین حرف نام خانوادگی شان خ یا ش است و به حروف د ، ه ختم می‌شوند نمایش دهید.

۹- شماره‌ی کارمندی تمام کارمندانی را که بخش آن ها در تهران است ارائه کنید.

۱۰- نام‌خانوادگی و نام تمام کارمندانی را که در تاریخ 01/01/1386.در پروژه‌هایشان شروع به کار کرده‌اند نمایش دهید.

۱۱- تمام بخش‌ها را براساس محل‌هایشان گروه‌بندی کنید.

۱۲- تفاوت بین عبارت‌های DISTINCT و GROUP BY چیست؟

۱۳- عبارت GROUP BY چگونه مقادیر NULL را مدیریت می‌کند؟ آیا این مدیریت به رفتار عمومی این مقادیر مربوط است؟

۱۴- تفاوت بین(*) COUNT و column(COUNT) چیست؟

۱۵- بزرگ ترین شماره ی کارمندی را به دست آورید.

۱۶- شغل‌هایی را که به وسیله‌ی بیش از دو کارمند انجام می‌شوند نمایش دهید.

۱۷- شماره ی کارمندی تمام کارمندانی را که منشی هستند یا برای بخش d۳ کار می‌کنند به دست آورید.

۱۸- اشتباه دستور زیر چیست؟

```
SELECT project_name
  FROM project
    WHERE project_no =
  (SELECT project_no FROM works_on WHERE Job = 'منشی')
```

شکل صحیح دستور را بنویسید.

۱۹- کاربرد عملی جدول‌های موقتی چیست؟

۲۰- تفاوت بین جدول‌های موقتی محلی و عمومی چیست؟

تمام راه حل‌های مربوط به تمریناتی را که از عمل پیوند استفاده می‌کنند با استفاده از شکل پیوندی واضح بنویسید.

۲۱- برای جدول‌های Project و works_on، موارد زیر را ایجاد کنید:

الف) پیوند طبیعی

ب) ضرب دکارتی

۲۲- اگر بخواهید چندین جدول را در یک پرس‌وجو پیوند دهید (n جدول)، چند شرط پیوندی مورد نیاز است؟

۲۳- شـــماره ی کارمندی و شغل تمام کارمندانی را که روی پروژه‌ی فرمان کار می‌کنند نمایش دهید.

۲۴- نام و نام‌خانوادگی تمام کارمندانی را که برای بخش‌های پژوهش یا حسابداری کار می‌کنند نمایش دهید.

۲۵- تاریخ شروع به کار تمام کارمندان دفتری (منشی) بخش d۱ را ارائه دهید.

۲۶- اسامی پروژه‌هایی را که دو یا چند کارمند دفتری (منشی) روی آن ها کار می‌کنند نمایش دهید.

۲۷- نـــام و نام‌خانوادگـــی کارمندانی را که مدیرند و روی پروژه‌ی میل لنگ کار می‌کنند نمایش دهید.

۲۸- نام و نام‌خانوادگی تمام کارمندانی را که شـــروع به کار آن ها همزمان با حداقل یک کارمند دیگر است به دست آورید.

۲۹- شـــماره ی کارمندی کارمندانی را که در یک محـــل زندگی می‌کنند و کارمند یک بخش اند نمایش دهید(تذکر: از پایگاه داده‌ی sample توسعه یافته استفاده کنید).

۳۰ شـــماره ی کارمندی تمام کارمندان بخش بازاریابی را به دســـت آورید. دو راه حل با استفاده از موارد زیر ارائه دهید:

عملگر Join

پرس‌وجوی فرعی وابسته

توانایی تغییر داده‌ها

هدف های رفتاری

پس از پایان این فصل، هنرجو قادر خواهد بود:

- دستور INSERT را شرح دهد و با استفاده از آن رکوردهایی را در جدول درج کند؛
- دستور UPDATE را شرح دهد و با استفاده از آن رکوردهای جدول را ویرایش کند؛
- دستور DELETE را شرح دهد و با استفاده از آن رکوردهای جدول را حذف کند؛
- با استفاده از عبارت OUTPUT خروجی رکوردهای تغییر یافته را نمایش دهد.

۱-۱۲ دستور INSERT

قبلاً در این کتاب آموختید که با استفاده از ابزار Microsoft SQL Server Management می‌توان سطرها را در جدول‌ها درج کرد. در این قسمت، چگونگی انجام این کار را با دستور INSERT مشاهده خواهید کرد. این دستور، سطرها (یا بخشی از آن ها) را در جدول درج می‌کند. این دستور دارای دو شکل متفاوت است:

۱) INSERT [INTO] tab_name [(col_list)]
DEFAULT VALUES | VALUES ({ DEFAULT | NULL | *expression* }
[,...*n*])

۲) INSERT INTO tab_name | view_name [(col_list)]
{select_statement | execute_statement}

با استفاده از شکل اول، فقط یک سطر (یا بخشی از آن) در جدول tab_name درج می‌شود. دومین شکل دستور INSERT، نتایج به دست آمده از دستور SELECT یا روال ذخیره شده را،که با دستور EXECUTE اجرا می‌شود درج می‌کند(روال ذخیره شده باید داده‌هایی را برگرداند که در جدول درج می‌شوند. دستور SELECT می‌تواند مقادیر را از جدول متفاوت یا همان جدول مقصد دستور INSERT برگرداند ولی نوع ستون‌ها باید سازگار باشند).

در هر دو شکل، مقدار درج شده در جدول باید دارای نوع داده‌ی سازگار با نوع داده‌ی ستون مربوطه باشد. برای اطمینان از سازگاری، تمام مقادیر کاراکتری و داده‌های زمانی باید در نشانه ی آپستروف‌ها قرار گیرند،در حالی که مقادیر عددی به نشانه ای نیاز به چیزی ندارند.

۱-۱-۱۲ درج یک سطر (رکورد)

در هر دو شکل دستور INSERT، مشخصه ی لیست ستون، اختیاری است.به این معنی

که حذف لیست ستون‌ها معادل تعیین لیست تمام ستون‌ها در جدول است.

گزینه‌ی DEFAULT VALUES مقادیر پیش‌فرض را برای تمام ستون‌ها درج می‌کند. اگر ستونی از نوع داده‌ی TIMESTAMP بوده یا دارای مشخصه‌ی IDENTITY باشد، مقداری که به طور خودکار به وسیله‌ی سیستم ایجاد شده است، درج خواهد شد.

برای سایر انواع داده‌ها، اگر پیش‌فرض موجود باشد، ستون با مقدار پیش‌فرض غیرپوچ مناسبی، مقداردهی می‌شود.در غیر این صورت،مقداردهی با NULLصورت می گیرد. اگر ستون قابلیت داشتن مقادیر پوچ را نداشته باشد و مقدار DEFAULT نیز نداشته باشد، دستور INSERT قطع خواهد شد و خطایی را مشخص خواهد کرد.

مثال‌های ۱ تا ۴ سطرها را در چهار جدول از پایگاه داده‌ی sample درج می‌کند. این عمل، استفاده از دستور INSERT را برای ورود مقادیر کمی از داده‌ها در جدول‌ها نشان می‌دهد.

مثال ۱) داده‌هایی را در جدول Department درج کنید:

```
USE sample;
INSERT INTO Department VALUES ('d4', 'هشتگرد', 'تولید');
INSERT INTO Department VALUES ('d5', 'شهریار', 'انبار');
INSERT INTO Department VALUES ('d6', 'شهرری', 'نقلیه');
```

نتیجه ی اجرای این دستورات سبب خواهد شد که سه رکورد جدید به جدول اضافه شود:

dept_no	dept_name	Location
d1	پژوهش	تهران
d2	حسابداری	کرج
d3	بازاریابی	تهران
d4	تولید	هشتگرد
d5	انبار	شهریار
d6	نقلیه	شهرري

مثال ۲) رکوردهایی را در جدول employee درج کنید:

```
USE sample;
INSERT INTO Employee VALUES (35498, 'قمصری' ,'علیرضا','d6');
INSERT INTO Employee VALUES (100101, 'شمس' ,'زینب','d3');
INSERT INTO Employee VALUES (18475, 'صداقتی' ,'سارا', 'd1');
INSERT INTO Employee VALUES (92436, 'جماعتی' ,'حمید', 'd5');
INSERT INTO Employee VALUES (3091, 'بهروزی' ,'فریبا', 'd4');
INSERT INTO Employee VALUES (1852, 'حسنی' ,'کمیل', 'd6');
INSERT INTO Employee VALUES (98825, 'محب زاده' ,'سیما', 'd5');
```

نتیجه ی اجرای این دستورات، سبب اضافه شدن ۷ رکورد جدید به جدول Employee خواهد شد:

emp_no	emp_fname	emp_lname	dept_no
100101	زینب	شمس	d3
10102	نگین	خلیلی	d1
10316	حسین	علوی	d3
18475	سارا	صداقتی	d1
1852	کمیل	حسنی	d6
25348	احمد	قلمکار	d2
2581	سمیه	شیرازی	d3
28559	علی	جهرمی	d2
29346	کامران	حیدری	d2
3091	فریبا	بهروزی	d4
35498	علیرضا	قمصری	d6
9031	مهناز	خسروی	d1
92436	حمید	جماعتی	d5
98825	سیما	محب زاده	d5

چرا رکوردهای موجود در جدول فوق، به ترتیبی که درج شده‌اند، قرار ندارند؟

مثال ۳) داده‌هایی را در جدول Project درج کنید:

```
USE sample;
Insert into  Project values ('p4', '2000000.00 ,'داشبورد');
Insert into  Project values ('p5', '5000000.00 ,'ضد سرقت');
Insert into  Project values ('p6', '10000000.00 ,'صندلی');
```

project_no	project_name	Budget
p1	میل لنگ	12000000.0000
p2	سیستم ترمز	9500000.0000
p3	فرمان	18650000.0000
p4	داشبورد	2000000.0000
p5	ضد سرقت	5000000.0000
p6	صندلی	10000000.0000

مثال ۴) داده‌هایی را در جدول works_on درج کنید:

```
USE sample;
Insert into works_on values (100101,'p4', '1386.10.1' ,'منشی');
Insert into works_on values (100101, 'p6', '1388.1.1' ,'منشی');
Insert into works_on values (35498, 'p5', '1387.2.15' ,'مدیر');
Insert into works_on values (18475, 'p6', NULL, '1387.6.1');
Insert into works_on values (92436, 'p5', NULL, '1386.12.15');
Insert into works_on values (1852, 'p6', '1387.10.15' ,'تحلیلگر');
Insert into works_on values (3091, 'p4', '1387.4.15' ,'مدیر');
Insert into works_on values (98825, 'p4', 'NULL', '1387.8.1');
Insert into works_on values (98825, 'p5', '1388.2.1' ,'منشی');
Insert into works_on values (3091, 'p6', '1386.11.15' ,'تحلیلگر');
Insert into works_on values (92436, 'p1','1387.1.4' ,'منشی');
```

emp_no	project_no	job	enter_date
100101	p4	منشی	1386-10-01
100101	p6	منشی	1388-01-01
10102	p1	تحلیلگر	1385-10-01
10102	p3	مدیر	1387-01-01
10316	p2	NULL	1386-06-01
18475	p6	NULL	1387-06-01
1852	p6	تحلیلگر	1387-10-15
25348	p2	منشی	1386-02-15
2581	p3	تحلیلگر	1386-12-15
28559	p1	NULL	1387-08-01
28559	p2	منشی	1388-02-01
29346	p1	منشی	1387-01-04
29346	p2	NULL	1385-12-15
3091	p4	مدیر	1387-04-15
3091	p6	تحلیلگر	1386-11-15
35490	p5	مدیر	1387-02-15
9031	p1	مدیر	1386-04-15
9031	p3	منشی	1385-11-15
92436	p1	منشی	1387-01-04
92436	p5	NULL	1386-12-15
98825	p4	NULL	1387-08-01
98825	p5	منشی	1388-02-01

در هرکدام از جدول‌های بانک اطلاعاتی بیمارستان، با استفاده از دستور INSERT ۵ رکورد جدید درج کنید.

برای درج مقادیر در یک سطر جدید،روش های متفاوتی وجود دارد. مثال‌های ۵ تا ۷ این امکان را نشان می‌دهند.

مثال ۵)

```
USE sample;
INSERT INTO Employee VALUES ('دولت آبادی', 'داود', ۱۵۲۰۱, NULL);
```

دستور INSERT در مثال ۵ مشابه دستور مثال‌های ۱ تا ۴ است. کاربرد صریح کلیدواژه‌ی NULL، مقدار پوچ را در ستون مربوطه درج می‌کند.

درج مقادیر در بعضی (نه همه) از ستون‌های جدول، معمولاً به تعیین صریح ستون‌های مربوطه نیاز دارد. ستون‌های نادیده گرفته شده باید قابلیت پوچ بودن را داشته باشند یا دارای مقدار DEFAULT باشند.

مثال ۶)

```
USE sample;
INSERT INTO Employee (emp_no, emp_fname, emp_lname)
                VALUES (15201, 'دولت آبادی', 'داود', NULL);
```

مثال‌های ۵ و ۶ معادل هم هستند. ستون dept_no تنها ستون با قابلیت پوچ در جدول Employee است، زیرا ستون‌های دیگر در این جدول با عبارت NOT NULL در دستور CREATE TABLE تعریف شده‌اند.

ترتیب اسامی ستون‌ها در عبارت VALUE دستور INSERT می‌تواند با ترتیب اصلی ستون‌ها، که در دستور CREATE TABLE تعیین شده است، متفاوت باشد. در این حالت، ضروری است که ستون‌ها را در ترتیب جدیدی لیست کنید.

مثال ۷)

```
USE sample;
Inser Into Employee (emp_lname, emp_fname, dept_no, emp_no)
                (15201, 'دولت آبادی', 'داود', NULL);
```

۲-۱-۱۲ درج چندین سطر

دومین شـکل دسـتور INSERT، یک یا چند سطر انتخاب شـده با یک پرس و جوی فرعی را درج می‌کند.

مثال ۸)

```
USE sample;
CREATE TABLE tehran_dept
    (dept_no CHAR(4) NOT NULL,
    dept_name CHAR(20) NOT NULL);
INSERT INTO tehran_dept (dept_no, dept_name)
  SELECT dept_no, dept_name
     FROM Department
    WHERE Location = 'تهران';
```

در مثـال فوق، جـدول جدیدی به نام tehran_dept ایجاد می‌شـود که دارای همان سـتون‌های جدول Department به جز سـتون Location اسـت. پـرس و جوی فرعی در دسـتور INSERT تمام سـطرهایی را که سـتون Location آن ها با تهران برابر است انتخاب می‌کند. سـطرهای انتخاب شده در جدول جدید، درج خواهند شد. محتوای جدول tehran_dept می‌تواند با دستور SELECT زیر، نمایش داده شود:


```
USE sample;
SELECT * FROM tehran_dept;
```

نتیجه به صورت زیر است:

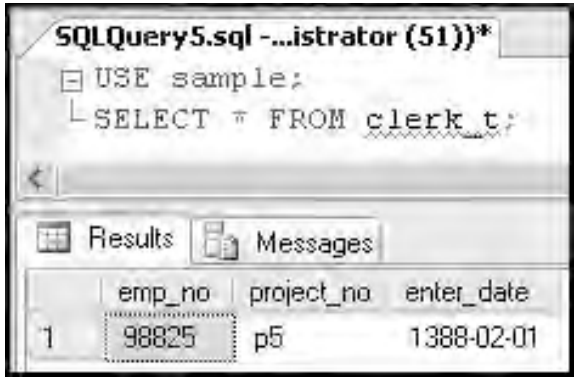
مثال ۹، مثال دیگری است که نشان می‌دهد چگونه چندین سطر می‌توانند با استفاده از شکل دوم دستور INSERT درج شوند.

مثال ۹)

شـــماره ی کارمندی، کد پروژه و تاریخ شروع به کار تمام کارمندان دفتری (منشی)را که روی پروژه‌ی P۵ کار می‌کنند را به‌دست آورید و در جدول جدیدی درج کنید.

```
USE sample;
CREATE TABLE clerk_t
    (emp_no INT NOT NULL,
    project_no CHAR(4),
    enter_date DATE);
INSERT INTO clerk_t (emp_no, project_no, enter_date)
  SELECT emp_no, project_no, enter_date
     FROM works_on
     WHERE job = 'منشی'
    AND project_no = 'p5';
```

جدول جدید (clerk_t) شامل سطر زیر است:



جدول‌های tehran_dept و clerk_t (مثال‌های ۸ و ۹) قبل از این که دستور INSERT سطرها را درج کند، خالی بودند. اگر جدولی از قبل وجود داشته باشد و سطرهایی در آن وجود دارد، سطرهای جدید به آن اضافه خواهند شد.

نکته

می‌توان به جای هر دو دستور (CREATE TABLE و INSERT) مثال ۹، از دستور SELECT به همراه عبارت INTO استفاده کرد (مثال ۴۲ واحدکار دهم).

۳-۱-۱۲ سازنده‌های مقدار جدولی و INSERT

SQL Server 2008 ویژگی جدیدی به نام table (or row) value constructor

را معرفی می‌کند.این ویژگی تعیین چندین چندگانه (سطر) را با دستورات DML ،مثل INSERT یا UPDATE ،امکان پذیر می‌کند. مثال ۱۰ نشان می‌دهد که چگونه می‌توان چندین سطر را با استفاده از یک سازنده به همراه دستور INSERT تعیین کرد.

مثال ۱۰)

```
USE sample;
INSERT INTO Department VALUES ('d7','کارگزینی','کاشان'),
                              ('d8','پخش','قزوین'),
                              ('d9','فروش','کاشان');
```

دستور INSERT در مثال ۱۰ سه سطر را به طور هم‌زمان در جدول department با استفاده از سازنده‌ی مقدار جدول، درج می‌کند. همان طور که در مثال مشاهده می‌کنید، شکل کلی سازنده، بسیار ساده است. برای استفاده از سازنده ی مقدار جدول، مقادیر هر سطر را داخل زوج پرانتز لیست کنید و هر لیست را از دیگری با یک کاما جدا سازید.

dept_no	dept_name	Location
d1	پژوهش	تهران
d2	حسابداری	کرج
d3	بازاریابی	تهران
d4	تولید	هشتگرد
d5	انبار	شهریار
d6	نقلیه	شهرري
d7	کارگزيني	کاشان
d8	پخش	قزوين
d9	فروش	کاشان

۲-۱۲ دستور UPDATE

این دستور، مقادیر سطرهای جدول را ویرایش می‌کند (تغییر می‌دهد). شکل کلی دستور به صورت زیر است:

```
UPDATE tab_name
    { SET column_1 = {expression | DEFAULT | NULL} [,...n]
    [FROM tab_name1 [,...n]]
     [WHERE condition]
```

سطرها در جدول tab_name براساس عبارت WHERE تغییر می‌کنند. برای هر سطری که ویرایش می‌شود، دستور UPDATEبه تغییر مقادیر ستون‌ها در عبارت SET می پردازد و یک ثابت (یا به طور کلی یک عبارت) را به ستون مربوطه انتساب می‌دهد. اگر عبارت WHERE حذف شود، دستور UPDATE تمام سطرهای جدول را اصلاح می‌کند(عبارت FROM را در ادامه شرح می‌دهیم).

نکته

دستور UPDATE فقط می‌تواند داده‌های یک جدول را ویرایش کنید.

مثال ۱۱) یک شغل کارمند به شماره ی کارمندی ۱۸۳۱۶ را که روی پروژه‌ی P۲ کار می‌کند به شغل مدیر تغییر دهید.


```
USE sample;
UPDATE works_on
   SET job = 'مدیر'
   WHERE emp_no = 18316
    AND project_no = 'p2';
```

دستور UPDATE در مثال فوق، فقط یک سطر از جدول works_on را ویرایش

می‌کند، زیرا ترکیب ستون‌های emp_no و project_no، کلید اولیه‌ای از آن جدول را ایجاد کرده و مقدار آن منحصر به فرد است. این مثال، شغل کارمندی را که قبلاً تعیین نشده یا NULL بوده است ویرایش می‌کند.

مثال ۱۲، سطرهای جدول را با یک عبارت، ویرایش می‌کند.

مثال ۱۲) بودجه‌ی تمام پروژه‌ها را به پوند انگلیس تغییر دهید. نرخ تقریبی تبدیل ریال به پوند انگلیس برابر ۰/۰۰۰۰۶ است.

```
USE sample;
UPDATE Project
    SET Budget = Budget*0.00006;
```

در این مثال، تمام سطرهای جدول Project ویرایش خواهند شد، زیرا عبارت WHERE وجود ندارد. سطرهای تغییر یافته از جدول Project را می‌توان با دستور زیر نمایش داد:

```
SELECT * FROM Project;
```

نتیجه عبارت است از:

	project_no	project_name	Budget
1	p1	میل لنگ	720.00
2	p2	سیستم ترمز	570.00
3	p3	فرمان	1119.00
4	p4	داشبورد	120.00
5	p5	ضد سرقت	300.00
6	p6	صندلي	600.00

تعداد پذیرش در روز تمام پزشکانی را که تخصص آن‌ها عمومی است برابر با ۳۰ نفر قرار دهید.

مثال ۱۳) به علت بیماری خانم بهروزی، تمام شغل‌های او را در تمام پروژه‌ها به NULL تغییر دهید.

```
USE sample;
 UPDATE works_on
   SET job = NULL
   WHERE emp_no IN
   (SELECT emp_no
     FROM Employee
      WHERE emp_lname = 'بهروزی');
```

پیغام (2row(s) affected) را دریافت خواهید کرد.

مثال ۱۳، از یک پرس‌وجوی درونی در عبارت WHERE دستور UPDATE استفاده می‌کند. به دلیل استفاده از عملگر IN، از این پرس‌وجو می‌توان بیش از یک سطر به دست آورد.

مثال ۱۳را می‌توان با استفاده از عبارت FROM دستور UPDATE حل کرد. عبارت FROM شامل اسامی جدول‌هایی است که در دستور UPDATE قرار می‌گیرند. تمام این جدول‌ها باید به ترتیب با هم پیوند برقرار کنند. مثال ۱۴، کاربرد عبارت FROM را نشان می‌دهد. این مثال با مثال قبلی، یکسان است.

مثال ۱۴)

```
USE sample;
UPDATE works_on
  SET job = NULL
  FROM works_on, Employee
  WHERE emp_lname = 'بهروزی'
   AND works_on.emp_no = Employee.emp_no;
```

قبل از اجرای مثال ۱۵، مبلغ کل بودجه‌ها را از پوند به ریال تبدیل کنید.

