

آزمایشگاه پایگاه داده با SQL Server 2012

برای رشته های مهندسی کامپیوتر
IT و علوم کامپیوتر

مؤلفین:

مهندس رمضان عباس نژاد ورزی

مهندس فاطمه عبدی سقاواز

مهندس بهارک شاکری اسکی



برخی از عناوین مهم

مراحل طراحی بانک اطلاعاتی
دستورات ایجاد، تغییر و حذف بانک اطلاعاتی و جدول
بازیابی اطلاعات، پرس و جوهای فرعی و پیوند جداول
ایجاد، تغییر و حذف دیدها
ایجاد، تغییر و حذف رویه های ذخیره شده و توابع
بازیابی اطلاعات از کاتالوگ
پشتیبان گیری و بازیابی پشتیبان
ایجاد و استفاده از کرسرها
ایجاد و استفاده از تریگرها
دستور کار آزمایشگاه

آزمایشگاه پایگاه داده

با

SQL Server ۲۰۱۲

تالیف:

مهندس رمضان عباس نژادورزی
مهندس فاطمه عبدی سقاواز
مهندس بهارک شاکری اسکی



فن آوری نوین

سرشناسه	: عباس نژاد ورزی، رمضان