مثال ۱۵، کاربرد عبارت CASE در دستور UPDATE را شرح می‌دهد(برای کسب اطلاعات بیش تر، به واحدکار دهم رجوع کنید).

مثال ۱۵) بودجه‌ی هر پروژه را، بسته به مبلغ قبلی، به میزان ۲۰، ۱۰ یا ۵ درصد افزایش دهید. پروژه‌هایی با بودجه‌ی کم تر، درصد افزایش بیش تری خواهند داشت.

```
USE sample;
UPDATE Project
  SET Budget = CASE
     WHEN Budget >0 and Budget < 10000000 THEN Budget*1.20
       WHEN Budget >= 10000000 and Budget < 20000000 THEN
Budget*1.10
     ELSE Budget*1.05
```

END

project_no	project_name	Budget
p1	میل لنگ	13200000.0000
p2	سیستم ترمز	11400000.0000
p3	فرمان	20515000.0000
p4	داشبورد	2400000.0000
p5	ضد سرقت	6000000.0000
p6	صندلي	11000000.0000

توضیح: اضافه کردن ۲۰ درصد به یک عدد را می‌توان به صورت زیر نوشت:

۱- عدد موردنظر را در ۲۰ ضرب و بر ۱۰۰ تقسیم می‌کنیم.

۲- حاصل را با خود عدد جمع می‌کنیم.

۳- عدد به‌دست آمده، نسبت به عدد قبلی، ۲۰ درصد اضافه شده است.

y= x + x*20 /100 یا y= x*1.20

۳-۱۲ دستور DELETE

دستور DELETE سطرهای یک جدول را حذف می‌کند. این دستور، دارای دو شکل متفاوت است:

1) DELETE FROM table_name
[WHERE predicate];

2) DELETE table_name
FROM table_name [,…]
[WHERE condition];

تمام سطرهایی که در شرط WHERE مشخص شده‌اند، حذف خواهند شد. واضح است که فراخوانی ستون‌ها در دستور DELETE غیرضروری است، زیرا دستور DELETE روی

سطرها عمل می‌کند نه روی ستون‌ها.

مثال ۱۶) تمام مدیران را در جدول works_on حذف کنید.

```
USE sample;
DELETE FROM works_on
    WHERE job = 'مدیر';
```

عبارت WHERE در دستور DELETE می‌تواند شامل یک پرس‌وجوی درونی باشد.

مثال ۱۷) خانم بهروزی شـرکت را ترک کرده اسـت. تمام سطرهای مربوط به وی را از پایگاه داده حذف کنید:

```
USE sample;
DELETE FROM works_on
  WHERE emp_no IN
  (SELECT emp_no
     FROM Employee
     WHERE emp_lname = 'بهروزی');
DELETE FROM Employee
    WHERE emp_lname = 'بهروزی';
```

مثـال ۱۷را می‌توان با اسـتفاده از عبارت FROM نیز اجرا کـرد (مانند مثال ۱۸). این عبارت دارای همان مفهوم در دستور UPDATE است.

مثال ۱۸)

```
USE sample;
DELETE works_on
  FROM works_on, Employee
  WHERE works_on.emp_no = Employee.emp_no
  AND emp_lname = 'بهروزی';
```

```
DELETE FROM Employee
    WHERE emp_lname = 'بهروزی';
```

بـه کارگیـری عبـارت WHERE در دسـتور DELETE اختیاری اسـت. اگر عبارت WHERE نوشته نشود، تمام سطرهای جدول، حذف خواهند شد (مثال ۱۹).

مثال ۱۹)

```
USE sample;
DELETE FROM works_on;
```

نکته

تفاوت آشـکاری بین دستورهای DELETE و DROP TABLE وجود دارد. دسـتور DELETE (بخش یا کل) محتوای یک جدول را حذف می‌کند، در حالی که دستور DROP TABLE هم محتوا و هم شمای یک جدول را حذف می‌کند. بنابراین، بعد از دستور DELETE، جدول هنوز در پایگاه داده موجود است (اگرچه ممکن اسـت هیچ سطری نداشته باشـد)، ولی بعد از دستور DROP TABLE، جدولی وجود ندارد.

فرض کنید مدیریت بیمارستان می‌خواهد بخش گوش بیمارستان را منحل کند.در این صورت، اطلاعات تمام پزشکانی را که تخصص آن ها گوش است از بانک اطلاعاتی بیمارستان حذف کنید

۴-۱۲ دستور TRUNCATE TABLE

این دستور، یک نسخه‌ی "اجرای سریع" از دستور DELETE بدون عبارت WHERE را ارائه می‌کند. دستور TRUNCATE TABLE تمام سطرهای جدول را بسیار سریع تر از دستور DELETE حذف می‌کند، زیرا دستور TRUNCATE TABLE محتوای جدول را صفحه به صفحه حذف می‌کند، در حالی که DELETE محتوا را سطر به سطر حذف می‌کند.

دستور TRUNCATE TABLE دارای شکل کلی زیر است:

TRUNCATE TABLE table_name

نکته

در صورتی که می‌خواهید تمام سطرهای یک جدول را حذف کنید از دستور TRUNCATE TABLE استفاده کنید. این دستور به طور چشم گیری از دستور DELETE سریع تر است.

۵-۱۲ عبارت OUTPUT

نتیجه‌ی اجرای دستورهای INSERT، UPDATE و DELETE همیشه شامل متنی است که تعداد سطرهای تغییر یافته را اعلان می‌کند (برای مثال، "3rows deleted"). اگر محتوای چنین نتیجه‌ای مطابق نیازتان نیست می‌توانید از عبارت OUTPUT استفاده کنید، به طور صریح سطرهایی را که درج، ویرایش یا حذف شده‌اند نمایش می‌دهد.

عبارت OUTPUT از جدول‌های inserted و deleted استفاده می‌کند تا نتیجه‌ی مربوطه را نمایش دهد. هم چنین عبارت OUTPUT باید به همراه INTO استفاده شود تا جدول را پر کند. به همین دلیل، از متغیر جدولی برای ذخیره‌ی نتیجه استفاده کنید.

مثال ۲۰ نشان می‌دهد که چگونه دستور OUTPUT همراه با دستور DELETE کار می‌کند.

مثال ۲۰)

```
USE sample;
DECLARE @del_table TABLE (emp_no INT, emp_lname CHAR(20));
DELETE Employee
OUTPUT DELETED.emp_no, DELETED.emp_lname INTO @del_table
WHERE emp_no > 15000;
SELECT * FROM @del_table
```

نتیجه به صورت زیر است:

	emp_no	emp_lname
1	100101	شمس
2	18316	علوي
3	18475	صدافتي
4	25348	همتيار
5	28559	جهرمي
6	29346	حيدري
7	35498	قمصري
8	92436	جماعتي
9	98825	محب زاده

ابتدا در مثال فوق، یک متغیر جدولی به نام del_table @ به همراه دو ستون emp_no و emp_lname تعریف شـده است. این جدول برای ذخیره‌ی سطرهای حذف شده مورد استفاده قرار خواهد گرفت. شکل کلی دستور DELETE با گزینه‌ی OUTPUT به صورت زیر است:

OUTPUT DELETED.emp_no, DELETED.emp_lname INTO @del_table

با اسـتفاده از این گزینه، سیستم، سطرهای حذف شـده را در جدول deleted ذخیره می‌کند و سپس، در متغیر جدولی del @ کپی می‌شود.

مثال ۲۱، استفاده از گزینه‌ی OUTPUT در دستور UPDATE را نشان می‌دهد.

مثال ۲۱)

```
USE sample;
DECLARE @update_table TABLE
   (emp_no INT, project_no CHAR(20),old_job CHAR(20),new_job
```

```
CHAR(20));
    UPDATE works_on
    SET job = NULL
    OUTPUT DELETED.emp_no, DELETED.project_no,
        DELETED.job, INSERTED.job INTO @update_table
    WHERE job = 'منشی';
     SELECT * FROM @update_table
```

نتیجه عبارت است از:

	emp_no	project_no	old_job	new_job
1	100101	p4	منشي	NULL
2	100101	p6	منشي	NULL
3	92436	p1	منشي	NULL
4	98825	p5	منشي	NULL

۶-۱۲ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شـده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی آن را درک کند.

واژه	معنی	واژه	معنی
delete	حذف کردن	remain	باقی ماندن
by using	با استفاده از	search	جست جو کردن
statement	عبارت	specify	تعیین کردن - مشخص نمودن
remove	حذف کردن - از بین بردن	qualify	مقید ساختن - محدود کردن
view	دیدگاه - منظره	additional	اضافی
simplify	ساده‌سازی کردن	clause	عبارت
form	شکل - فرم	condition	شرط - وضعیت
syntax	نحو - قاعده - شکل کلی	qualification	شرط- تعریف - صلاحیت
parameter	پارامتر	predicate	گزاره - مستند کردن
meet	ملاقات کردن - برخورد کردن	row	سطر

Deleting Rows by Using DELETE

The DELETE statement removes one or more rows in a table or view.

A simplified form of the DELETE syntax is:

DELETE table_or_view

FROM table_sources

WHERE search_condition

The parameter table_or_view names a table or view from which the rows are to be deleted. All rows in table_or_view that meet the

qualifications of the WHERE search condition are deleted. If a WHERE clause is not specified, all the rows in table_or_view are deleted. The FROM clause specifies additional tables or views and join conditions that can be used by the predicates in the WHERE clause search condition to qualify the rows to be deleted from table_or_view. Rows are not deleted from the tables named in the FROM clause, only from the table named in table_or_view.

Any table that has all rows removed remains in the database. The DELETE statement deletes only rows from the table; the table must be removed from the database by using the DROP TABLE statement.

بـه طور کلی، فقط با سـه دسـتور SQL که می‌توان محتوای جـدول (داده‌ها) را تغییر داد،کـه عبارت‌انـد: INSERT، UPDATE و DELETE. برای درج سـطرها از دسـتور INSERT و برای ویرایش و حذف سطرها به ترتیب از دستورهای UPDATE و DELETE اسـتفاده کنید. دستور غیر اسـتاندارد TRUNCATE TABLE شـکل دیگری از دستور DELETE بدون شرط است که سبب حذف تمام رکوردهای جدول می‌شود. حذف سطرها با TRUNCATE TABLE از DELETE سریع تر انجام می‌شود.

بـا اسـتفاده از عبـارت OUTPUT می‌توان نتیجـه‌ی اجرای دسـتورات تغییر داده‌ها (INSERT, UPDATE, DELETE)را کـه در جدول‌هـای موقتـی ذخیره می‌شـوند به صورت جدولی مشاهده کرد و از درست بودن عملیات مطمئن شد.

خودآزمایی

۱- داده‌های کارمند جدیدی را به نام جعفر قشقایی که شماره ی کارمندی وی ۱۱۱۱۱ است درج کنید. شماره ی بخش وی هنوز مشخص نیست.

۲- جدول جدیـدی به نام d۲_emp_d۱ با تمام کارمندانی که برای بخش d۱ یا d۲ کار می‌کنند، ایجاد کنید و سطرهای مربوطه را از جدول Employee در آن درج نمائید. دو راه حل متفاوت و در عین حال یک سان پیدا کنید.

۳- جدول جدیدی ایجاد کنید و در آن از بانک اطلاعاتی بیمارستان،نام تمام بیمارانی را که به وسیله ی پزشک عمومی ویزیت شده‌اند، درج کنید.

۴- جدول جدیدی از تمام کارمندانی که تاریخ شـروع به کار آن ها در سال ۱۳۸۷ بوده است، ایجاد کنید و سطرهای مربوطه را از جدول Employee در آن درج کنید.

۵- شغل تمام کارمندانی را که در پروژه‌ی P۱ به عنوان مدیر کار می‌کنند ویرایش کنید. آن ها از این به بعد باید به عنوان منشی کار کنند.

۶- بودجه‌ی تمام پروژه‌هایی را که تعیین نشده‌اند به NULL تغییر دهید.

۷- شغل کارمندی با شماره ی کارمندی ۲۸۵۵۹ را تغییر دهید. از این به بعد، او در تمام پروژه‌ها مدیر خواهد بود.

۸- بودجـه‌ی پروژه‌ای را که مدیر آن دارای شـماره کارمندی ۱۰۱۰۲ اسـت ۱۰درصد افزایش دهید.

۹- نام بخشی را که کارمندی به نام محب زاده در آن کار می‌کند تغییر دهید. نام جدید بخش، فروش است.

۱۰- تاریخ شروع به کار کارمندانی را که روی پروژه‌ی P۱ کار می‌کنند و متعلق به بخش فروش هستند تغییر دهید. تاریخ جدید ۱۳۸۷،۱۲،۱۲است.

۱۱-تمام بخش‌هایی را که در تهران هستند حذف کنید.

۱۲- پروژه‌ی P۳ خاتمه یافته است. تمام اطلاعات مربوط به این پروژه را در پایگاه داده‌ی sample حذف کنید.

۱۳- اطلاعات جدول works_onرا برای تمام کارمندانی که بخش آن ها در کرج است حذف کنید.

۱۴- چرا ســرعت اجرای دستور TRUNCATE در حذف تمام رکوردها بیش از دستور DELETE است؟

۱۵- آیا می‌توان به جای دستور UPDATE از ترکیب دو دستور DELETE و INSERT استفاده کرد؟ چگونه؟

توانایی کار با جدول

هدف های رفتاری

پس از پایان این فصل، هنرجو قادر خواهد بود:

- دستورات برنامه‌نویسی و اجزای آن را شرح دهد و به کار گیرد؛
- رویّه‌های دستورات SQL را ایجاد و اجرا کند؛
- توابع را ایجاد، ویرایش و حذف کند؛
- توابع تعریف شده‌ی کاربر را در بازیابی اطلاعات مورد استفاده قرار دهد؛
- عملگر APPLY را شرح دهد و به کار گیرد.

روال‌های ذخیره شده[1] مجموعه‌ای از دستورات SQL هستند که با هم کامپایل می‌شوند و با یک دستور SQL به اجرا در می‌آیند. این روال‌ها معمولاً برای بازیابی اطلاعات، اضافه کردن رکوردها، تغییر داده‌های جدول، حذف رکوردها و انجام کارهای سیستمی و مدیریتی به کار می‌روند.

ترتیبی از دستورات Transact-SQL و دستورات برنامه‌نویسی را می توان یک رویّه دانست. یک رویّه می‌تواند روال ذخیره شده یا تابع تعریف شده‌ی یک کاربر[2] (UDF) باشد. رویّه می‌تواند به صورت شیء پایگاه داده (یک روال ذخیره شده یا UDF) ذخیره شود.

بعضی از رویه‌ها به وسیله‌ی کاربران نوشته می‌شوند و برخی دیگر را به وسیله‌ی Microsoft ارائه می دهند و رویه‌های سیستمی نامیده می‌شوند. در مقایسه با روال‌های ذخیره شده، UDFها مقداری را به فراخوان برمی‌گردانند. تمام رویه‌ها می‌توانند در Transact-SQL نوشته شوند. رویه‌ها کاربردهای مختلفی دارند که برخی از آن ها عبارت اند از:

۱-۱۳ دستورات برنامه‌نویسی[3]

واحدکارهای قبلی، دستورات Transact-SQL را معرفی کردند که متعلق به زبان تعریف داده‌ها و زبان کارکردن با داده‌ها بودند. اغلب این دستورات می‌توانند با هم گروه‌بندی شوند تا یک رویه را ایجاد کنند. همان طور که قبلاً نیز ذکر شد، ترتیبی از دستورات SQL و دستورات برنامه‌نویسی یک رویه است که برای اجرا به سیستم پایگاه داده ارسال می‌شود. تعداد دستورات یک رویه براساس اندازه‌ی شیء رویه‌ای کامپایل شده، محدود می‌شود. مزیت اصلی یک رویه، گروه‌بندی دستورات مجزایی است که تمام دستورات را به صورت پشت سرهم به یک باره اجرا می‌کند تا کارایی را افزایش دهد.

1- Stored Procedure

2- User Defined Function

۳- نظر به آشنا بودن هنرجویان با مفاهیم برنامه‌نویسی، این قسمت به طور مبسوط بیان نشده است.

نکته

برای جدا کردن دستورات DDL از هم دیگر، از دستور GO استفاده کنید.

قسمت‌های زیر، هر کدام از دستورات برنامه‌نویسی زبان Transact-SQL را به طور مجزا شرح می‌دهند.

۱۳-۱-۱ بلاک‌بندی دستورات

یک بلاک، امکان ایجاد واحدهایی با دو یا چند دستور Transact-SQL را فراهم می‌کند. هر بلاک با دستور BEGIN شروع و به دستور END ختم می‌شود:

```
BEGIN
statement_1
statement_2
…
END
```


یک بلاک می‌تواند درون دستور IF مورد استفاده قرار گیرد تا امکان اجرای بیش از یک دستور،که وابسته به شرط اصلی هستند فراهم کند (مثال ۱ را مشاهده کنید).

۱۳-۱-۲ دستور IF

دستور IF در زبان Transact-SQL دقیقاً مشابه دستور IF است، که تقریباً تمام

زبان‌های برنامه‌نویسی آن را پشتیبانی می‌کنند. اگر شرط بعد از دستور IF درست باشد دستور یا دستورات بعد از آن اجرا می‌شوند و اگر این دستور دارای دستور ELSE باشد، در صورت نادرست بودن شرط IF، دستورات بعد از ELSE اجرا خواهند شد.

مثال ۱)

```
USE sample;
IF (SELECT COUNT(*)
     FROM works_on
     WHERE project_no = 'p1'
     GROUP BY project_no ) > 3
  PRINT 'The number of employees in the project p1 is 4 or more'
  ELSE BEGIN
    PRINT 'The following employees work for the project p1'
    SELECT emp_fname, emp_lname
    FROM Employee, works_on
    WHERE Employee.emp_no = works_on.emp_no
   AND project_no = 'p1'
  END
```

مثال ۱،کاربرد بلاک درون دستور IF را نشان می‌دهد. عبارت منطقی دستور IF، که در پایگاه داده ی نمونه برابر با TRUE،به شکل زیر است:


```
(SELECT COUNT(*)
     FROM works_on
     WHERE project_no = 'p1'
      GROUP BY project_no) > 3
```

بنابراین، دستور PRINT اجرا می‌شود. توجه داشته باشید که این مثال از پرس و جوی

فرعی‌ای برای برگرداندن تعداد سطرها (با استفاده از تابع COUNT) استفاده می‌کند که شرط 'WHERE (project_no = 'p2) را برآورده می‌سازد. نتیجه‌ی مثال ۱ به صورت زیر است:

The number of employees in the project p1 is four or more

نکته

بخش ELSE دستور IF در مثال ۱ شامل دو دستور است: PRINT و SELECT. بنابراین، بلاک‌بندی با دستورات BEGIN و END برای دربرگرفتن دو دستور، مورد نیاز است(دستور PRINT دستور دیگری است مربوط به الحاقات روالی،که یک پیغام را نمایش می‌دهد).

۳-۱-۱۳ دستور WHILE

دستور WHILE به طور تکراری یک یا چند دستور را، تا زمانی که عبارت منطقی آن درست باشد، اجرا می‌کند. به بیان دیگر، اگر عبارت، درست باشد دستور (یا بلاک) اجرا می‌شود و سپس عبارت دوباره بررسی می‌شود و در صورت درست بودن، دستور (یا بلاک) دوباره اجرا خواهد شد. این عملیات تا زمانی که عبارت، نادرست شود، تکرار می‌شود.

بلاک درون دستور WHILE می‌تواند به طور اختیاری شامل یکی از دو دستور می باشد که برای کنترل اجرای دستورات درون بلاک مورد استفاده قرار می گیرند: BREAK یا

CONTINUE. دستور BREAK اجرای دستورات درون بلاک را قطع و اجرای دستور را، بلافاصله بعد از بلاک شروع می‌کند.در حالی که دستور CONTINUE فقط اجرای فعلی دستورات بلاک را قطع و اجرای بلاک را از ابتدا شروع می‌کند. مثال ۲، کاربرد دستور WHILE را نشان می‌دهد.

مثال ۲)

```
USE sample;
WHILE (SELECT SUM(Budget)
        FROM Project) < 50000000
    BEGIN
     UPDATE Project SET Budget = Budget*1.1
     IF (SELECT MAX(Budget)
        FROM Project) > 24000000
     BREAK
      ELSE CONTINUE
    END
```

در مثال ۲، بودجه‌ی تمام پروژه‌ها ده درصد افزایش خواهد یافت تا زمانی که بودجه ی مجموع پروژه‌ها از ۵۰۰۰۰۰۰۰ ریال بیش تر شود.حالا، اگر بودجه‌ی یکی از پروژه‌ها از ۲۴۰۰۰۰۰۰ ریال بیش تر شود، حلقه ی تکرار متوقف خواهد شد. اجرای مثال ۲ خروجی زیر را نمایش می‌دهد:


```
(3 rows affected)
(3 rows affected)
(3 rows affected)
```

٤-١-١٣ متغیرهای محلی

برای ذخیره‌ی مقادیر (از هر نوعی) درون یک رویّه، از متغیرهای محلی استفاده می‌شود. آن ها «محلی» هستند زیرا فقط درون رویه‌ای که تعریف شده‌اند، قابل شناسایی هستند.

هر متغیر محلی باید با استفاده از دستور DECLARE تعریف شود(شکل کلی این دستور را در مثال ۳ مشاهده می‌کنید). تعریف هر متغیری شامل نام و نوع داده‌ی آن است. متغیرها همیشه در یک رویه با استفاده از پیشوند @ شناسایی می‌شوند. مقداردهی یک متغیر محلی به یکی از دو شکل زیر انجام می‌شود:

شکل خاصی از دستور SELECT

دستور SET

کاربرد هر دو حالت مقداردهی در مثال ۳ آورده شده است.

مثال ۳)

```
USE sample;
DECLARE @avg_budget MONEY, @extra_budget MONEY
SET @extra_budget = 1500000
SELECT @avg_budget = AVG(budget) FROM Project
IF (SELECT Budget
FROM Project
WHERE project_no='p1') < @avg_budget
BEGIN
UPDATE Project
SET Budget = Budget + @extra_budget
WHERE project_no ='p1'
PRINT 'Budget for p1 increased by @extra_budget'
END
ELSE PRINT 'Budget for p1 unchanged'
```

نتیجه عبارت است از:

Budget for p1 increased by @extra_budget

رویه مثال ۳، میانگین بودجه‌ی تمام پروژه‌ها را محاسبه و این مقدار را با بودجه‌ی پروژه P1 مقایسه می‌کند. اگر مبلغ پروژه‌ی P1 از میانگین کوچک تر باشد، بودجه‌ی پروژه‌ی P1 به اندازه‌ی متغیر محلی extra_budget @ افزایش خواهد یافت.

۵-۱-۱۳ سایر دستورات برنامه‌نویسی

زبان Transact-SQL شامل دستورات زیر نیز هست:

- RETURN
- GOTO
- WAITFOR

دستور RETURN در داخل یک رویه دارای همان وظیفه‌ای است که BREAK در درون WHERE دارد.به این بدین معنی که دستور RETURN سبب می‌شود تا اجرای رویه قطع شود و اولین دستور بعد از رویه به اجرا در آید.

دستور GOTO سبب پرش به برچسبی می‌شود که قبل از دستور Transact-SQL به داخل رویه آورده شده است.

دستور WAITFOR وقفه ی زمانی (در صورت استفاده از گزینه‌ی DELAY) یا زمان تعیین شده‌ای (در صورت استفاده از گزینه‌ی TIME) را تعریف می‌کند که سیستم باید، قبل از اجرای دستور بعدی در رویه، منتظر بماند. شکل کلی این دستور، به صورت زیر است:

WAITFOR {DELAY 'time' | TIME 'time' | TIMEOUT 'timeout' }

گزینه‌ی DELAY به سیستم پایگاه داده اعلام می‌کند تا به میزان سپری شدن زمان

تعیین شـــده، منتظر بماند. TIME، زمانی را در یکی از قالب‌های قابل قبول داده‌های زمانی تعیین می‌کند. TIMEOUT ،میزان زمان برحســـب میلی‌ثانیـــه را تعیین می‌کند تا منتظر دریافت پیامی در صف باشد.

۲-۱۳ توابع تعریف شده‌ی کاربر

چنان که می‌دانید، در زبان‌های برنامه‌نویسی، به طور کلی دو نوع رویه وجود دارد:

- روال‌های ذخیره شده
- توابع تعریف شده‌ی کاربر (UDFها)

همان طور که قبلاً نیز بیان شد، روال‌های ذخیره شده از چندین دستور تشکیل شده‌اند کـــه فاقد یا حایـــز چند پارامتر ورودی انـــد، ولی معمولاً هیچ مقـــداری را برنمی‌گردانند. در مقایســـه، توابع همیشه دارای یک مقدار برگشـــتی هستند. این قسمت، ایجاد و به کارگیری توابع را شرح می‌دهد.

توابع، برنامه‌هایی هســـتند که از قبل نوشـــته شده‌اند و می‌توان از آن ها استفاده کرد. دو نوع تابع در SQL Server وجود دارد:

۱- توابعی که برنامه ی SQL Server آن ها را ارائه می‌کند،که قبلاً با تعدادی از آن ها آشنا شـــدید و بقیه ی آن ها در پیوست ب در انتهای کتاب معرفی شده‌اند و لیست آن ها در پوشه‌ی system Functions قابـــل مشـــاهده اســـت (شکل مقابل).

۲- توابعی که برنامه‌نویس، می‌نویسد. برنامه‌نویس ممکن است به توابعی نیاز داشته باشد که در SQL Server موجود نباشـــند. بنابراین، باید قادر شد که توابعی را بنویسد و به بانک اطلاعاتی اضافه کند.توابع نوشته شده را ویرایش و حذف کند.

۱-۲-۱۳ ایجاد و اجرای توابع تعریف شده‌ی کاربر

به دو روش می‌توان تابع را ایجاد کرد:

۱- ایجاد تابع در محیط Management Studio

۲- ایجاد تابع با دستور CREATE FUNCTION

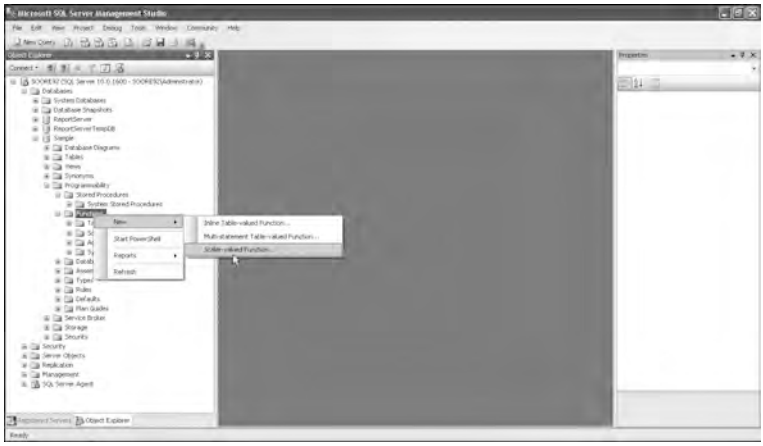
برای ایجاد تابع با روش اول، مراحل زیر را انجام دهید:

۱- روی بانک اطلاعاتی مورد نظر (sample) دابل کلیک کنید تا لیســت شیءهای آن را مشاهده کنید.

۲- روی پوشه‌ی Programmability دابل کلیک کنید تا باز شود.

۳- روی پوشه‌ی Functions دابل کلیک کنید تا لیست توابع را مشاهده کنید.

۴- روی پوشـــه‌ی Functions راســت کلیک کنید و از منوی بازشـــو، گزینه‌ی New و سپس Scalar-Valued Function را انتخاب کنید.



۵- دستورات زیر ظاهر می‌شوند:

```
-- ============================================
-- Template generated from Template Explorer using:
-- Create Scalar Function (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the function.
-- ============================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:          <Author,,Name>
-- Create date: <Create Date, ,>
-- Description:     <Description, ,>
-- ============================================
CREATE FUNCTION <Scalar_Function_Name, sysname, FunctionName>
(
    -- Add the parameters for the function here
```

```
<@Param1, sysname, @p1> <Data_Type_For_Param1, , int>
)
RETURNS <Function_Data_Type, ,int>
AS
BEGIN
    -- Declare the return variable here
    DECLARE <@ResultVar, sysname, @Result> <Function_Data_
Type, ,int>
    -- Add the T-SQL statements to compute the return value here
SELECT <@ResultVar, sysname, @Result> = <@Param1,sysname, @ p1>
    -- Return the result of the function
    RETURN <@ResultVar, sysname, @Result>
END
GO
```

۶- این دستورات، الگوی اضافه کردن تابع را نشان می‌دهند. دستورات را مطابق مطالبی که در ادامه بیان می‌شود، تغییر دهید.

۷- کلید F5 را فشار دهید تا تابع جدید، ایجاد شود.

توابع را با دســتور CREATE FUNCTION نیز می‌توان ایجاد کرد،که شکل کلی آن به صورت زیر است:


```
CREATE FUNCTION [schema_name.]function_name
[({@param } type [= default]) {,...}
RETURNS {scalar_type | [@variable] TABLE}
[WITH {ENCRYPTION | SCHEMABINDING}
[AS] {block | RETURN (select_statement)}
```

schema_name، نام شمایی است که مالک تابع ایجاد شده، تعیین می‌کند. _function name، نام تابع جدید است. param @ یک پارامتر ورودی است که type نوع داده‌ی آن را تعیین می‌کند. پارامترها مقادیر ارسال شده از فراخوان تابع اند و داخل تابع، مورد استفاده قرار می‌گیرند. default مقدار پیش‌فرض اختیاری پارامتر مربوطه را تعیین می‌کند (پیش‌فرض می‌تواند NULL نیز باشد).

عبارت RETURNS نوع داده‌ی مقدار برگشتی تابع را تعریف می‌کند. این نوع داده می‌تواند یکی از انواع داده‌ی استاندارد پشتیبانی شده به وسیله‌ی سیستم پایگاه داده باشد(تنها نوع داده‌ای که نمی‌توان استفاده کرد، TIMESTAMP است).

توابع مقدار اسکالر یا جدولی دارند. یک تابع با مقدار برگشتی اسکالری مقدار غیر قابل تجزیه (اسکالر) را برمی‌گرداند.به این بدین معنی که در عبارت RETURNS یک تابع، مقدار اسکالر،که یکی از انواع داده‌های استاندارد است، برگردانده می‌شود. اگر عبارت RETURNS مجموعه‌ای از سطرها را برگرداند، تابع با مقدار برگشتی جدولی خواهد بود.

گزینه‌ی WITH ENCRYPTION، اطلاعاتی را در سیستم کدگذاری می‌کند که شامل متن دستور CREATE FUNCTION باشد. در آن حالت، نمی‌توان متن مورد استفاده برای ایجاد تابع را مشاهده کرد(از این گزینه برای توسعه‌ی امنیت سیستم پایگاه داده استفاده کنید).

عبارت جای گزین UDF ،(WITH SCHEMA BINDING را به شیءهایی که به آن رجوع می‌کنند، مقید می‌نماید. هر تغییری در ساختار شیء پایگاه داده، ارجاعات تابع را قطع می‌کند.

در صورتی که در طول ایجاد تابع از عبارت SCHEMABINDING استفاده می‌کنید، شیءهای پایگاه داده، که به وسیله‌ی تابع رجوع می‌شوند، باید شرایط زیر را به طور کامل داشته باشند:

- تمام دیدگاه‌ها و توابعی که به وسیله تابع رجوع می‌شوند باید مقید به شمای باشند.
- تمام شیء‌های پایگاه داده (جدول‌ها، دیدگاه‌ها یا UDFها) باید در همان پایگاه داده‌ای باشند که تابع هستند.

block، بلاک BEGIN/END است که شامل پیاده‌سازی تابع است. دستور پایانی بلاک باید یک دستور RETURN همراه با یک آرگومان باشد(مقدار آرگومان همان مقدار برگشتی تابع است). در بدنه‌ی یک بلاک BEGIN/END، فقط دستورات زیر ارائه می‌شوند:

- دستورات انتساب مثل SET
- دستورات کنترلی مثل WHILE و IF
- دستورات تعریف متغیرهای داده‌ای محلی (DECLARE).
- دستورات SELECT، شامل لیست‌های SELECT همراه با عباراتی که متغیرهای محلی تابع را مقداردهی می‌کنند.
- دستورات INSERT، UPDATE و DELETE ،که متغیرهای نوع داده‌ی TABLE را (که برای تابع محلی هستند) اصلاح می‌کنند.

به طور پیش فرض، اعضای sysadmin، db_owner و db_ddladmin می‌توانند از دستور CREATE FUNCTION استفاده کنند.

مثال ۴، ایجاد تابعی به نام compute_costs را نشان می‌دهد.

مثال ۴) روی دکمه‌ی New Query کلیک و دستورات زیر را تایپ کنید:

```
-- This function computes additional total costs that arise
-- if budgets of projects increase
USE sample;
GO
CREATE FUNCTION compute_costs (@percent INT =10)
```

```
RETURNS DECIMAL(16,2)
BEGIN
DECLARE @additional_costs DEC (14,2), @sum_budget dec(16,2)
SELECT @sum_budget = SUM (Budget) FROM Project
SET @additional_costs = @sum_budget * @percent/100
RETURN @additional_costs
END
```

کلید F10 را فشـار دهید. در صورتی که پیام `Command(s) completed successfully` را دریافت کردید، تابع با موفقیت ایجاد شده است.

تابـع compute_costs هزینـه ی اضافـی را، که ناشـی از افزایـش بودجه ی همه ی پروژه‌هاسـت، محاسـبه می‌کند. تنها متغیر ورودی (percent @)، درصد افزایش بودجه را تعییـن می‌کند. بلاک BEGIN/END ابتدا دو متغیر محلی را تعریف می‌کند که عبارت‌اند از: additional_costs @ و sum_budget@ @. سپس تابع با استفاده از شکل خاصی از دسـتور SELECT، حاصل جمع تمام بودجه‌ها را در متغیر sum_budget@ قرار می‌دهد. بعد از آن، تابع، کل هزینه‌ی اضافی را محاسـبه می کند و این مقدار را با اسـتفاده از دستور RETURN برمی‌گرداند.

اجرای توابع تعریف شده‌ی کاربر

هر تابع می‌تواند در دستورات Transact-SQL مثل SELECT، INSERT، UPDATE یا DELETE فراخوانده شود. برای فراخوانی تابع، نام آن و به دنبال آن پرانتزها را قرار دهید. داخـل پرانتزها می‌توان یک یا چنـد آرگومان را تعیین کرد. آرگومان‌هـا مقادیر یا عباراتی هستند که به پارامترهای ورودی ارسال می‌شوند. هنگامی که تابعی را فراخوانی می‌کنید،باید تمام پارامترهایی را که مقادیر پیش فرض ندارند تعیین کنید و ترتیب آرگومان‌ها باید مطابق

با ترتیب پارامترها در هنگام تعریف تابع باشد.

مثال ۵ ،کاربرد تابع compute_costs (مثال ۴) در دستور SELECT را نشان می‌دهد.

مثال ۵)

```
USE sample;
SELECT project_no, project_name
    FROM Project
     WHERE Budget < dbo.compute_costs(25)
```

نتیجه به صورت زیر است:

	project_no	project_name
1	p1	میل لنگ
2	p2	سیستم ترمز
3	p4	داشبورد
4	p5	ضد سرقت
5	p6	صندلي

دستور SELECT در مثال ۵،نام و شماره‌ی تمام پروژه‌هایی را که بودجه‌ی آن ها از کل هزینه‌ی اضافی تمام پروژه‌ها برای درصد ارائه شده کم تر است نمایش می‌دهد.

نکته

هر تابعی از آن که در دستور Transact-SQL استفاده می‌شود، باید با استفاده از نام دو قسمتی آن مشخص شود:

schema_name.function_name

پرس‌وجویی بنویسید که به کمک یک تابع، مشخصات پزشکانی را نمایش دهد که تعداد پذیرش آن‌ها در روز کم تر از ۱۰ نفر است.

نوع داده‌ی جدولی

همان طور که می‌دانید، تابع در صورتی که عبارت RETURNS مجموعه‌ای از سطرها را برگرداند، جدول مقدار است. تابع جدول مقدار، به تناسب تعریف بدنه‌ی تابع، می‌تواند به صورت درون خطی[1] یا چند دستوری[2] تقسیم‌بندی شود. اگر عبارت RETURNS جدولی را بدون هیچ لیستی از ستون‌ها مشخص کند تابع، یک تابع درون خطی است.

توابع درون خطی، نتیجه‌ی دستور SELECT را به صورت متغیری از نوع داده‌ی TABLE برمی‌گردانند (مثال ۶ را مشاهده کنید). یک تابع مقدار جدولی چند دستوری، شامل نامی است که به دنبال TABLE ارائه می‌شود(نام، یک متغیر درونی از نوع TABLE را تعریف می‌کند). می‌توان از این متغیر، برای درج سطرها در درون آن و سپس برگرداندن متغیر به صورت مقدار برگشتی تابع، استفاده کرد.

مثال ۶، تابعی را نشان می‌دهد که متغیری از نوع داده‌ی TABLE را برمی‌گرداند.

مثال ۶)

```
USE sample;
GO
CREATE FUNCTION employees_in_project (@pr_number CHAR(4))
   RETURNS TABLE
  AS RETURN (SELECT emp_fname, emp_lname
```

1- Inline Table-valued Function

2- Multi-statement Table-valued Function

```
FROM works_on, Employee
WHERE Employee.emp_no = works_on.emp_no
AND project_no = @pr_number)
```

تابع employees_in_project ،نام کارمندانی را که متعلق به پروژه‌ی خاصی هستند نمایش می‌دهد. پارامتر ورودی @pr_number کد پروژه را تعیین می‌کند. در حالی که تابع به طور کلی مجموعه‌ای از سطرها را برمی‌گرداند، عبارت RETURNS شامل نوع داده‌ی TABLE است(توجه داشته باشید که بلاک BEGIN/END در مثال ۶، هنگامی که عبارت RETURNS شامل دستور SELECT است، باید حذف شود).

مثال ۷،کاربرد تابع employees_in_project را نشان می‌دهد.

مثال ۷)

```
USE sample;
SELECT *
   FROM employees_in_project('p3');
```

نتیجه به صورت زیر است:

	emp_fname	emp_lname
1	نگین	حمیدی
2	سمیه	شیرازی
3	مهناز	خسروی

زبان Transact-SQL برای ویرایش ساختار تابع از دستور ALTER FUNCTION استفاده می‌کند. تمام گزینه‌های این دستور مشابه گزینه‌های دستور CREATE FUNCTION هستند. برای ویرایش تابع، می‌توان روی نام آن در Object Explorer کلیک راست نموده و گزینه‌ی Modify را انتخاب کرد. حذف تابع نیز به همین روش با انتخاب گزینه‌ی Delete ممکن است. یک تابع با استفاده از دستور DROP FUNCTION

نیز حذف می‌شود. تنها مالک تابع می‌تواند تابع را حذف کند.

DROP FUNCTION { [*schema_name***.**] *function_name* } [**,***...n*]

۱۳-۳ عملگر APPLY

این عملگر شبیه پرس‌وجوهای بازگشتی است، زیرا از CTEها (قبلاً با آن ها آشنا شده‌اید) به همان طریق استفاده می‌کند. این عملگر امکان ترکیب یک تابع با مقدار جدولی را با سطری که به وسیله‌ی یک عبارت جدولی بیرون از پرس‌وجو برگردانده شده است فراهم می‌کند.

تابع با مقدار جدولی به صورت ورودی راست و عبارت جدولی بیرونی به صورت ورودی چپ عمل می‌کنند. ورودی راست به ازای هر سطری از ورودی چپ، ارزیابی می شود و سطرهای تولید شده برای خروجی نهایی ترکیب می‌شوند. لیست ستون‌های تولید شده به وسیله‌ی عملگر APPLY، مجموعه‌ای از ستون‌ها در ورودی چپ است که به دنبال ورودی راست برگردانده می‌شوند.

۱۳-۳-۱ توابع جدول مقدار و APPLY

عملگر APPLY می‌تواند با تابع جدول مقدار ترکیب شود تا نتایجی شبیه به عملیات پیوند بین دو جدول را تولید کند. دو مثال زیر نشان می‌دهند که چگونه می‌توان از APPLY استفاده کرد.

مثال ۸)

```
 generate function--
create function dbo.fn_getjob(@empid AS INT)
    RETURNS TABLE AS
RETURN
  SELECT job
     FROM works_on
```

```
WHERE emp_no = @empid
AND job IS NOT NULL AND project_no = 'p1';
```

تابع fn_getjob() در مثال ۸، سطرهایی از جدول works_on را برمی‌گرداند. این نتیجه در مثال زیر با محتوای جدول Employee پیوند خورده است.

مثال ۹)

```
use CROSS APPLY--
SELECT E.emp_no, emp_fname, emp_lname, job
FROM Employee as E
CROSS APPLY dbo.fn_getjob(E.emp_no) AS A
use OUTER APPLY--
SELECT E.emp_no, emp_fname, emp_lname, job
FROM Employee as E
OUTER APPLY dbo.fn_getjob(E.emp_no) AS A
```

نتیجه به صورت زیر است:

	emp_no	emp_fname	emp_lname	job
1	10102	نگین	حمیدی	تحلیلگر
2	9031	مهناز	خسروی	مدیر

	emp_no	emp_fname	emp_lname	job
1	10102	نگین	حمیدی	تحلیلگر
2	1852	کمیل	حسنی	NULL
3	2581	سمیه	شیرازی	NULL
4	3091	فریبا	بهروزی	NULL
5	9031	مهناز	خسروی	مدیر

در اولین پرس‌وجوی مثال ۹،حاصل تابع جدول مقدار(fn_getjobl) با محتوای جدول Employee (با استفاده از عملگر CROSS APPLY) پیوند می‌خورد.(fn_getjobl) به

صورت ورودی راســت و جــدول Employee به صورت ورودی چــپ عمل می‌کند. ورودی راســت به ازای هر ســطر از ورودی چپ اجرا می شــود و ســطرهای تولید شده برای عملگر نهایی ترکیب می‌شوند.

پرس‌وجوی دوم شــبیه اولی اســت، ولی از OUTER APPLY اســتفاده می‌کند که به عمل فراپیوند دو جدول مربوط می شود.

۱۳-٤ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شــده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی آن را درک کند.

واژه	معنی	واژه	معنی
types	**انواع**	perform	**انجام دادن**
function	**تابع - وظیفه - کارکرد**	variety	**متعدد - مختلف**
scalar	**عدد - عددی**	operation	**عملیات - عمل**
user-defined	**تعریف شده‌ی کاربر**	modify	**اصلاح کردن - ویرایش کردن**
single	**منفرد - تک**	access	**دسترسی - دست یابی**
except	**به جز**	directly	**به طور مستقیم**
table-valued	**جدول مقدار**	return	**برگرداندن - بازگشت**
inline	**داخلی**	**execute**	**اجرا کردن**
built-in	**درونی**	successfully	**به طور موفقیت آمیز**
provide	**ارائه کردن - تهیه کردن**	generate	**تولید کردن**

Types of Functions

Scalar Functions

User-defined scalar functions return a single data value of the type

defined in the RETURNS clause. For an inline scalar function, there is no function body; the scalar value is the result of a single statement. For a multistatement scalar function, the function body, defined in a BEGIN... END block, contains a series of Transact-SQL statements that return the single value. The return type can be any data type except **text**, **ntext**, **image**, **cursor**, and **timestamp**.

Table-Valued Functions

User-defined table-valued functions return a **table** data type. For an inline table-valued function, there is no function body; the table is the result set of a single SELECT statement.

Built-in Functions

Built-in functions are provided by SQL Server to help you perform a variety of operations. They cannot be modified. You can use built-in functions in Transact-SQL statements to:

Access information from SQL Server system tables without accessing the system tables directly.

Perform common tasks such as SUM, GETDATE, or IDENTITY.

Built-in functions return either scalar or table data types. For example, @@ERROR returns ۰ if the last Transact-SQL statement executed successfully. If the statement generated an error, @@ERROR returns the error number. And the function SUM(parameter) returns the sum of all the values for the parameter

یک روال ذخیره شـــده، نوع خاصی از رویّه اســـت که در زبان Transact-SQL نوشـــته می‌شود.

روال‌های ذخیره شده برای هدف‌های زیر، مورد استفاده قرار می‌گیرند:

- برای کنترل مجوز دسترسی؛
- برای ایجاد دنباله‌ی بازبینی از عملیات در جدول‌های پایگاه داده؛
- برای اعمال قواعد پایداری و تجاریِ مرتبط با تغییر داده‌ها؛
- برای بهبود کارایی عملیات تکراری.

توابع تعریف شـــده‌ی کاربر، اشـــتراک زیادی با روال‌های ذخیره شده دارند. تفاوت اصلی آن ها این اســـت که توابع از پارامترها پشـــتیبانی نمی‌کنند ولی مقداری را برمی‌گردانند که می‌تواند یک جدول نیز باشد. توابع دارای انواع زیر هستند:

- اسکالر : توابع اسکالر، مقدار منفرد را برمی‌گردانند.
- جدول مقدار : نوع داده‌ی جدولی (یعنی مجموعه‌ای)، سطرها را برمی‌گرداند.
- درونی: این توابع به صورت آماده در سیستم بانک اطلاعاتی موجود و غیرقابل ویرایش اند و می‌توانند برای دسترسی به جدول‌های سیستمی و انجام عملیاتی مانند محاسبه ی مجموع و میانگین مقادیر، به کار روند.

از عملگر APPLY می‌توان برای ترکیب توابع جدولی و پرس‌وجوهای Select اســـتفاده کرد تا نتایج آن ها را با هم مطابقت دهد.

خودآزمایی

۱- رویّه‌ای ایجاد کنید که ۳۰۰۰ سطر در جدول Employee درج کند. مقادیر ستون emp_no باید منحصر به فرد و بین ۱ تا ۳۰۰۰ باشند. تمام مقادیر ستون‌های emp_fname، lname و dept_no باید به ترتیب با مقادیر 'جواد'، 'سلامتی' و 'd۱' مقداردهی شوند.

۲- رویه‌ی تمرین ۱ را طوری تغییر دهید که مقادیر ستون emp_no به طور تصادفی با استفاده از تابع RAND تولید شوند(راهنمایی: از توابع زمانی سیستمی DATEPART و GETDATE برای تولید مقادیر تصادفی می‌توانید استفاده کنید).

دیدگاه‌ها[1] و تراکنش‌ها[2]

هدف‌های رفتاری

پس از پایان این فصل، هنرجو قادر خواهد بود:

- ایجاد، ویرایش و حذف دیدگاه‌ها را شرح و انجام دهد؛
- با استفاده از دیدگاه‌ها، رکوردهایی را به جدول اضافه کند؛
- با استفاده از دیدگاه‌ها، رکوردهای موجود جدول را حذف کند؛
- با استفاده از دیدگاه‌ها، رکوردهای جدول را ویرایش کند؛
- مفهوم تراکنش را بیان کند.

1- Views

2-Transactions

در این واحدکار، شـــیء دیگری از پایگاه داده‌ها به نام دیدگاه را معرفی می‌کنیم. دیدگاه ماننـــد پنجره‌ای در یک اتاق اســـت و هر یـــک از افراد مختلفی کـــه در جاهای مختلف اتاق نشســـته‌اند، منظره بیرون از اتاق را از زاویه ی دید خاصی مشـــاهده می‌کنند. دیدگاه مربوط بـــه یک جدول نیز به همین صورت اســـت یعنی هر کاربـــری، رکوردها و فیلدهای خاصی از جدول را مشـــاهده می‌کند. دیدگاه‌ها در مقایســـه با جدول‌های مبنـــا، بدون محدودیت‌های اصلی نمی‌توانند برای عملیات ویرایشـــی مورد استفاده قرار گیرند. این محدویت‌ها در پایان هر قسمت، شرح داده می‌شوند.

۱-۱۴ دستورات DDL و دیدگاه‌ها

در واحدکارهای قبلی، از جدول‌های مبنا برای تشریح دستورات DDL و DML استفاده شـــد. جدول مبنا شامل داده‌های ذخیره شده ی روی دیســـک است. در مقابل، دیدگاه‌ها به طـــور پیش فرض به صـــورت فیزیکی ایجاد نمی‌شـــوند. دیدگاه‌ها شـــیءهایی از پایگاه داده هســـتند که همیشـــه از یک یا چند جدول مبنا (یا دیدگاه‌ها) با استفاده از اطلاعات داده ها، مشـــتق می‌شـــوند. این اطلاعات (شـــامل نام دیدگاه و روشی که ســـطرها از جدول‌های مبنا بازیابی می‌شوند)، تنها اطلاعات مربوط به دیدگاه‌هاست که به طور فیزیکی ذخیره می‌شوند. از این رو، دیدگاه‌ها را جدول‌های مجازی نیز می‌نامند.

دیدگاه‌ها می‌توانند برای اهداف مختلفی استفاده شوند:

برای محدود کردن کاربرد ســـتون یا ســـطرهای خاصـــی از جدول. بنابرایـــن، دیدگاه‌ها می‌توانند برای کنترل دسترسی به بخش خاصی از یک یا چند جدول استفاده شوند.

برای پنهان کردن جزئیات پرس و جوهای پیچیده. اگر برنامه‌ی کاربردی پایگاه داده نیاز به پرس و جوهایی دارد که عملیات پیوندی پیچیده‌ای را در بر می‌گیرند، ایجاد یک دیدگاه مرتبط می‌تواند کاربرد چنین پرس و جوهایی را ساده کند.

برای محدود کردن مقادیر درج و ویرایش شده به محدوده‌های اصلی.

۱-۱-۱٤ ایجاد دیدگاه

به دو روش می‌توان دیدگاه را ایجاد کرد:

۱- در Microsoft SQL Server Management Studio

۲- با استفاده از دستور CREATE VIEW

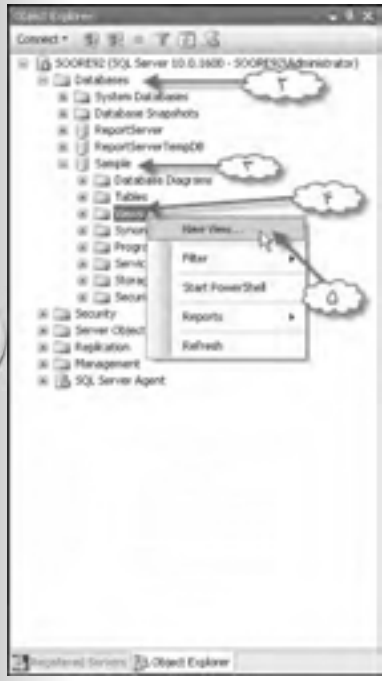
ایجاد دیدگاه در Management Studio

برای ایجاد دیدگاه در این محیط، مراحل زیر را انجام دهید:

۱- برنامه ی SQL Server Management Studio را اجرا کنید.

۲- روی پوشه‌ی Databases در پنجره‌ی Object Explorer دابل کلیک کنید.

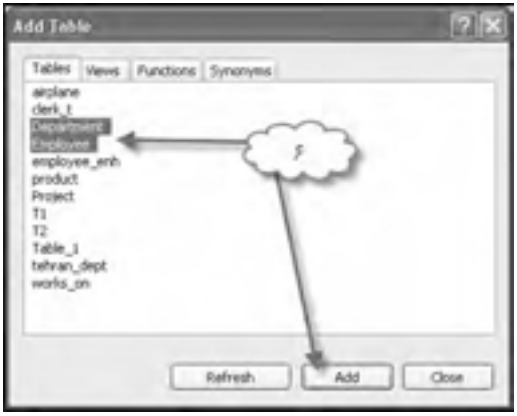
۳- روی نام بانک اطلاعاتی موردنظر (sample) دابل کلیک کنید تا شیء‌های درون آن را مشاهده نمائید.



۴- روی پوشــه‌ی Views کلیــک راســت کنید.

۵- گزینــه‌ی... New View را از منوی میانبر باز شــده، انتخاب کنید،تا پنجره‌ی Add Table ظاهــر شــود. در این پنجــره می‌توانید جدول‌های موردنظر را انتخاب کنید.

۶-جدول‌های Employee و Department را انتخاب و روی دکمه‌ی Add کلیک کنید.



۷- روی دکمه‌ی Close کلیک کنید.

۸- فیلدهـای موردنظـر را از جدول‌هـا انتخـاب کنیـد(در این مثـال، dept_name, Location, emp_fname, emp_lname).

۹- در سـتون Sort Type مربـوط به فیلدهـای emp_lname و Location گزینه‌ی Ascending را انتخاب و در ستون Sort Order، به ترتیب اعداد ۱ و ۲ را وارد کنید. اکنون رکوردها براسـاس نام خانوادگی کارمندها مرتب می‌شوند.چنان چه نام خانوادگی یک سانی داشـته باشیم، رکوردها براساس محل بخشـی که کارمند متعلق به آن است، مرتب خواهد شد.

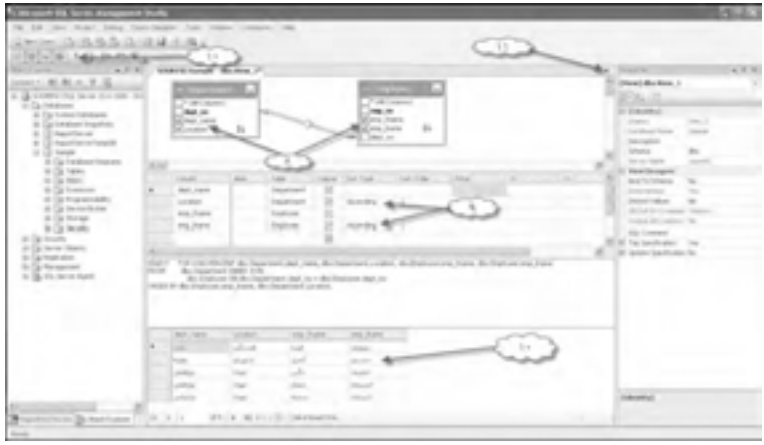
۱۰- روی دکمـه‌ی Execute SQL کلیـک کنید تا نتیجه ی اجرا را در قسـمت پایین پنجره مشاهده نمائید.

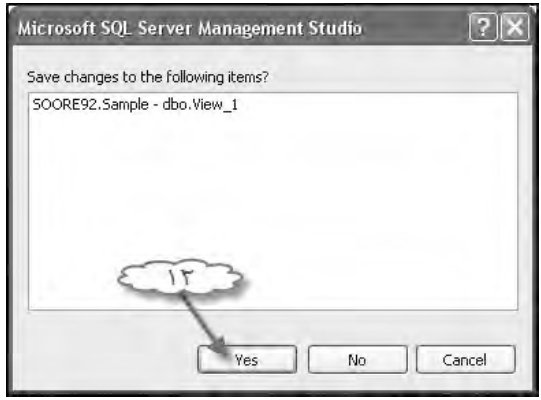
۱۱- روی دکمه‌ی Close کلیک کنید تا از پنجره‌ی ایجاد دیدگاه خارج شوید.

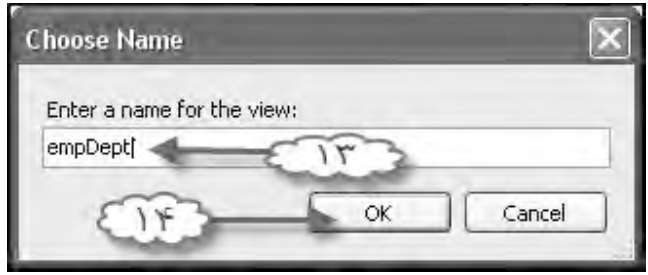
۱۲- در کادر محاوره‌ای ظاهر شوید و روی Yes کلیک کنید.

۱۳- نام دیدگاه را تعیین کنید (ما empDept را وارد کرده‌ایم).

۱۴- روی OK کلیک کنید. به این ترتیب دیدگاه جدید به لیست شیءهای پایگاه داده اضافه می‌شود.




Microsoft SQL Server Management Studio
Save changes to the following items?
SOORE92.Sample - dbo.View_1
۱۲
Yes
No
Cancel


Choose Name
Enter a name for the view:
empDept
۱۳
۱۴
OK
Cancel

ایجاد دیدگاه با دستور CREATE VIEW

یک دیدگاه را می‌توان با استفاده از دستور CREATE VIEW نیز ایجاد کرد. شکل کلی این دستور به صورت زیر است:

```
CREATE VIEW view_name [(column_list)]
[WITH {ENCRYPTION | SCHEMABINDING | VIEW_METADATA}]
AS select_statement
[WITH CHECK OPTION]
```

نکته

دستور CREATE VIEW باید تنها دستور در یک بسته باشد(به این معنی که باید از دستور GO برای جدا کردن این دستور از سایر دستورات در یک گروه دستور استفاده کرد).

view_name نام دیدگاه تعریف شده است.

column_list لیستی از اسامی مورد استفاده برای ستون‌های دیدگاه است. اگر مشخصات اختیاری نادیده گرفته شوند اسامی ستون‌های جدول مربوطه مورد استفاده قرار می‌گیرند.

گزینه‌ی WITH ENCRYPTION دستور SELECT را رمزگذاری می‌کند تا امنیت سیستم پایگاه داده افزایش یابد.

عبـــارت SCHEMABINDING دیدگاه را به شـــمایی از جدول مربوطه مقید می‌کند. هنگامی که این عبارت تعیین شـــود، شـــیءهای پایگاه داده، که در دستور SELECT ارجاع می‌شـــوند، باید اسامی دو قسمتی به شـــکل owner.db_objectداشته باشند و ممکن است db_object جدول، دیدگاه یا تابع تعریف شده‌ی کاربر باشد.

شـــرح گزینه‌ی VIEW_METADATA در حوصله ی این کتاب نیست و علاقمندان می‌توانند به کتاب های مرجع رجوع کنند.

select_statement دستور SELECT را مشخص می‌کند تا سطرها و ستون‌ها را از یک یا چند جدول (یا دیدگاه) بازیابی نماید.

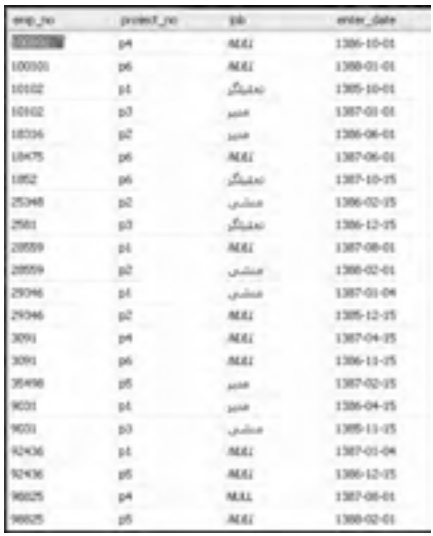
در گزینـــه‌ی WITH CHECK OPTION ،هـــر نوع اضافه کـــردن رکورد و به هنگام سازی رکوردها با تعریف اولیه دیدگاه مقایسه می‌شود. در صورتی که تعریف اولیه دیدگاه را نقض کند، انجام نخواهد شد.

مثال ۱، ایجاد یک دیدگاه را نشان می‌دهد.

مثال ۱)

```
USE sample;
GO
CREATE VIEW v_clerk
  AS SELECT emp_no, project_no, enter_date
FROM works_on
WHERE job = 'منشی';
```


پرس و جو در مثال ۱، ســـطرهایی از جدول works_on را بازیابی می‌کند که شـــرط را برآورده سازند. دیدگاه v_clrek، به عنوان سطر و ستون‌هایی که به وسیله‌ی این پرس و جو برگردانده می‌شوند، تعریف می‌شود. جدول ۱، جدول works_on را نشان می‌دهد.



emp_no	project_no	job	enter_date
[redacted]	p4	NULL	1386-10-01
100101	p6	NULL	1388-01-01
10102	p1	تحقیقگر	1385-10-01
10102	p3	مدیر	1387-01-01
18316	p2	مدیر	1386-08-01
18475	p6	NULL	1387-06-01
1852	p6	تحقیقگر	1387-10-15
25348	p2	مشاور	1386-02-15
2581	p3	تحقیقگر	1386-12-15
28559	p1	NULL	1387-08-01
28559	p2	مشاور	1388-02-01
29346	p1	مشاور	1387-01-04
29346	p2	NULL	1385-12-15
3091	p4	NULL	1387-04-15
3091	p6	NULL	1386-11-15
35498	p5	مدیر	1387-02-15
9031	p1	مدیر	1386-04-15
9031	p3	مشاور	1385-11-15
92436	p1	NULL	1387-01-04
92436	p5	NULL	1386-12-15
96825	p4	NULL	1387-08-01
96825	p5	NULL	1388-02-01

جدول ۱) جدول مبنای works_on

مثال ۱، انتخابی از سطرها را مشخص می‌کند (زیرمجموعه‌ای از جدول works_on را ایجاد می‌کند).ایجاد دیدگاهی که سطرهای محدودی از جدول را دارد نیز ممکن است. مثال ۲، ایجاد چنین دیدگاهی را نشان می‌دهد.

مثال ۲)

```
USE sample;
GO
CREATE VIEW v_without_budget
  AS SELECT project_no, project_name
    FROM project;
```


دیدگاه v_without_budget در مثال ۲، شامل تمام ستون‌های جدول project به جز Budget است. همان طور که می‌دانید، تعیین اسامی ستون‌ها با یک دیدگاه در قالب کلی دستور CREATE VIEW، اختیاری است. به عبارت دیگر،در دو حالت ،تعیین اسامی

ستون‌ها به طور واضح، مورد نیاز است:

- اگر ستونی از دیدگاه از یک عبارت یا یک تابع تجمعی مشتق شود.
- اگر دو یا چند ستون از دیدگاه دارای نام یک‌سانی در جدول مربوطه باشند.

مثال ۳، تعیین اسامی ستون‌های مرتبط با یک دیدگاه را نشان می‌دهد.

مثال ۳)

```
USE sample;
GO
CREATE VIEW v_count(project_no, count_project)
  AS SELECT project_no, COUNT(*)
FROM works_on
GROUP BY project_no;
```

اسامی ستون‌های دیدگاه v_count در مثال ۳ باید به طور دقیق تعیین شوند،زیرا دستور SELECT شامل تابع تجمعی (*) COUNT است و تمام ستون‌ها در دیدگاه باید نام‌گذاری شوند.

اگر از سرآیندهای ستون مانند مثال ۴ استفاده می‌کنید، می‌توانید از تعیین دقیق لیست ستون در دستور CREATE VIEW خودداری نمائید.

مثال ۴)


```
USE sample;
GO
CREATE VIEW v_count1
  AS SELECT project_no, COUNT(*) count_project
     FROM works_on
  GROUP BY project_no;
```

یک دیدگاه می‌تواند از دیدگاه موجود دیگری مشتق شود (مانند مثال ۵).

مثال ۵)

```
USE sample;
GO
CREATE VIEW v_project_p2
 AS SELECT emp_no
    FROM v_clerk
    WHERE project_no ='p2';
```

دیدگاه v_project_p2 در مثال ۵ از دیدگاه v_clerk (مثال ۱) مشتق شده است. هر پرس‌وجو با استفاده از دیدگاه v_project_p2 به پرس‌وجوی معادل روی جدول مبنای works_on تبدیل می‌شود.

۲-۱-۱٤ ویرایش و حذف دیدگاه‌ها

زبان Transact-SQL از دستور ALTER VIEW غیراستاندارد(که برای ویرایش تعریف پرس‌وجوی دیدگاه مورد استفاده قرار می‌گیرد) پشتیبانی می‌کند. شکل دستور ALTER VIEW شبیه دستور CREATE VIEW است.

برای جلوگیری از تعیین دوباره‌ی مجوزهای موجود دیدگاه، می‌توان از دستور ALTER VIEW استفاده کرد.نکته ی دیگر این که ویرایش دیدگاه موجود،با استفاده از این دستور، روی شیءهای پایگاه داده تأثیر نمی‌گذارد،که بستگی به دیدگاه دارد.بنابراین، اگر برای حذف و ایجاد دوباره‌ی دیدگاه از دستورات DROP VIEW و CREATE VIEW استفاده کنید، هیچ شیء پایگاه داده‌ای که به کار می‌برید به طور مناسب کار نخواهد کرد (حداقل در زمان بین حذف و ایجاد دوباره‌ی دیدگاه).

مثال ۶، کاربرد دستور ALTER VIEW را نشان می‌دهد.

مثال ۶)

```
USE sample;
GO
ALTER VIEW v_without_budget
  AS SELECT project_no, project_name
      FROM project
    WHERE project_no >= 'p3';
```

دستور ALTER VIEW در مثال فوق، دستور SELECT دیدگاه v_without_budget (مثال ۲) را با شرط جدیدی در عبارت WHERE توسعه می‌دهد.

دستور DROP VIEW تعریف دیدگاه تعیین شده از جدول‌های سیستمی را حذف می‌کند. مثال ۷، کاربرد دستور DROP VIEW را نشان می‌دهد.

مثال ۷)

```
USE sample;
GO
DROP VIEW v_count;
```

اگر دستور DROP VIEW، یک دیدگاه را حذف کند، تمام دیدگاه‌های دیگر مشتق شده از آن نیز حذف خواهند شد (مثال ۸).

مثال ۸)


```
USE sample;
GO
DROP VIEW v_clerk;
```

دستور DROP VIEW در مثال ۸، دیدگاه v_project_p2 (مثال ۵) را حذف می‌کند. برای مثال، اگر این دیدگاه را در پرس و جو به کار ببرید، خطایی را دریافت خواهید کرد: "Invalid object name = 'v_clerk'.

نکته

اگر جدولی حذف شـــود، دیدگاه مربوط به آن به طور خودکار حذف نمی‌شـــود.به این معنی که هر دیدگاه از جدول حذف شده باید به طور روشن با استفاده از دستور DROP VIEW حذف شود.

۱۴-۲ دستورات DML و دیدگاه‌ها

دیدگاه‌ها با همان دســـتورات Transact-SQL،که برای جدول‌های مبنا مورد اســـتفاده قرار می‌گیرند، بازیابی و ویرایش می‌شـــوند. بخش‌های زیر، چهار دســـتور DML مربوط به دیدگاه‌ها را شرح می‌دهند.

۱۴-۲-۱ بازیابی دیدگاه

دیدگاه دقیقاً شـــبیه جدول مبنای پایگاه داده مورد اســـتفاده قرار می‌گیرد. به کارگیری دستور SELECT در دیدگاه را می توان معادل انجام این کار در جدول‌های مبنا تصور کرد. مثال ۹،این موضوع را نشان می‌دهد.

مثال ۹)

```
USE sample;
GO
CREATE VIEW v_d2
  AS SELECT emp_no, emp_lname
     FROM Employee
```

```
    WHERE dept_no ='d1';
GO
SELECT emp_lname
    FROM v_d2
    WHERE emp_lname LIKE 'خ%';
```

نتیجه عبارت است از:

دستور SELECT در مثال فوق، با استفاده از جدول مربوط به دیدگاه v_d۲، به شکل معادل زیر تبدیل می‌شود:

```
SELECT emp_lname
FROM employee
WHERE emp_lname LIKE 'خ٪'
AND dept_no ='d1';
```

۲-۲-۱٤ دستور INSERT و یک دیدگاه

دیدگاه مانند «جدول مبنا» همراه با دستور INSERT مورد استفاده قرار می گیرد. هنگامی که دیدگاهی برای درج سطرها به کار می‌رود، سطرها به طور واقعی در جدول مربوطه درج می‌شوند.

مثال ۱۰)

```
USE sample;
GO
CREATE VIEW v_dept
```

```
  AS SELECT dept_no, dept_name
      FROM Department;
GO
INSERT INTO v_dept
  VALUES('d4', 'برنامه‌ریز');
```

دیـدگاه v_dept کـه در مثـال ۱۰ ایجاد شـده، شـامل دو سـتون ابتدایـی جدول Department است. دستور INSERT سطر را در جدول مربوطه با مقادیر d۴ و برنامه‌ریز درج می کند. سـتون Location،که به وسـیله ی دیـدگاه v_dept ارجاعی ندارد، با مقدار NULL مقداردهی می شود.

مثال های ۱۱ و ۱۲، تفاوت بین به کار گرفتن و نگرفتن از گزینه ی WITH CHECK OPTION را به ترتیب نشان می دهند.

مثال ۱۱)

```
USE sample;
GO
CREATE VIEW v_1386_check
  AS SELECT emp_no, project_no, enter_date
     FROM works_on
     WHERE enter_date BETWEEN '01.01.1386' AND '12.31.1386'
     WITH CHECK OPTION;
GO
INSERT INTO v_1386_check
   VALUES (22334, 'p2', '1.15.1387');
```

در مثال فوق، سیستم بررسی میکند که مقدار درج شده در ستون enter_date با شرط WHERE مطابقت دارد یا نه؟ درج انجام نمی شود، زیرا شرط برقرار نیست.

مثال ۱۲)

```
USE sample;
GO
CREATE VIEW v_1386_nocheck
  AS SELECT emp_no, project_no, enter_date
      FROM works_on
            WHERE enter_date BETWEEN '01.01.1386' AND
'12.29.1386';
GO
INSERT INTO v_1386_nocheck
   VALUES (22334, 'p2', '1.15.1387');
SELECT *
   FROM v_1386_nocheck;
```

نتیجه به صورت مقابل است:

	emp_no	project_no	enter_date
1	100101	p4	1386-10-01
2	18316	p2	1386-06-01
3	25348	p2	1386-02-15
4	2581	p3	1386-12-15
5	3091	p6	1386-11-15
6	9031	p1	1386-04-15
7	92436	p5	1386-12-15

به دلیل این که در مثال ۱۲ از WITH CHECK ORTION استفاده نمی کنید، دستور INSERT به اجرا در می آید و سطر در جدول works_on درج می شود. توجه کنید که دستور SELECT، سطر درج شده را نمایش نمی دهد، زیرا با استفاده از دیدگاه v_1386_nocheck نمی‌توان آن را بازیابی کرد.

اگر دیدگاه مربوطه شـامل یکی از ویژگی های زیر باشـد، درج سطرها در جدول ممکن نیست:

- عبارت FROM، در تعریف دیدگاه دو یا چند جدول را شـامل می شـود و لیست، در برگیرنده ی ستون هایی بیش از یک جدول است.
- ستونی از دیدگاه، از یک تابع تجمعی مشتق شود.
- دستور SELECT در دیدگاه، شامل عبارت GROUP BY یا گزینه ی DISTINCT است.

ستونی ازدیدگاه، از یک ثابت یا یک عبارت مشتق شود.

مثال ۱۳، دیدگاهی را نشان می دهد که نمی تواند برای درج سطرها در جدول مبنا مورد استفاده قرار گیرد.

مثال ۱۳)

```
USE sample;
GO
CREATE VIEW v_sum(sum_of_budget)
  AS SELECT SUM(budget)
     FROM project;
GO
SELECT *
    FROM v_sum;
```


مثـال ۱۳، دیـدگاه v_sum را ایجـاد می کند که شـامل یک تابع تجمعی در دسـتور SELECT اسـت. به دلیل این که دیدگاه در مثال، نتیجه ی تجمیع چندین سـطر را ارائه می کند، نمی توانید سطری را در جدول با استفاده از این دیدگاه، درج کنید.

۳-۲-۱۴ دستور UPDATE و یک دیدگاه

دیدگاه می تواند مانند «جدول مبنا» همراه با دستور UPDATE مورد استفاده قرار گیرد. هنگامی که از دیدگاه برای ویرایش سطرها استفاده می کنید، محتوای جدول مبنا به طور واقعی ویرایش می شود.

مثال ۱۴ دیدگاهی را ایجاد می کند که برای ویرایش جدول works_on مورد استفاده قرار می گیرد.

مثال ۱۴)

```
USE sample;
GO
CREATE VIEW v_p1
  AS SELECT emp_no, job
     FROM works_on
     WHERE project_no = 'p1';
GO
UPDATE v_p1
    SET job = NULL
   WHERE job = 'Manager';
```

می توان به هنگام سازی دیدگاه در مثال ۱۴ را ،به صورت دستور UPDATE زیر، معادل سازی کرد:


```
UPDATE works_on
   SET job = NULL
   WHERE job = 'Manager'
   AND project_no = 'p1'
```

گزینـــه ی WITH CHECK OPTION همـــان معنای منطقی را دارد که در دســـتور INSERT دارد. مثال ۱۵، استفاده از این گزینه را نشان می دهد.

مثال ۱۵)

```
USE sample;
GO
CREATE VIEW v_100000
  AS SELECT project_no, Budget
     FROM Project
    WHERE Budget > 10000000
    WITH CHECK OPTION;
GO
UPDATE v_100000
   SET Budget = 93000
   WHERE project_no = 'p3';
```

در مثـــال ۱۵، موتور پایگاه داده بررســـی می کند که آیا مقدار ویرایش شـــده ی ســـتون Budget با شـــرط WHERE مطابق است یا نه؟ به دلیل این که شرط درست نیست، تغییر اعمال نمی شود (مقدار ۹۳۰۰۰۰۰ بزرگ تر از ۱۰۰۰۰۰۰۰ نیست).

مثال ۱۶، دیدگاهی را نشـــان می دهد که نمی تواند برای ویرایش مقادیر سطر در جدول مبنای مربوطه مورد استفاده قرار گیرد.

مثال ۱۶)

```
USE sample;
GO
CREATE VIEW v_uk_pound (project_number, budget_in_pounds)
```

```
AS SELECT project_no, Budget*0.00065
    FROM Project
   WHERE Budget > 10000000;
GO
SELECT *
  FROM v_uk_pound;
```

نتیجه عبارت است از:

	project_number	budget_in_pounds
1	p1	8580.000000000
2	p2	7410.000000000
3	p3	13334.750000000
4	p6	7150.000000000

دیـدگاه v_uk_pound در مثال فوق نمی تواند با دسـتور UPDATE مورد اسـتفاده قرار گیرد، زیرا ستون budget_in_pounds با استفاده از عبارت ریاضی محاسبه می شود. بنابراین ستون اصلی جدول مربوطه نمایش داده نمی شود.

٤-٢-١٤ دستور DELETE و دیدگاه

دیدگاه میتواند برای حذف سـطرهایی از جدولی که به آن وابسـته است، مورد استفاده قرار گیرد (مثال ١٧).

مثال ١٧)

```
USE sample;
GO
CREATE VIEW v_project_p2
  AS SELECT emp_no, job
```

```
    FROM works_on
    WHERE project_no = 'p2';
GO
DELETE FROM v_project_p2
    WHERE job = 'منشی';
```

مثال ۱۷، دیدگاهی را ایجاد می کند که برای حذف سطرها از جدول works_on مورد استفاده قرار می گیرد. حذف سطرها در جدول مربوطه ممکن نیست، در صورتی که دیدگاه مربوطه شامل یکی از ویژگی های زیر باشد:

- عبارت FROM، در تعریف دیدگاه دو یا چند جدول را شامل می شود و لیست در برگیرنده ی ستون هایی بیش از یک جدول است.
- ستونی از دیدگاه، از یک تابع تجمعی مشتق شود.
- دستور SELECT در دیدگاه شامل عبارت GROUP BY یا گزینه ی DISTINCT است.

مثال ۱۸، دیدگاهی را نشان می دهد که می توان آن را برای حذف سطرها مورد استفاده قرار داد، ولی درج و ویرایش مقادیر ستون انجام نمی شود.

مثال ۱۸)

```
USE sample;
GO
CREATE VIEW v_budget (budget_reduction)
 AS SELECT Budget*0.9
     FROM project;
GO
DELETE FROM v_budget;
```

دستور DELETE در مثال فوق، تمام سطرهای جدول project را، که به وسیله ی دیدگاه v_budget مورد رجوع قرار گرفته است، حذف می کند.

۳-۱۴ تراکنش‌ها

تراکنش، دنباله‌ای از دستورات Transact-SQL را تعریف می‌کند که به وسیله‌ی برنامه نویسان پایگاه داده،به منظور با هم به کار بردن تا عملیات خواندن و نوشتن مورد استفاده قرار می گیرد(به صورت یک بسته). بنابراین، سیستم پایگاه داده می‌تواند پایداری داده‌ها را تضمین کند.

تراکنش (transaction)، برنامه ی فعالی است که دنباله‌ای از دستورات را شامل می‌شود و به طور خاص بعضی عملیات آن روی پایگاه داده است و دارای سه عمل خاص است:

start، که نشان می‌دهد یک تراکنش در حال شروع شدن است. commit ،که دلالت بر پایان عادی تراکنش دارد و abort که بیان کننده ی پایان یافتن تراکنش به دلیل قطع شدن آن است. تمام اثرات تراکنش قطع شده باید rollback یا بی‌اثر شود. وقتی تراکنش commit می‌شود،باید تأثیرش روی پایگاه داده دائمی شود.

نکات یک تراکنش، از طریق مثال، بهتر شرح داده می‌شود. در پایگاه داده‌ی sample، برای کارمندی به نام سمیه شیرازی، باید یک شماره‌ی کارمندی جدید تعیین شود. شماره‌ی کارمندی باید در دو جدول به صورت هم زمان تغییر یابد.باید سطری در جدول Employee و تمام سطرهای مربوطه در جدول works-on هم زمان تغییر یابند(اگر فقط یکی از این جدول‌ها اصلاح شود، داده‌های پایگاه داده‌ی sample سازگار نخواهند بود، زیرا مقادیر کلید اولیه در جدول Employee و مقادیر مربوطه در کلید خارجی جدول works-on برای خانم شیرازی یک سان نخواهند بود).

۱-۳-۱٤ مشخصات ترا کنش‌ها

هر تراکنش باید پایگاه داده را از یک حالت سازگار به حالت سازگار بعدی ببرد. تراکنش باید دارای خواص ACID باشـد تا پایگاه داده را در حالت سازگار نگه دارد. خواص ACID سر واژه‌ای چهار خاصیت زیر هستند:

- Atomicity (تجزیه ناپذیری)
- Consistency (سازگاری)
- Isolation (ایزوله کردن)
- Durability (ماندگاری)

این مشـخصات، تضمیـن می‌کنند که داده‌های مورد اسـتفاده به وسـیله‌ی برنامه‌های کاربردی، پایدار خواهند بود،به این شرح:

- مشـخصه‌ی تجزیه ناپذیری، غیر قابل تقسـیم بودن مجموعه‌ای از دستورات است که داده‌هـای پایگاه داده را اصلاح می‌کنند و بخشـی از یک تراکنش اسـت.به این بدین معنی کـه یا اصلاح تمام داده‌ها در تراکنش اجرا می‌شـود یا در صورت بـروز خطا، تمام تغییرات انجام شـده نادیده گرفته می‌شـوند. به عبارت دیگر، تراکنش‌ها اتمیک هستند یعنی یا اصلاً شـروع نمی‌شوند یا وقتی آغاز شـدند قطعاً به پایان می‌رسند(یا همه عملیات انجام می‌شود یا هیچ کدام).

- مشـخصه‌ی سازگاری تضمین می‌کند که تراکنش به پایگاه داده‌ای که شامل داده‌های ناسازگار اسـت، اعمال نخواهد شد. به عبارت دیگر، تبدیلات تراکنشی روی داده‌های پایگاه داده از یک وضعیت پایدار به وضعیت پایدار دیگری، صورت می گیرد. تراکنش، یا پایگاه داده را به حالت سازگار جدیدی می‌برد یا اگر شکستی رخ داد تمام داده‌ها به حالت قبل از شروع تراکنش برمی‌گردند.

- مشـــخصه‌ی ایزوله کردن، تراکنش‌های هم‌زمـــان را از یکدیگر مجزا می‌کند. به عبارت دیگر، تراکنش فعال نمی‌تواند تغییرات داده‌ها را در تراکنش هم‌زمان و ناقص مشاهده کند. به این معنی که بعضی از تراکنش‌ها ممکن اســـت برای تضمین ایزوله‌ســـازی، نادیده گرفته شوند. تراکنشی که در حال اجراست و هنوز به پایان نرسیده، تأثیرش از بقیه پنهان می‌ماند، مگر این که commit شـــده باشـــد. اجرای هم‌زمان تراکنش‌ها باید به صورتی باشد که انگار پشت سرهم اجرا شده‌اند. حفظ این خاصیت بر عهده‌ی کنترل هم‌زمانی است.

- مشخصه‌ی ماندگاری، یکی از مهم‌ترین مفاهیم پایگاه داده را تضمین می‌کند: پایداری داده‌ها. این ویژگی، اطمینان می‌دهد که تأثیر تراکنش خاصی پایدار است، حتی اگر خطای سیســـتمی رخ دهد. به همین دلیل، اگر یک خطای سیســـتمی همزمان با این که تراکنشی فعال است رخ دهد، تمام دستورات آن تراکنش نادیده گرفته خواهند شد. از وقتی تراکنشی commit شـــود، تأثیرش دائمی است (حتی در صورت خرابی سیستم، داده در حالت معتبر باقی می‌ماند).

۲-۳-۱٤ ثبت تراکنش

سیستم‌های پایگاه داده‌ی رابطه‌ای، هر تغییری را که در طول فعالیت تراکنش در پایگاه داده رخ می‌دهد ثبت می‌کنند(انجام این کار ضروری است، زیرا ممکن است خطایی در طول اجرای تراکنش رخ دهد). در این حالت،باید تمام دستورات اجرا شده‌ی قبلی درون تراکنش، نادیده گرفته شوند. به محض این که سیستم خطایی را تشخیص دهد، از رکوردهای ذخیره شـــده اســـتفاده می‌کند تا پایگاه داده را به وضعیت پایداری، که قبل از شروع تراکنش وجود داشت، برگرداند.

موتـــور پایگاه داده تمام رکوردهای ذخیره شـــده در قبل و بعـــد از تغییرات را در یک یا چنـــد فایل به نـــام Transaction Log نگه‌داری می‌کند. هر پایـــگاه داده دارای فایل ثبت

تراکنش خاص خودش اســت. بنابراین، اگر صرف نظر کردن از یک یا چند تغییر انجام شده روی جدول‌هــای پایگاه داده‌ی جاری، ضروری اســت، موتور پایگاه داده از ورودی های فایل ثبت تراکنش (برای بازیابی مقادیر ســتون‌هایی که پایگاه داده قبل از شروع تراکنش داشته است) استفاده می‌کند.

فایل ثبت تراکنش برای صرف نظر کردن یا بازیابی تراکنش مورد اســتفاده قرار می‌گیرد. اگر خطایی رخ دهد و تراکنش به طور کامل اجرا نشــود، سیستم از تمام مقادیر موجود قبل از ثبت تراکنش برای صرف نظر کردن از تمام تغییرات، استفاده می‌کند. فرآیندی را که قبل از ثبت تراکنش برای صرف نظر کردن از تغییرات استفاده می‌شود undo می‌نامند.

۱۴-٤ اصول خواندن و درک متن انگلیسی

متن زیر از راهنمای SQL Server انتخاب شــده است و از هنرجو انتظار می‌رود پس از به خاطر سپردن معانی واژه‌ها، بتواند متن را بخواند و مفهوم اصلی را درک کند.

معنی	واژه	معنی	واژه
ترجمه کردن- تبدیل کردن	translate	**دیدگاه**	view
عمل - عملیات - فعالیت	action	**ساختار**	structure
در برابر - در مقابل	against	**جاری - فعلی**	current
وابسته بودن - بستگی داشتن	depend on	**بررسی کردن**	check
حذف کردن	drop	**معتبر - صحیح - درست**	valid
تولید کردن	produce	**متن**	context
نقض کردن	violate	**اصلاح - ویرایش**	modification
قواعد	rules	**جامعیت**	integrity

A view can be created only in the current database. A view can have a maximum of 1,024 columns.

When querying through a view, the Database Engine checks to make sure that all the database objects referenced anywhere in the statement exist and that they are valid in the context of the statement, and that data modification statements do not violate any data integrity rules. A check that fails returns an error message. A successful check translates the action into an action against the underlying table or tables.

If a view depends on a table or view that was dropped, the Database Engine produces an error message when anyone tries to use the view. If a new table or view is created and the table structure does not change from the previous base table to replace the one dropped, the view again becomes usable. If the new table or view structure changes, the view must be dropped and re-created.

دیدگاه ها می توانند برای اهداف مختلفی، مورد استفاده قرار گیرند:

- محدود کردن استفاده از ستون ها و سطرهای خاصی از جدول ها؛
- پنهان کردن جزئیات پرس‌وجوهای پیچیده؛
- محدود کردن مقادیر درج و ویرایش شده به محدوده های خاص.

دیدگاه ها با همان دستورات Transact-SQL، که برای ایجاد، بازیابی و اصلاح جدول های مبنا مورد استفاده قرار می گیرند، ایجاد، بازیابی و ویرایش می شوند. پرس‌وجو روی یک دیدگاه همیشه به پرس‌وجوی معادل روی جدول تبدیل می شود. یک عمل ویرایشی، شبیه عملیات بازیابی، در نظر گرفته می شود. تنها تفاوت این است که روی دیدگاه مورد استفاده برای درج، ویرایش و حذف داده ها از جدولی که به آن وابسته است، چندین محدودیت وجود دارد.

دیدگاه‌ها در حقیقت جدول‌های مجازی هستند که مدیر پایگاه داده به تناسب نیاز کاربران، در اختیارشان قرار می‌دهد. همان طور که بیان شد، دیدگاه پنجره‌ای به یک منظره است که با توجه به زاویه ی دید شما این منظره متفاوت خواهد بود.

تراکنش، دنباله‌ای از دستورات است که روی شیء‌های یک پایگاه داده اجرا می‌شوند و به اعمال تغییرات در آن ها منجر می‌شود. هر تراکنشی دارای خصوصیاتی است که به آن ACID می گویند.

خودآزمایی

۱- دیدگاهی ایجاد کنید که داده های تمام کارمندانی را که برای دپارتمان d۱ کار می کنند به دست آورد.

۲- برای جدول project، دیدگاهی ایجاد کنید که بتواند به وسیله ی کارمندانی مورد استفاده قرار گیرد که امکان مشاهده ی تمام داده های این جدول به جز ستون budget را دارند.

۳- دیدگاهی ایجاد کنید که نام و نام‌خانوادگی تمام کارمندانی را داشته باشد که در نیمه ی دوم سال ۱۳۸۷ در پروژه هایشان شروع به کار کرده اند.

۴- تمرین ۳ را به نحوی تغییر دهید که ستون های f_name و l_name در دیدگاه به ترتیب اسامی first و last را داشته باشند.

۵- از دیدگاه تمرین ۱ استفاده کنید تا جزئیات کامل کارمندی را که نام‌خانوادگی وی با حرف م شروع می شود نمایش دهید.

۶- دیدگاهی ایجاد کنید که جزئیات کامل تمام پروژه هایی را که کارمندی با نام سمیه در آن ها مشغول به کار است، داشته باشد.

۷- با استفاده از دستور ALTER VIEW، شرط دیدگاه تمرین ۱ را اصلاح کنید. دیدگاه اصلاح شده باید داده هایی از کارمندانی که در دپارتمان d۱ یا d۲ یا هر دو کار می کنند، را داشته باشد.

۸- دیدگاه ایجاد شده در تمرین ۳ را حذف کنید.در این صورت، چه اتفاقی برای دیدگاه ایجاد شده در تمرین ۴ رخ می دهد؟

۹- از دیدگاه تمرین ۲ استفاده کنید و جزئیات پروژه ی جدیدی با شماره ی p۲ و نام گیربکس درج کنید.

۱۰- دیدگاهی (با عبارت WITH CHECK OPTION) ایجاد کنید که از نام و نام‌خانوادگی کارمندانی که شماره ی کارمندشان از ۱۰۰۰۰ کم تر است، تشکیل شده باشد. بعد از آن، از دیدگاه برای درج داده های کارمند جدیدی به نام ؟ با شماره ی کارمندی ۲۲۱۲۳ ،که در دپارتمان d۳ کار می کند، استفاده کنید.

۱۱- تمرین ۱۰ را بدون عبارت WITH CHECK OPTION حل کنید و تفاوتی مرتبط با درج داده ها به دست آورید.

۱۲- دیدگاهی (با عبارت WITH CHECK OPTION) ایجاد کنید که جزئیات کامل جدول works_on را برای کارمندانی که در طول سال ۱۳۸۷ و ۱۳۸۸ به پروژه وارد شده اند داشته باشد. بعد از آن، تاریخ ورود کارمندی با شماره ی کارمندی ۲۹۳۴۶ را تغییر دهید. تاریخ جدید ۱۳۸۶/۰۲/۰۶ است.

۱۳- تمرین ۱۲ را بدون عبارت WITH CHECK OPTION حل کنید و تفاوت مربوط به اصلاح داده ها را به دست آورید.

پیوست الف

نصب SQL Server ۲۰۰۸

۱–الف) برنامه‌ریزی نصب

مشـخصات و خصوصیات یک برنامه‌ی نصب، همیشـه باید قبل از نصب واقعی سیسـتم SQL Server در نظر گرفته شـود. برنامه‌ریزی دقیق ضروری است، زیرا قبل از شروع نصب باید چندین تصمیم گرفته شـود. مدیر سیسـتم باید قبل از شروع فرآیند نصب، پاسخ های روشنی به پرسش‌های زیر ارائه کند:

هدف از نصب سیستم SQL Server چیست؟

نیازمندی های سخت‌افزاری و شبکه‌ای کدام اند؟

چه تعدادی کاربر، به طور هم زمان فعال خواهند بود؟

نکته

همان طور که می‌دانید، SQL Server نام پوششی برای چندین مؤلفه مثل موتـور پایگاه داده، خدمات تجزیـه و تحلیل، خدمات گزارش گیری و جامعیت اسـت. در طول فرآیند نصـب، می‌توانید تصمیم بگیرید کـه کدام یک از این مؤلفه‌هـا روی کامپیوتر نصب شـوند. در این پیوسـت، SQL Server به تمام مؤلفه‌های سیستم، که روی کامپیوتر نصب شده‌اند، اشاره دارد.

۲-الف) هدف سیستم SQL Server

هدف از سیستم SQL Server می‌تواند متنوع باشد. برای مثال، سیستم‌تان ممکن است برای تحصیل (آموزش) به کار گرفته شود یا یک سیستم تولیدی باشد. در سیستم‌های تولیدی، نیاز است که تعداد کاربران و میزان داده‌هایی را که ذخیره خواهند شد تعیین کنید، زیرا این سیستم‌ها از نظر گستردگی متفاوت‌اند. تصمیم دیگری که در سیستم‌های پایگاه داده‌ی امروزی باید مدنظر قرار گیرد، این است که آیا سیستم برای وظایف عملیاتی یا تحلیلی مورد استفاده قرار خواهد گرفت؟

در صورتی که پایگاه داده‌تان بزرگ و برخوردار از چند صد کاربر باشد یا اگر سیستم پایگاه داده تراکنش‌های بزرگی را بارگذاری می‌کند، کارایی و عملکرد عملیات پایگاه داده‌ای موضوع مهمی خواهد بود.

در هر دو حالت، استفاده از کامپیوترهای چندپردازنده موردنیاز است تا زمان پاسخ مناسب سیستم و قابلیت مقیاس پذیری را تضمین کند. اگر یک پایگاه داده‌ی بسیار بزرگی داشته باشید، فضای ذخیره‌سازی دیسک آن نیز بسیار مهم خواهد بود. در این حالت، اگر از چندین دیسک کوچک تر، به جای یک یا دو دیسک بزرگ استفاده کنید، معمولاً عملکرد بهتری خواهد داشت.

باید بین سیستم‌های مورد استفاده برای وظایف عملیاتی (سیستم‌هایی هستندکه نیاز به دسترسی سریع و تراکنش‌های کوتاه دارند) و سیستم‌های مورد استفاده برای وظایف تحلیلی (سیستم‌هایی هستند که از عملیات بازیابی پیچیده روی پایگاه داده‌های بزرگ استفاده می‌کنند) تفاوت قائل شد، زیرا هر دو نوع وظیفه این در نوع، با استفاده از سرور پایگاه داده به صورت بهینه قابل دست یابی نخواهند بود. به همین دلیل، موتور پایگاه داده (Database Engine) برای وظایف عملیاتی و خدمات تحلیلی (Analysis Services) برای وظایف تحلیلی مورد استفاده قرار می‌گیرند.

۳-الف) نیازمندی های سخت‌افزاری و شبکه

این مطلب که سیستم SQL Server فقط روی سیستم‌عامل‌های مایکروسافت اجرا می‌شود، تمرکز روی نیازمندی های سخت‌افزاری و شبکه را ساده‌تر می‌کند. مدیر شبکه فقط باید به سخت‌افزار و شبکه‌ی موردنیاز توجه کند.

۱-۳-الف) نیازمندی های سخت‌افزاری

سیستم‌عامل‌های ویندوز روی پلاتفرم‌های سخت‌افزاری اینتل و سازگار با آن پشتیبانی می‌شوند. سرعت پردازنده باید حداقل یک گیگاهرتز (۱ GHZ) باشد.

نکته

به طور کلی، دو گروه ویرایش از SQL Server وجود دارد: ۳۲ بیتی و ۶۴ بیتی و نیازمندی های این دو متفاوت اند. در این جا موارد عمومی مطرح می شود.و برای کسب اطلاعات بیش تر در مورد هر گروه ویرایشی به سایت زیر رجوع کنید:

http://msdn2.microsoft.com/en-us/library/ms143506(SQL.100).aspx

حداقل حافظه‌ی مورد نیاز، ۵۱۲ مگابایت است. ولی تقریباً هر کسی می‌داند که چنین پیکربندی حداقلی به خوبی اجرا نخواهد شد و بهتر است که حدود یک گیگابایت یا بیش‌تر حافظه در اختیار داشته باشیم.

فضای مورد نیاز دیسک سخت به پیکربندی سیستم و برنامه‌های کاربردی‌ای که برای نصب انتخاب می‌کنید،بستگی دارد.

۲-۳-الف) نیازمندی های شبکه

برای اتصال به هر جزئی از SQL Server، باید پروتکل شبکه را که فعال شده است داشته باشید. سیستم SQL Server می‌تواند تقاضای چندین پروتکل را به صورت هم‌زمان پاسخ دهد. اگر برنامه‌ی سرویس گیرنده را به نحوی پیکربندی کنید که به طور متوالی چندین پروتکل را به کار بگیرد، سرویس گیرنده‌ها با استفاده از یک پروتکل به سیستم متصل می‌شوند.

SQL Server برای سرویس گیرنده‌ها،مانند یک سیستم سرویس گیرنده/سرویس دهنده، این امکان را فراهم می‌کند که برای ارتباط با سرور و برعکس از پروتکل‌های مختلف شبکه استفاده کنند. مدیر سیستم در طول نصب اتصال، باید تصمیم بگیرد که کدام پروتکل‌های شبکه (به عنوان کتابخانه) برای دسترسی سرویس گیرنده‌ها به سیستم قابل دسترس خواهند بود. پروتکل‌های شبکه‌ای زیر در سمت سرور می‌توانند انتخاب شوند:

- Shared memory
- (Transmission control protocol/ Internet protocol(TCP/IP
- named pipes

Virtual Interface Adapter(VIA) protocol

Shared memory - اتصال به سیستم از یک سرویس گیرنده با استفاده از پروتکل حافظه اشتراکی اجرا می‌شود. حافظه ی اشتراکی، مشخصه‌ی قابل پیکربندی ندارد و این پروتکل همیشه ابتدا اجرا می‌شود.

Named pipes یک پروتکل شبکه‌ی جای گزین در پلاتفرم‌های ویندوز است. بعد از فرآیند نصب، می‌توان پشتیبانی از این پروتکل را رها کرد و پروتکل شبکه‌ی دیگری را برای ارتباط بین سرور و سرویس گیرنده‌ها به کار برد.

پروتـــکل شـــبکه‌ی TCP/IP به سیســـتم امکان می‌دهـــد تا با اســـتفاده از Windows Sockets اســـتاندارد، بـــا روش IPC(Internet Protocol Communication) از طریق پروتکل TCP/IP ارتباط برقرار کند.

پروتکل VIA(Virtual Interface Adapter) با سخت افزار VIA کار می‌کند.

٤-الف) ویرایش‌های SQL Server

هم زمـــان با این که نصب را برنامه‌ریزی می‌کنید، نیاز داریـــد تا نوع ویرایش‌های موجود SQL Server را بدانید. غیر از گروه‌های ویرایش ۳۲ و ۶۴ بیتی، مایکروســـافت ویرایش‌های زیر را از SQL Server ۲۰۰۸ ارائه می‌کند:

- **ویرایش Express**- نســـخه ی ســـبکی از SQL Server ۲۰۰۸ است. این محصول باید به وســـیله‌ی تولیدکنندگان برنامه‌ی کاربردی مورد اســـتفاده قرار گیرد. به همین دلیل، محصول شامل برنامه‌ی XM(Express Manager) اصلی است و از تجمیع CLR و XML پشتیبانی می‌کند.
- **ویرایش Workgroup**- برای مشـــاغل کوچک طراحی شـــده اســـت و در سطح دپارتمان‌ها نیز مورد اســـتفاده قرار می‌گیرد. این ویرایـــش، از پایگاه داده‌های رابطه‌ای بدون آگاهـــی تجـــاری (BI) و از قابلیت‌هایی با قابلیت دسترســـی بالا (حداکثـــر از دو پردازنده و حداکثر ۲گیگابایت حافظه‌ی RAM) پشتیبانی می‌کند.

- **ویرایش Enterprise**- شـــکل خاصی از سیســـتم SQL Server اســـت که برای برنامه‌های کاربردی با تعداد کاربران زیاد مناســـب اســـت. در مقایســـه با ویرایش استاندارد، این ویرایش ویژگی‌هـــای اضافی ای دارد که می‌تواند برای نصب‌های چندپردازنده‌ای یا چند دیســـکی، مفید باشـــد. مهم ترین ویژگی های اضافی این ویرایش، بخش‌بندی داده‌ها، نسخه فوری پایگاه داده و نگه داری برخط پایگاه داده است.

- **ویرایش Developer**- به برنامه‌نویسان این امکان را می‌دهد که هر نوع برنامه‌ی کاربردی را با SQL Server روی پلاتفرم‌های ۳۲ و ۶۴ بیتی ایجاد و آزمایش کنند. این ویرایش شامل تمام وظایف ویرایش Enterprise است ولی فقط برای استفاده در تکوین، آزمایش و تشریح تائید شده است. هر License از این ویرایش برای یک برنامه‌نویسی نرم افزار روی چندین سیستم استفاده می‌شود و برنامه‌نویسان دیگر می‌توانند با خریدن License اضافی از نرم‌افزار استفاده کنند.
- **ویرایش Compact**- نسخه‌ی کوچکی است از SQL Server که می‌توان آن را روی Pocket PCs، تلفن‌های هوشمند، Tablet PCs و رایانه‌های شخصی رومیزی مورد استفاده قرار داد. پایگاه داده‌های Compact Edition، معمولاً به عنوان پایگاه داده‌های افزوده شده برای برنامه‌های کاربردی به کار می‌روند.

۵-الف) نیازمندی های نصب

در طول فرآیند نصب، باید چند گزینه را انتخاب کنید. توصیه می‌کنیم که بهتر است با این عوامل مؤثر قبل از اجرای برنامه‌ی Setup، آشنا شوید. باید قبل از شروع فرآیند نصب، به پرسش‌های زیر پاسخ دهید:

- فهرست ریشه در کجا ذخیره خواهد شد؟
- چند نمونه از Database Engine مورد استفاده قرار خواهد گرفت؟
- چه نوع شناسایی برای Database Engine مورد استفاده قرار خواهد گرفت؟

نکته

قبل از شروع فرآیند نصب، باید دقیقاً بدانید که کدام یک از مؤلفه های SQL Server ۲۰۰۸ را می‌خواهید نصب کنید.

۱-۵-الف) محل ذخیره‌ی فهرست ریشه

فهرست ریشه محلی است که برنامه‌ی Setup تمام فایل‌های برنامه را، که هنگام استفاده از سیستم SQL Server تغییر نمی‌کنند، ذخیره می‌کند. فرآیند نصب، به طور پیش‌فرض، تمام فایل‌ها را در زیرفهرست Microsoft SQL Server ذخیره می‌کند. اگر چه می‌توان در طول عمل نصب، این تنظیمات را تغییر داد. استفاده از نام پیش فرض توصیه می‌شود، زیرا نسخه‌ی سیستم را به طور منحصر به فردی تعیین می‌کند.

۲-۵-الف) چه تعداد نمونه از Database Engine نصب می‌شود

همراه Database Engine می‌توان چندین نمونه‌ی مختلف را نصب کرد و به کار برد. یک نمونه، سرور پایگاه داده است که سیستم، آن را به اشتراک نمی‌گذارد و پایگاه داده‌های کاربر به همراه نمونه‌های (سرورهای) دیگر روی همان کامپیوتر اجرا می‌شوند.

دو نوع نمونه وجود دارد که عبارت‌اند از:

- Default
- Named

نمونه‌ی پیش فرض از سرور پایگاه داده، در نسخه‌های قبلی عمل می‌کند و فقط سرور پایگاه داده بدون پشتیبانی نمونه وجود دارد. نام کامپیوتری که نمونه در آن اجرا می‌شود نام نمونه‌ی پیش‌فرض را به تنهایی تعیین می‌کند. هر نمونه‌ای از سرور پایگاه داده به غیر از نمونه‌ی پیش فرض را نمونه‌ی نام گذاری شده می‌نامند.

برای شناسایی یک نمونه‌ی نام گذاری شده، باید نام آن را بر اساس نام کامپیوتری که نمونه روی آن اجرا می‌شود، تعیین کنید:برای مثال،INSTANCE۱۱\NTB۱۱۹۰۱. روی یک کامپیوتر، می‌تواند چندین نمونه‌ی نام گذاری شده (علاوه بر نمونه‌ی پیش فرض) داشته باشد. به علاوه، می‌توان نمونه‌های نام گذاری شده روی یک کامپیوتر را، که نمونه‌ی پیش

فرض ندارند پیکربندی کرد.

اگر چه تمام نمونه‌های اجرا شده ی روی یک کامپیوتر، اغلب منابع سیستم را به اشتراک نمی‌گذارند، ولی اجزائی وجود دارند که از طریق آن ها به اشتراک گذاشته می‌شوند:

- SQL Server Program Group
- Analysis Services Server
- Development Libraries

وجـــود فقط یک گروه برنامـــه‌ی SQL Server روی یک کامپیوتر، به این معنی که فقط یـــک کپـــی از هر برنامه‌ی ســـودمند وجود دارد که به وســـیله‌ی یک آیکـــن در گروه برنامه نمایش داده می‌شود. بنابراین، هر برنامه‌ی سودمندی با تمام نمونه‌های پیکربندی شده روی کامپیوتر کار می‌کند.

در شرایط زیر، باید استفاده از چندین نمونه را در نظر بگیرید:

- انواع مختلفی از پایگاه داده‌ها را روی کامپیوترتان دارید.
- کامپیوترتان به اندازه‌ی کافی قوی است تا چندین نمونه را مدیریت کند.

هدف اصلی داشـــتن چندین نمونه، تقســـیم پایگاه داده‌هایی اســـت که در سازمانتان در گروه‌های متفاوت وجود دارند. برای مثال، اگر سیســـتم، پایگاه داده‌هایی را که به وســـیله‌ی کاربران مختلف استفاده می‌شوند مدیریت می‌کند (پایگاه داده‌های تولید، آزمایش و...)، باید آن ها را برای راه‌اندازی تحت نمونه‌های مختلف تقسیم کنید.

این روشی است که می‌توانید پایگاه داده‌های تولید را از پایگاه داده‌هایی که به وسیله‌ی کاربران حرفه‌ای و مبتدی استفاده می‌شوند، مجزا کنید. یک کامپیوتر تک پردازنده، پلاتفرم ســـخت‌افزاری مناسبی برای اجرای چندین نمونه از Database Engine نخواهد بود، زیرا منابع محدود اســـت. به همین دلیل، باید به کارگیری چندین نمونه را فقط با کامپیوترهای

چند پردازنده در نظر بگیرید.

۳-۵-الف) کدام حالت شناسایی انتخاب شود؟

در ارتباط با Database Engine، دو حالت تائید مختلف وجود دارد:

- **حالت Windows**، که امنیت را به طور واضح در سطح سیستم عامل تعیین می‌کند (این همان روش ورود کاربران به سیستم عامل ویندوز با استفاده از حساب کاربری و عضویت گروه است).
- **حالت Mixed**، که به کاربران امکان اتصال به Database Engine (با استفاده از شناسایی ویندوز یا SQL Server)را می‌دهد.به این معنی که بعضی از حساب‌های کاربری می‌توانند برای استفاده از زیرسیستم امنیتی ویندوز تنظیم شوند، در حالی که کاربران دیگر می‌توانند، علاوه بر ویندوز، از زیرسیستم امنیتی SQL Server استفاده کنند.

مایکروسافت حالت Windows را توصیه می‌کند.

۶-الف) نصب SQL Server

اگر قبلاً برنامه‌ی نرم‌افزار پیچیده ای را نصب کرده باشید، احتمالاً می‌دانید که در نصب اولین بار، احساس راحتی نخواهید داشت. این احساس نگرانی ناشی از پیچیدگی نرم‌افزار نصب شده و گوناگونی پرسش‌های مطرح شده در طول عمل نصب است و این که ممکن است به طور کامل با نرم‌افزار آشنا نباشید و به پرسش‌های نصب، پاسخ دقیقی ارائه ندهید. این بخش، به شما کمک می‌کند تا روش نصب و پاسخ پرسش‌ها را به دست آورید.

برای شروع نصب، دیسک DVD نرم‌افزار SQL Server ۲۰۰۸ را در درایو DVD قرار دهید. ویزارد Install Shield را اجرا کنید.در این صورت، به شما اعلان می‌شود که محل ذخیره‌ی فایل‌های باز شده را تعیین کنید. هنگامی که روی Next کلیک کنید، Install Shield تمام فایل‌های ضروری را از DVD باز و وظایف آن را کامل می‌کند.

۱-۶-الف) پیش از شروع برنامه‌ی نصب

چندیـــن عمل وجود دارد که باید آن ها را قبل از شـــروع برنامـــه‌ی Setup اجرا و کامل کنید. بهترین مکان برای شروع، خواندن Release Notes است. این فایل، شامل جدیدترین اطلاعاتی است که می‌تواند مورد استفاده کاربر قرار گیرد.

سایت مایکروسافت، منبع دیگری برای کسب اطلاعات بیش تر است. در این وب گاه(وب سایت) می‌توانید مستندات مورد علاقه را جست جو و دانلود کنید.

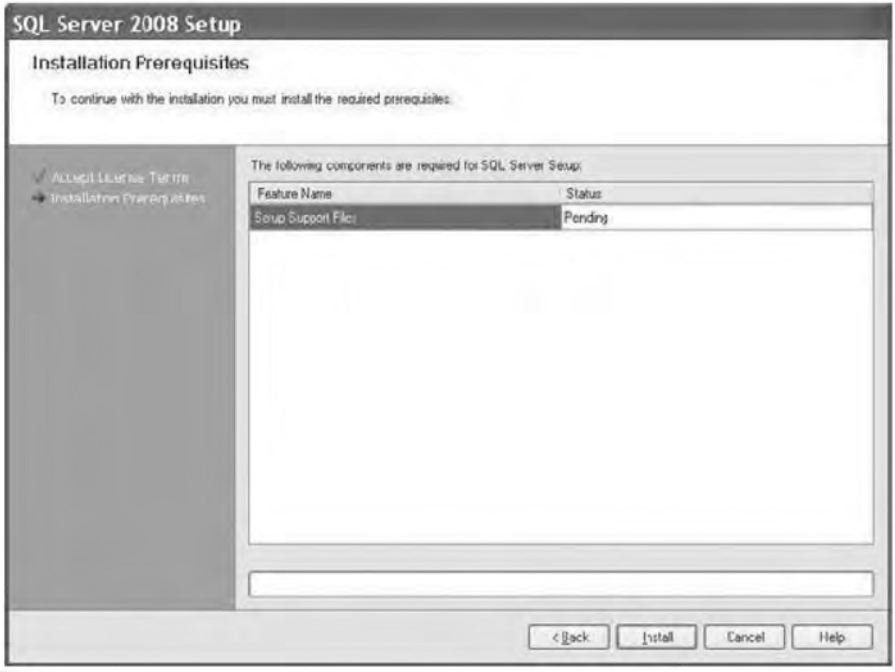
۲-۶-الف) شروع برنامه‌ی نصب و نصب مقدمات

در صفحـــه‌ی Installation Prerequisites (شـــکل ۱-الـــف)، SQL Server ۲۰۰۸ Setup قبل از شروع نصب، تعیین می‌کند که کدام مؤلفه‌ی نرم‌افزاری مورد نیاز است. برای شروع عملیات، روی Install کلیک کنید.

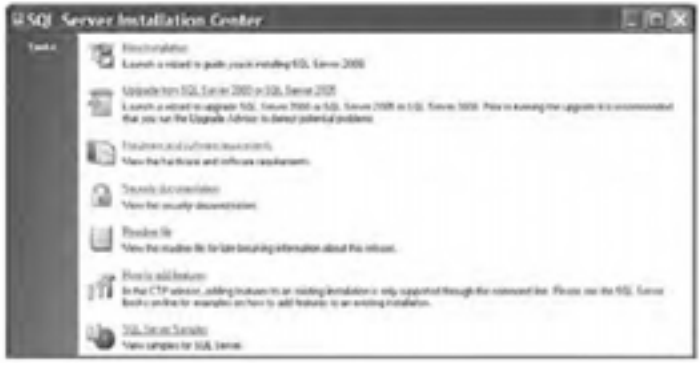
۳-۶-الف) نصب مؤلفه‌های SQL Server

بعـــد از نصب موفق مقدمـــات، SQL Server Installation Center فرآیند نصب را به طور خودکار، با وظایفی که می‌توانند اجرا شـــوند، نشـــان می‌دهد (شکل ۲-الف). برای نصب SQL Server ۲۰۰۸، گزینـــه‌ی New Installation را نصب کنید. این مرحله ویزاردی را اجرا می‌کند که شما را در طول فرآیند نصب راهنمایی می‌کند.

در صفحـــه اول ویزارد، صفحـــه‌ی System Configuration Check (شـــکل۳-الف)، برنامه‌ی نصب، کامپیوترتان را برای وضعیت‌هایی که ممکن است فرآیند نصب آن ها متوقف شـــود، پویش می‌کند. در صورتی که روی دکمـــه‌ی Show Details کلیک کنید، می‌توانید تمام جزئیات فرآیند بررســـی را مشاهده کنید. برای ادامه ی نصب، روی Next کلیک کنید. در صفحـــه‌ی Registration Information، اطلاعـــات را در کادرهای مربوطه وارد و روی Next کلیک کنید.



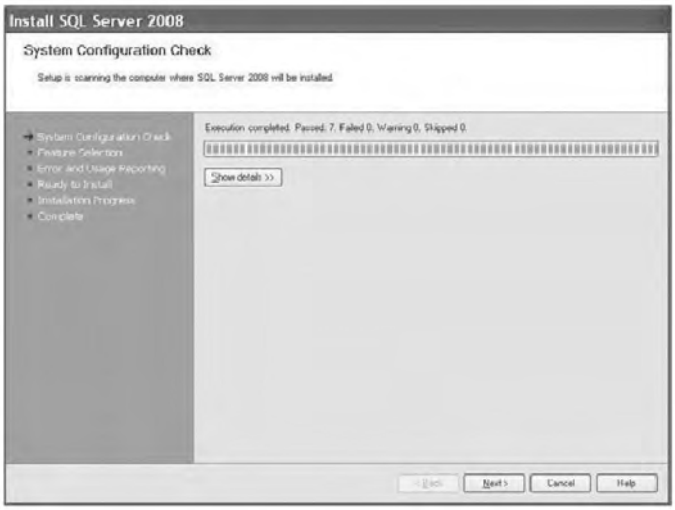
شکل ۱-الف) صفحه‌ی مقدمات نصب



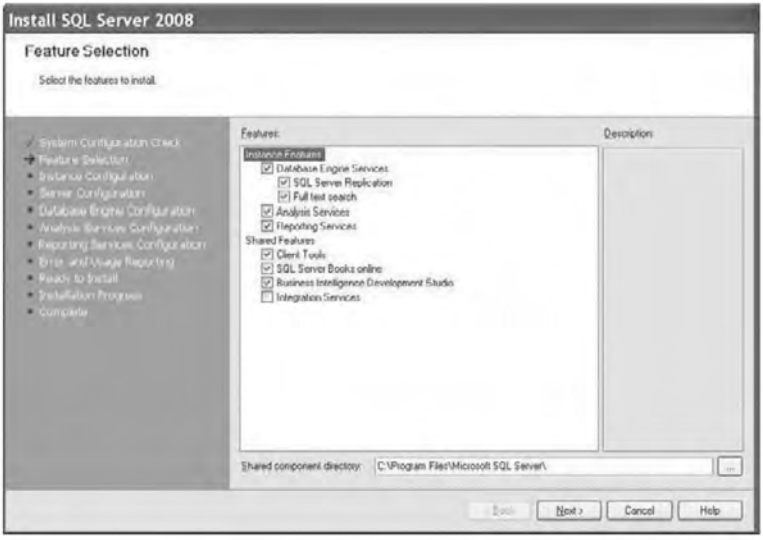
شکل ۲-الف) SQL Server Installation

در صفحـــه‌ی Feature Installation (شـــکل ۴-الف)، با برگزیـــدن کادرهای علامت، مؤلفه‌هایی را برای نصب، انتخاب کنید. هم‌چنین، در پایین صفحه می‌توان فهرستی را برای

ذخیره‌ی مؤلفه‌های اشتراکی تعیین کرد. برای ادامه، روی Next کلیک کنید.



شکل ۳-الف) صفحه‌ی System Configuration Check



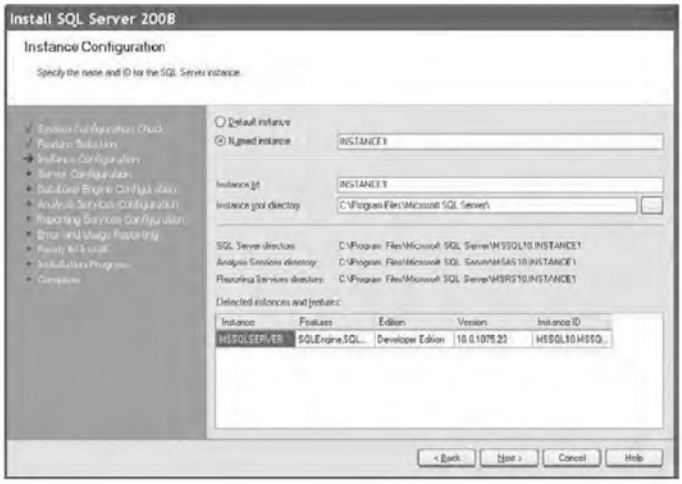
شکل ۴-الف) صفحه‌ی Feature Selection

نکته

مؤلفه‌های SQL Server که انتخاب شـدند، یکی بعد از دیگری، به ترتیبی که در صفحه‌ی Feature Selection لیسـت شده‌اند، نصب خواهند شد. فقط مؤلفه‌های انتخاب شده نصب می‌شوند.

در صفحـه‌ی Instance Configuration (شـکل ۵-الـف)، می‌تـوان نصـب نمونه‌ی پیش‌فرض یا نام گذاری شـده را انتخاب کرد. برای نصب نمونه‌ی پیش‌فرض، روی Default Instance کلیک کنید. اگر نمونه‌ی پیش فرض قبلاً نصب شده باشد و این گزینه را انتخاب کنید، برنامه‌ی نصب آن را به هنگام می‌کند و علاوه بر آن، امکان نصب مؤلفه‌های اضافی را هم فراهم می‌کند.

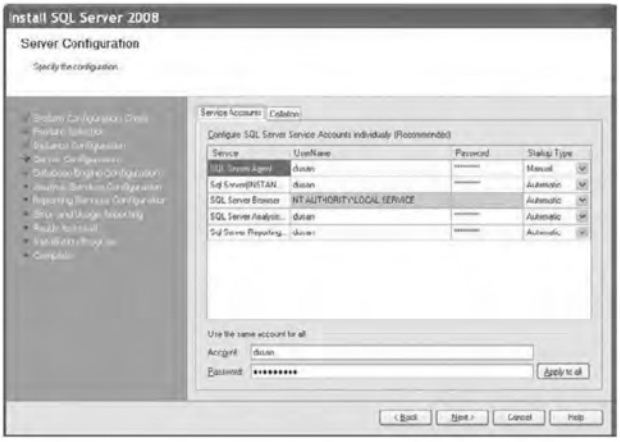
برای نصب یک نمونه‌ی نام گذاری شده‌ی جدید، روی Named Instance کلیک نمائید و نام جدیدی را در کادر متن تایپ کنید. در بخش پایین صفحه می‌توان لیستی از نمونه‌هایی را که قبلاً روی سیستم نصب شده‌اند مشاهده کرد. همان طوری که در شکل ۵-الف مشاهده می‌کنیـد، نمونـه‌ای به نام INSTANCE ۱ را،که قبلاً نمونـه‌ی پیش فرض روی آن نصب شـده اسـت، نصب می‌کنیم(MSSQLSERVER نام نمونه‌ی پیش‌فرض برای Database Engine است). برای ادامه روی Next کلیک کنید.



شکل ۵-الف) صفحه‌ی Instance Configuration

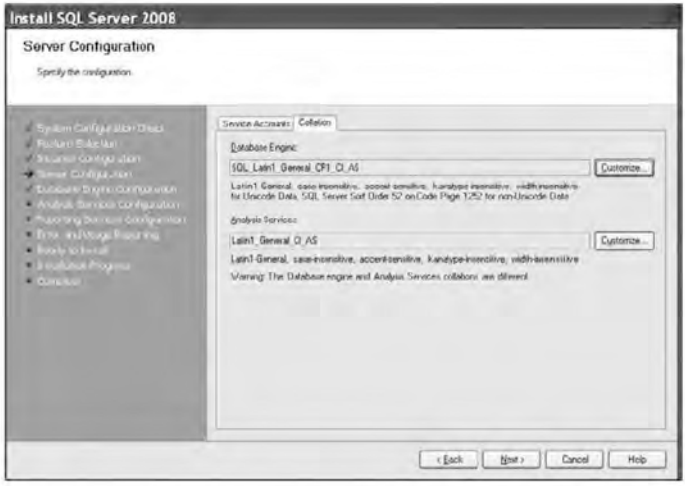
صفحه‌ی بعد (شکل ۶-الف)، امکان تعیین نام‌های کاربری و گذرواژه های مربوطه را برای خدمات مؤلفه‌هایی که در فرآیند نصب، نصب خواهند شد فراهم می‌کند(می‌توان یک حساب کاربری را برای تمام سرویس‌ها اعمال کرد).

برای انتخاب تطبیق نمونه‌تان، روی زبانه‌ی Collation از صفحه‌ی Server Configuration کلیک کنید (شکل ۷-الف)(تطبیق، رفتار مرتب‌سازی برای نمونه‌تان را تعریف می‌کند). می‌توان تطبیق‌های پیش فرضی را برای مؤلفه‌هایی که نصب خواهند شد انتخاب کرد یا برای انتخاب سایر مقایسه‌هایی که به وسیله‌ی سیستم پشتیبانی می‌شوند، روی Customize کلیک نمود. برای ادامه روی Next کلیک کنید.

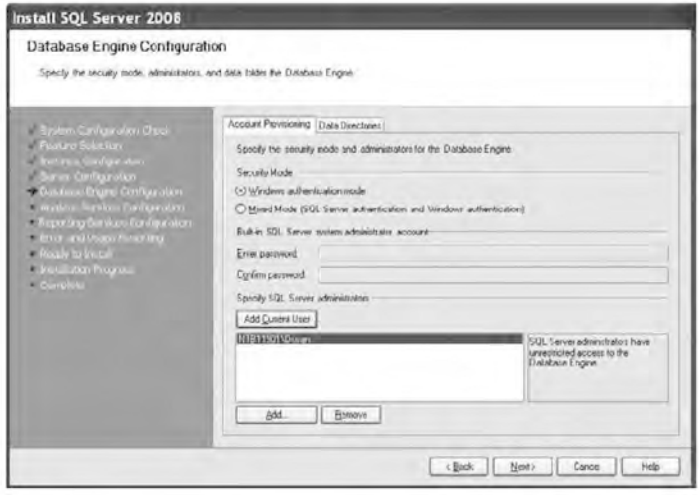


شکل ۶-الف) Server Configuration (زبانه‌ی Service Accounts)

در صفحه‌ی Database Engine Configuration (شـکل ۸-الف)، حالت شناسـایی را برای سیسـتم Database Engine انتخاب کنید. در صورتی که می‌خواهید یک یا چند کاربری را که دسترسـی نامحدود بـه نمونه‌ی Database Engine خواهند داشـت اضافه کنید، روی Add Current User کلیک نمائید.

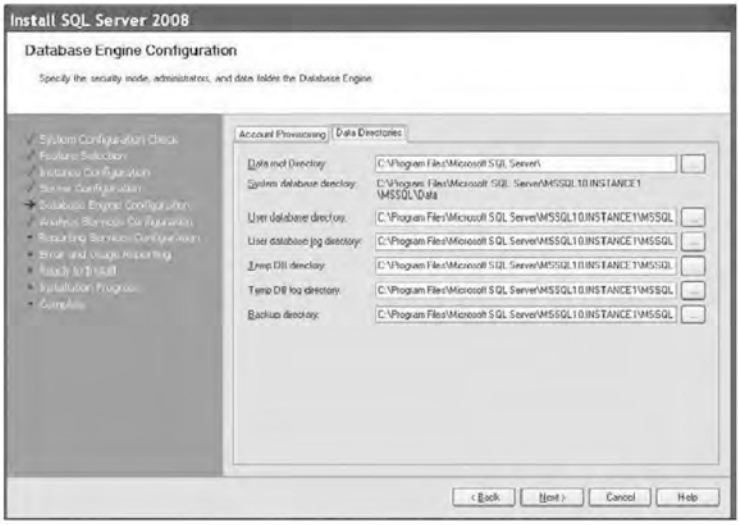


شکل ۷-الف) Server Configuration (زبانه‌ی Collation)



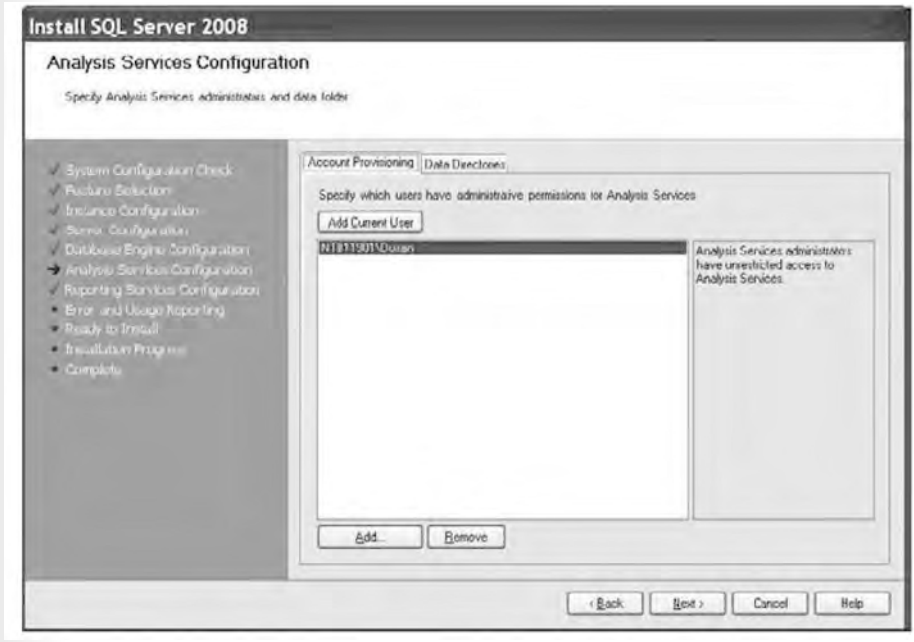
شکل ۸-الف) Database Engine Configuration
(زبانه‌ی Account Provisioning)

زبانه‌ی Data Directors (شکل ۹-الف) امکان تعیین تمام فهرست‌هایی را که فایل‌های مربوط به Database Engine در آن‌ها ذخیره خواهند شـد فراهم می‌کند(ویژگی جدید SQL Server ۲۰۰۸ این اسـت که مکان ذخیره‌سازی پایگاه داده‌ی سیستمی tempdb در طول نصب تعیین می‌شود). برای ادامه روی Next کلیک کنید.



شکل ۹-الف) Database Engine Configuration (زبانه‌ی Data Directors)

آن چه در مرحله‌ی بعد ظاهر می‌شـود، به انتخاب نصب Reporting Services بستگی دارد.اگـر تعیین کرده‌ایـد کـه Reporting Services نصب شـود،صفحه‌ی Reporting Services Configuration ظاهـر می‌شـود. در این مرحله، فقـط می‌توان تصمیم گرفت که سـرور گزارش(بدون پیکربندی)نصب شـود یـا آن را نصب و پیکربندی نمود. سـومین جایگزین، تجمیع سـرور گزارش بـا نرم‌افـزار Microsoft Office Share point Server اسـت(نرم افزاری که برای همکاری ساده مورد استفاده قرار می گیرد و ویژگی های مدیریت محتـوا و پیاده‌سـازی عملیات تجاری را انجام می دهد). بعـد از آن، برای ادامه روی Next کلیک کنید.



شکل ۱۰-الف) صفحه‌ی Analysis Services Configuration

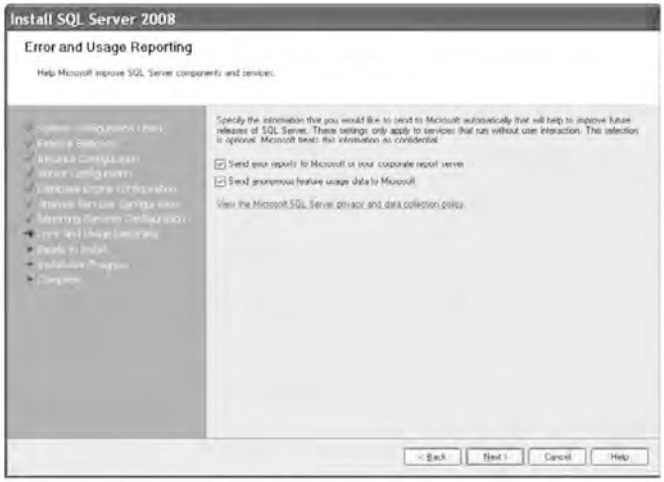
اگر می خواهید در Error and Usage Reporting (شـکل ۱۱-الف)،اطلاعات خطاها به طور خودکار به مایکروسافت ارسال شود، آن‌ها را تعیین کنید. اگر این بخش گزارش خطا را نمی‌خواهید، هر دو کادر علامت را پاک کنید. روی Next کلیک کنید.

آخریـن صفحه (قبل از شـروع نصب واقعی)، صفحه‌ی Ready To Install اسـت. این صفحـه، امکان بازنگـری خلاصه‌ی تمام مؤلفه‌های SQL Server را که نصب خواهند شـد ارائه می‌کند. برای شروع عملیات نصب، روی Install کلیک کنید. همان طوری که در شکل ۱۲-الف مشاهده می‌کنید، برنامه‌ی Setup امکان مشاهده ی فرآیند نصب را فراهم می‌کند. در صورتی که فرآیند نصب با موفقیت به پایان رسید، روی Next کلیک کنید.

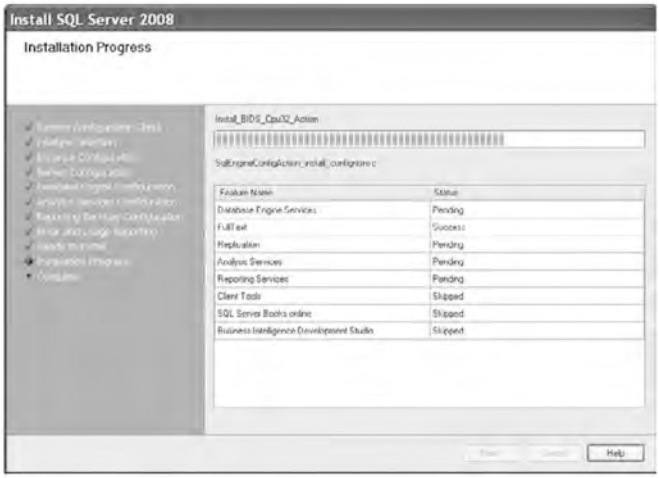
صفحه‌ی Complete ظاهر می‌شـود و محل فایـل Log را نمایش می‌دهد. برای کامل شدن فرآیند نصب، روی Close کلیک کنید. بعد از آن می‌توانید از تمام مؤلفه‌هایی که نصب

شده‌اند، استفاده کنید.

بخش بعدی، به کارگیری نمونه ای از Database Engine را شرح می‌دهد.



شکل ۱۱-الف) صفحه‌ی Error And Usage Reporting



شکل ۱۲-الف) صفحه‌ی Installation Progress

پیوست ب

توابع سیستمی SQL Server

۱-ب) توابع عددی

توابــع عددی درون Transact-SQL توابع ریاضی هســتند که برای کارکردن با مقادیر عددی به کار می‌روند. توابع عددی زیر، قابل دسترس هستند:

تابع	شرح
ABS(n)	**قدر مطلق یک مقدار را بر می‌گرداند. برای مثال:** SELECTABS(-5.767)=5.767,SELECT ABS(6.384)=6.384
ACOS(n)	**آرک کسینوس n را بر می‌گرداند. n و نتیجه، یک داده‌ی FLOAT هستند.**
ASIN(n)	**آرک سینوس n را بر می‌گرداند. n و نتیجه، یک داده‌ی FLOAT هستند.**
ATAN(n)	**آرک تانژانت n را بر می‌گرداند. n و نتیجه، یک داده‌ی FLOAT هستند.**
ATAN2(n,m)	**آرک تانژانت n/m را بر می‌گرداند. n و m و نتیجه، یک داده‌ی Float هستند.**
CEILING(n)	**کوچک ترین عدد صحیح بزرگ تر یا مساوی با پارامتر تعیین شده را بر می‌گرداند. برای مثال:** Select CEILING(4.88)=5 Select CEILING(-4.88)=-4
COS(n)	**کسینوس n را محاسبه می‌کند. n و نتیجه‌ی عبارت از نوع داده‌ی FLOAT هستند.**

COT(n)	**کتانژانت n را محاسبه می‌کند. n و نتیجه‌ی عبارت از نوع داده‌ی FLOAT هستند.**
DEGREES(n)	**رادیان را به درجه تبدیل می‌کند. مثال‌ها:** Select Degrees(PI()/۹۰,۰ =(۲ Select Degrees(۴۲,۹۷ =(۰,۷۵
EXP(n)	**مقدار en را محاسبه می‌کند. مثال: SELECT EXP(۱) = ۲,۷۱۸۳**
FLOOR(n)	**بزرگترین عدد صحیح کوچک تر یا مساوی با n را بر می‌گرداند. مثال:** Select Floor(4.88)= 4
LOG(n)	**لگاریتم طبیعی n را محاسبه می‌کند. مثال‌ها:** Select LOG(4.67)= 1.54 Select LOG(0.12)= -2.12
LOG۱۰(n)	**لگاریتم مبنای ده عدد n را محاسبه می‌کند. مثال‌ها:** Select LOG10(4.67)= 0.67 Select LOG10(0.12)= -0.92
PI()	**مقدار عدد پی(۳/۱۴) را بر می‌گرداند.**
POWER(x,y)	**مقدار xy را محاسبه می‌کند. مثال‌ها:** Select Power(3.12,5)= 295.65 Select Power(81,0 5)= 9
RAND	**یک عدد تصادفی بین ۰و۱ از نوع داده‌ی FLOAT را بر می‌گرداند.**
ROUND (n,p,[t])	**مقدار n را با استفاده از دقت P گرد می‌کند. مقادیر مثبت P سمت راست نقطه‌ی اعشار و مقادیر منفی سمت چپ نقطه‌ی اعشار را گرد می‌کنند. پارامتر t اختیاری است و سبب می‌شود n کاهش یابد. مثال‌ها:** Select Round(5.4567,3)=5.4570 Select Round(345.4567,-1)= 350.0000 Select Round(345.4567,-1,1)=340.0000
ROW COUNT_BIG	**تعداد سطرهایی را که به وسیله‌ی اجرای آخرین دستور SQL تحت تأثیر قرار گرفته‌اند برمی‌گرداند. مقدار بازگشتی این تابع دارای نوع داده‌ی BIGINT است.**
SIGN(n)	**علامت مقدار n را به صورت یک عدد بر می‌گرداند (۱+ برای مثبت، ۱- برای منفی و ۰ برای صفر).مثال:** Select SIGN(0.88)= 1
SIN(n)	**سینوس n را محاسبه می‌کند. n و مقدار حاصله از نوع داده‌ی FLOAT هستند.**

SQRT(n)	**ریشه‌ی دوم n را محاسبه می‌کند. مثال:** Select SQRT(9)= 3
SQUARE(n)	**مربع n را بر می‌گرداند. مثال:** Select SQUARE(9)= 81
TAN(n)	**تانژانت n را محاسبه می‌کند. n و مقدار حاصله از نوع داده‌ی FLOAT هستند.**

۲-ب) توابع تاریخی

توابــع تاریخی بخش تاریخ یا زمان یک عبارت را محاســبه می‌کنند یا مقداری را از یک شمارش گر زمان بر می‌گردانند. Transact-SQL از توابع تاریخی زیر، پشتیبانی می‌کند:

تابع	شرح
GETDATE()	**تاریخ و زمان جاری سیستم را برمی‌گرداند. مثال:** Select GETDATE( )= ۱۳:۰۳:۳۱,۳۹۰ ۰۱-۰۱-۲۰۰۹
DATEPART (item,date)	**بخش تعیین شده‌ی item از تاریخ date را به صورت یک عدد صحیح بر می‌گرداند. مثال‌ها:** Select Datapart(month,'۱')۱=('۰۱,۰۱,۲۰۰۵=January) Select Datapart(weekday,'۷')۷=('۰۱,۰۱,۲۰۰۵=Sunday)
DATENAME (item,date)	**بخش تعیین شده‌ی item از تاریخ date را به صورت یک رشته‌ی کاراکتری بر می‌گرداند. مثال:** Select DATENAME(weekday,'01.01.2005')= Saturday
DATEDIFF **(dat۲,(item,dat۱**	**اختلاف بین دو تاریخ dat۱ و dat۲ را محاسبه و نتیجه را به صورت یک عدد صحیح در واحد تعیین شده بر اساس مقدار item برمی‌گرداند.** **مثال:** Select DataDiff(year,BirthDate,GETDATE()) As age from employee; **سن هر کدام از کارمندان را بر می‌گرداند.**
(DATEADD(i,n,d	**تعداد n واحد بر حسب i را به تاریخ d اضافه می‌کند. مثال:** Select Dataadd(DAY,3,HirreDate) As age from employee; **سه روز به تاریخ استخدام هر کارمندی اضافه می‌کند (پایگاه داده‌ی sample را مشاهده کنید).**

۳-ب) توابع رشته‌ای

توابع رشـــته‌ای برای کارکردن با مقادیر داده‌ای یک ســـتون که از نوع داده‌ی کاراکتری هستند، به کار می‌روند. Transact-SQL از توابع رشته‌ای زیر پشتیبانی می‌کند:

تابع	شرح
ASCII(character)	**کاراکتر تعیین شده را به کد عددی اسکی معادل، تبدیل می‌کند. عدد صحیحی را برمی‌گرداند. مثال:** SELECT ASCII('A')= 65
CHAR(integer)	**کد اسکی را به کاراکتر معادل تبدیل می‌کند. مثال:** SELECT CHAR(65)= 'A'
CHARINDEX(Z1,Z2)	**محل شروع رشته‌ی Z1 در رشته‌ی Z2 را بر می‌گرداند. اگر Z1 درون Z2 نباشد، صفر را برمی‌گرداند. مثال:** SELECT CHARINDEX('bl','table')= 3
DIFFERENCE(Z۱,Z۲)	**یک عدد صحیح بین ۰تا۴ را بر می‌گرداند که تفاوت مقادیر SOUNDEX دورشته‌ی Z1 و Z2 است(SOUNDEX عددی را برمی‌گرداند که صدای رشته است). با این روش، رشته‌هایی با صداهای مشابه می‌توانند تعیین شوند. مثال:** SELECT DIFFERENCE('spelling', 'telling')= 2
LEFT(Z,length)	**اولین کاراکترهای length را از رشته‌ی Z برمی‌گرداند.**
LEN(Z)	**تعداد کاراکترها را بر می‌گرداند.**
LOWER(Z۱)	**تمام حروف بزرگ رشته‌ی Z۱ را به حروف کوچک تبدیل می‌کند. حروف کوچک و اعداد و سایر کاراکترها را تغییر نمی‌دهد.مثال:** SELECT LOWER('BIG')= 'big'
LTRIM(Z)	**فضاهای خالی را از سمت چپ رشته‌ی Z حذف می‌کند. مثال:** SELECT LTRIM(' string')= 'string'
NCHAR(i)	**کاراکتر یونیکد عدد i را بر می‌گرداند.**

QUOTENAME (char_string)	**یک رشته‌ی یونیکد را به همراه حائل‌های اضافه شده بر می‌گرداند.**
PATINDEX(٪P٪ , expr)	**محل شروع اولین مورد از الگوی P در عبارت expr را بر می‌گرداند. اگر پیدا نشود، صفر را بر می‌گرداند. مثال:** 1) Select PATINDEX('%gs%','longstring')=4; 2)Selectright(CantactName,LEN (ContactName) -PATINDEX('٪٪', ContactName))AS First_name FROM Customers; **دومین پرس و جو، اسامی تمام مشتریان را از ستون Customers برمی‌گرداند.**
REPLACE(str1,str2,str3)	**تمام موارد str۲ در str۱ را با str۳ جای گزین می‌کند. مثال:** Select REPLACE('shave','s','be')= behave
REPLICATE(z,i)	**رشته‌ی z را i بار تکرار می‌کند. مثال:** Select REPLICATE('a',10)= 'aaaaaaaaaa'
REVERSE(Z)	**رشته‌ی z را معکوس می‌کند. مثال:** Select REVERSE('calculate')= 'etaluclac'
(RIGHT(Z,length	**به تعداد length کاراکتر از سمت راست رشته را جدا می‌کند و برمی‌گرداند. مثال:** SELECT RIGHT('Notebook',4)= 'book'
RTRIM(Z)	**فضاهای خالی در سمت راست رشته‌ی z را حذف می‌کند. مثال:** SELECT RTRIM('Notebook ')= 'Notebook'
SOUND EX(a)	**یک کد SOUNDEX چهار کاراکتری را برای تعیین شباهت بین دو رشته برمی‌گرداند. مثال:** SELECT SOUNDEX('spelling')= s145

SPACE(length)	**یک رشته از فضای خالی به طول length را برمی‌گرداند.** **مثال:** SELECT SPACE=’ ‘
STR(f,[len[,d]])	**عدد اعشاری f را به رشته تبدیل می‌کند.LEN طول رشته و d تعداد رقم‌های اعشار است. مثال:** SELECT STR(3.45678,4,2)= ‘3.46’
STUFF(Z1,a,length,Z2)	**از محل a رشته‌ی Z۱ شروع و رشته‌ی Z۲ را به طول length جای گزین می‌کند. مثال‌ها:** Select STUFF(‘Notebook’,5,0,’in a’)= ‘Note in a book’ Select STUFF(‘Notebook’,1,4,’Hand’) = ‘Handbook’
SUBSTRING(z,a,length)	**یک رشته‌ای از رشته‌ی z را به طول length که از مکان a شروع می‌شود، جدا می کند و برمی‌گرداند. مثال:** SELECT SUBSTRING(‘wardrobe’,1,4)= ‘ward’
UNICODE	**عدد صحیحی را که در استاندارد Unicode تعریف شده است را برای اولین کاراکتر عبارت ورودی برمی‌گرداند.**
UPPER(Z)	**تمام حروف کوچک رشته‌ی z را به حروف بزرگ تبدیل می‌کند. حروف بزرگ و اعداد تغییر نمی‌کنند. مثال:** SELECT UPPER(‘lower’)= ‘LOWER’

٤-ب) توابع سیستمی

توابع سیستمی SQL اطلاعات گسترده‌ای درباره شیءهای پایگاه داده ارائه می‌کنند. اغلب این توابع از یک شناسه‌ی عددی داخلی(ID) که زمان ایجاد آن به وسیله‌ی سیستم، برای هر شیء پایگاه داده تعیین می‌شود، استفاده می‌کنند. با استفاده از این شناسه، سیستم می‌تواند به طور منحصر به فرد هر شیء پایگاه داده را شناسایی کند. توابع سیستمی،

اطلاعاتی درباره‌ی سیستم پایگاه داده ارائه می‌کنند. جدول زیر، چند تابع سیستمی را شرح می‌دهد.

تابع	شرح
CAST(a AS type[length])	**عبارت a را به نوع داده‌ی type (در صورت امکان) تبدیل می‌کند. a هر عبارت معتبری می‌تواند باشد. مثال:** **SELECT CAST(۳۰۰۰۰۰۰۰۰۰ AS BIGINT) = ۳۰۰۰۰۰۰۰۰۰**
COALESCE(a1,a2,…)	**مقدار اولین عبارت را که NULL نیست از لیست ... و a۲ و a۱ بر می‌گرداند.**
COL_LENGTH(obj,col)	**اندازه‌ی ستون col مربوط به شیء پایگاه داده‌ی obj (جدول یا دیدگاه) را برمی‌گرداند. مثال:** Select COL_LENGTH (‘customers’,’cust_ID’) = 10
CONVERT(type[(length)],a)	**معادل CAST است، ولی آرگومان‌ها به طور متفاوت تعیین می‌شوند. این تابع با هر نوع داده‌ای مورد استفاده قرار می‌گیرد.**
CURRENT-TIMESTAMP	**تاریخ و زمان جاری را بر می‌گرداند. مثال:** Select CURRENT_TIMESTAMP=’200801-01-17:22:55 670’
CURRENT_USER	**نام کاربر جاری را بر می‌گرداند.**
DATALENGTH(z)	**طول نتیجه‌ی حاصل از z را، بر حسب بایت، محاسبه می‌کند. مثال:** Select DATALENGTH(ProductName) from Products **این پرس‌وجو طول هر فیلدی را بر می‌گرداند.**

GETANSINULL(<dbname>)	**اگر مقادیر NULL در پایگاه داده‌ی dbname مطابق استاندارد ANSI SQL باشد، مقدار۱ را برمی‌گرداند. مثال:** Select GETANSINULL (‘AdventureWorks’) = 1
ISNULL(expr,value)	**اگر value برابر با Null نباشد، مقدار expr را برمی‌گرداند. در غیر این صورت، value را برمی‌گرداند.**
ISNUMERIC(expression)	**تعیین می‌کند که آیا عبارت یک نوع عددی معتبر است یا نه.**
NEWID( )	**یک شماره ID منحصر به فرد ایجاد می‌کند که شامل رشته‌ی دودویی ۱۶بیتی برای ذخیره‌ی مقادیر نوع داده‌ی UNIQUEIDENTIFIER است.**
NEWSEQUENTIALID( )	**یک GUID ایجاد می‌کند که از GUID قبلی تولید شده ی قبلی بزرگ تر و به وسیله‌ی این تابع روی کامپیوتر خاص است(این تابع فقط می‌تواند به صورت مقدار پیش فرض یک ستون استفاده شود).**
NULLIF(expr1,expr2)	**اگر عبارت‌های expr۱ و expr۲ معادل هم باشند، مقدار NULL را برمی‌گرداند. مثال:** Select NULLIF(project-no,’p1’) from Projects
SERVERPROPERTY (property name)	**اطلاعاتی درباره‌ی سرور پایگاه داده را برمی‌گرداند.**
SYSTEM-USER	**Login ID کاربر جاری را بر می‌گرداند. مثال:** Select system-USER= LTB13942\dusan
USER_ID([user_name])	**شناسه‌ی کاربر user_name را برمی‌گرداند. اگر هیچ نامی تعیین نشود، شناسه‌ی کاربر جاری بازیابی می‌شود. مثال:** SELECT USER_ID(‘guest’)= 2
USER_NAME([id])	**نام کاربری با شناسه‌ی id را برمی‌گرداند. اگر هیچ شناسه‌ای تعیین نشود، نام کاربر جاری بازیابی می‌شود. مثال:** SELECT USER_NAME= ‘guest’

۵-ب) توابع Metadata

بــه طور کلی این توابع، اطلاعات مربوط به پایگاه داده و شــیءهای آن را برمی‌گردانند.

جدول زیر، چند تابع را شرح می‌دهد.

شرح	تابع
نام ستونی با شناسه‌ی tab_id و ستونی با شناسه‌ی col_id را بر می‌گرداند. مثال: Select col_nameE(OBJECT_ID (‘employee’),3) = ‘emp_lname’	COL_NAME(tab_id,col_id)
اطلاعاتی را درباره‌ی ستون تعیین شده برمی‌گرداند. مثال: SELECT COLUMN PROPERTY (object_ id(‘project’),’project_no’,’PRECISION’)= 4	COLUMN PROPERTY(id, col,property)
مقدار مشخصه‌ی تعیین شده برای پایگاه داده را برمی‌گرداند.	DATABASE PROPERTY (database, property)
شناسه‌ی پایگاه داده‌ی db_name را برمی‌گرداند. اگر نامی تعیین نشود، شناسه‌ی پایگاه داده‌ی جاری برگردانده می‌شود. مثال: SELECT DB_ID(‘Adventureworks’)= 6	DB_ID ([db_name])
نام پایگاه داده‌ای با شناسه‌ی db_id را برمی‌گرداند. اگر هیچ شناسه‌ای تعیین نشود، نام پایگاه داده‌ی جاری برگردانده می‌شود. مثال: SELECT DB_NAME(6)= ‘Adventureworks’	DB_NAME ([db_id])
نام ستون شاخص‌گذاری شده در جدول table را بر می‌گرداند.	INDEX_COL (table,i,no)

OBJECT_NAME (obj_id)	**نام شیء پایگاه داده با شناسه‌ی obj_id را برمی‌گرداند. مثال:** SELECT OBJECT-NAME(453576654)= ‘products’
OBJECT_ID (obj_name)	**شناسه‌ی شیء پایگاه داده با نام obj_name را برمی‌گرداند. مثال:** SELECT OBJECT_ID(‘products’)= 453576654
OBJECT PROPERTY (obj_id, property)	**اطلاعاتی درباره‌ی شیءهای پایگاه داده‌ی جاری، را برمی‌گرداند.**

پیوست پ

انواع داده های متفرقه در SQL Server

انواع داده‌های متفرقه در SQL

Transact-SQL از چندین نوع داده پشـــتیبانی می‌کند که بـــه هیچ‌کدام از گروه‌های داده‌ای که شرح دادیم، متعلق نیستند:

انواع داده‌های دودویی

BIT

انواع داده‌های شیئی بزرگ

CURSOR

UNIQUEIDENTIFIER

SQL-VARIANT

TABLE

XML

انواع داده‌های فضایی (مثل GEOGRAPHY و GEOMETRY).

HIERARCHYID

انواع داده ی Timestamp

انواع داده‌هایی که به وسیله‌ی کاربر تعریف می‌شوند.

بخش‌های زیر، بعضی از این نوع داده‌ها را شرح می‌دهند.

انواع داده‌های دودویی و BIT

BINARY و VARBINARY دو نـــوع داده‌ی دودویـــی هســـتند که بـــرای ذخیره‌ی

رشـــته‌های بیتی به کار می‌روند. به همین دلیل، مقادیر با استفاده از اعداد شانزده تایی وارد می‌شوند.

مقادیر نوع داده‌ی BIT در یک بیت ذخیره می‌شود. بنابراین، ستون‌های ۸ بیتی در یک بایت ذخیره می‌شوند. جدول زیر، مشخصات این نوع داده‌ها را خلاصه نموده است:

نوع داده	شرح
BINARY[(n)]	**یک رشته‌ی بیتی با طول ثابت n بایت را مشخص می‌کند(۰<n≤۸۰۰۰)**
VARBINARY[(n)]	**یک رشته‌ی بیتی با طول متغیر n بایت را مشخص می‌کند(۰<n≤۸۰۰۰)**
BIT	**برای تعیین نوع داده‌ی Boolean با سه مقدار False، TRUE و NULL مورد استفاده قرار می‌گیرد.**

انواع داده‌های شیء بزرگ

شـــیءهای بزرگ (LOBS)، شـــیءهای داده‌ای با حداکثر طول ۲ گیگابایت هستند. این شـــیءها برای ذخیره‌ی داده‌های متنی بزرگ و برای بارگـــذاری مدول‌ها و فایل‌های صوتی/تصویری مورد اســـتفاده قرار می‌گیرند. Transact-SQL از دو روش مختلف، برای تعیین و دسترسی به LOBS پشتیبانی می‌کند:

به کارگیری انواع داده‌های

Varbinary(MAX) و Nvarchavar(max) ، Varchar(max)

استفاده از نوع داده‌ی متنی/تصویری

نوع داده‌ی Uniqueidentifier

همان طور که از نام آن پیداســـت، یک شناســـه‌ی عددی یکتا (منحصر به فرد) به صورت رشـــته‌ی دودویی ۱۶ بایتی در آن ذخیره می‌شـــود. این نوع داده به شناسه‌ی یکتای جهانی (GUID) مرتبط است. با استفاده از این نوع داده، می‌توان داده‌ها وشیءها را در سیستم‌های توزیع شده به طور یکتا شناسایی کرد.

مقداردهی یک ســـتون یـــا متغیر از این نـــوع می‌تواند با اســـتفاده از توابع NEWID و

NEWSEQUENTIALID و ثابت رشـــته‌ای که به شکل خاصی از رقم‌های شانزده‌تایی و خط تیره اســـت، انجام شود.یک ســـتون از این نوع داده را می‌توان در یک پرس‌وجو با کلید واژه‌ی ROWGUIDCOL مورد اشـــاره قرار دارد تا تعیین شود که آن ستون شامل مقادیر ID اســـت. یک جدول می‌تواند چندین ســـتون از این نوع داده داشته باشد ولی فقط یکی از آن ها دارای کلید واژه‌ی ROWGUIDCOL است.

نوع داده‌ی SQL–VARIANT

این نوع داده می‌تواند برای ذخیره‌ی مقادیری از انواع مختلف داده‌ها به صورت هم‌زمان مثل مقادیر عددی، رشته‌ها و تاریخ مورد استفاده قرار گیرد (فقط انواع مقادیری که نمی‌توانند ذخیره شـــوند، عبارت‌اند از مقادیر TIMESTAMP). هر مقداری از یک ســـتون دارای دو بخش است: مقدار داده و اطلاعاتی که مقدار داده را توصیف می‌کند(این اطلاعات شامل تمام مشخصات نوع داده‌ی واقعی مثل طول، اندازه و دقت است).

Transact-SQL از تابـــع SQL-VARIANT-PROPERTY پشـــتیبانی می‌کند که شامل اطلاعات مربوط به مقدار یک ستون SQL-VARIANT است.

نکته

یک ســـتون از جدول را فقط در صورتی از نوع داده‌ی SQL-VARIANT تعریف کنید که واقعاً نیاز دارید.

نوع داده‌ی HIERARCHYID

ایـــن نـــوع داده برای ذخیره‌ی یک سلســـله مراتب کامل اســـتفاده می‌شـــود. این نوع به

صورت یک نوع CLR(Common Language Runtime) تعریف شـده به وسیله‌ی کاربر پیاده‌سـازی می‌شود که شامل چندین تابع سیسـتمی برای ایجاد و عملیات روی گروه‌های سلسله مراتب است. توابع زیر مربوط به این نوع داده هستند:

()GetAncestor( ), GetDescendant( ), Read( ), Write

نوع داده‌ی TimesTamp

این نوع داده، سـتونی را که به صورت VARBINARY(8)یا BINARY(8) تعریف شده است تعیین می‌کند که بستگی به NULL بودن ستون دارد. سیستم یک مقدار جاری برای هر پایگاه داده را نگه می‌دارد که با درج یا به هنگام شدن هر ستون از این نوع، افزایش می‌یابد. بنابراین، ستون‌های TimesTamp می‌توانند برای تعیین زمان مربوطه به آخرین تغییر سطر مورد استفاده قرار گیرند.

نکته

مقدار ذخیره شـده در یک ستون TIMESTAMP برای خودش مهم نیست. این ستون معمولاً برای تشخیص این که سطر خاصی از آخرین بار دسترسی تغییر یافته است، به کار می‌رود.

Decimal همراه با قالب ذخیره‌سازی Vardrcimal

نوع داده‌ی Decimal روی دیسـک به صورت یک داده‌ی طول ثابت ذخیره می‌شـود. با اسـتفاده از VARDECIMAL، می‌توان به طور چشم گیری فضای ذخیره‌سازی یک ستون DECIMAL را کاهـش داد، زیـرا مقادیر مختلف به اندازه‌ی طول خودشـان فضا اشـغال

می‌کنند.

نکته

VARDECIMAL یک قالب ذخیره‌سازی است و نه نوع داده.

قالب ذخیره‌ســازی VARDECIMAL هنگامی که باید بزرگ‌ترین مقدار ممکن برای فیلدی را که معمولاً مقادیر خیلی کوچک در آن ذخیره شــده‌اند تعیین کنید، مفید خواهد بود. جدول ۱-پ این مطلب را نشان می‌دهد.

نکته

VARDECIMAL برای نوع داده‌ی DECIMAL مشــابه نوع داده‌ی VARCHAR برای داده‌های الفبایی-عددی است.

جدول ۱-پ) تعداد بایت‌ها برای ذخیره سازی VARDECIMAL و طول ثابت

تعداد بایت‌ها: طول ثابت	تعداد بایت‌ها: VARDECIMAL	دقت
۵	۲	صفر یا پوچ
۵	۴	۱
۱۳	۱۲	۲۰
۱۷	۱۶	۳۰
۱۷	۲۰	۳۸

برای فعال‌سازی قالب ذخیره‌سازی VARDECIMAL، ابتدا باید آن را برای پایگاه داده و برای جدول خاصی، فعال کنید.

منابع

۱- مفاهیم بنیادی پایگاه داده‌ها، دکتر سید محمد تقی روحانی رانکوهی، انتشارات جلوه

۲- بانک اطلاعاتی علمی کاربردی، دکتر مصطفی حق‌جو، انتشارات دانشگاه علم و صنعت

3-An introduction to Database – Date, C.J – McGrawHill – 8 ed.

4-Teach Yourself Visually Access 2007 – Wempen, Faithe – SAMS

5-Access 2007 Complete Reference – John Leonie, Margaret Leonie Yang

6-Microsoft SQL Server 2008 (A Beginner's Guide) - Dušan Petkovic – McGraw Hill-2009

7-Introducing Microsoft SQL Server 2008 –Peter DeBetta, Greg Low, Mark Whitehorn – Microsoft – 2009

8-SQL Fundamentals – Third Edition – John J. Patrick – Prentice Hall - 2009

9-Beginning SQL Server 2008 for Developer: From novice to Professional – Dewson Robin- Apress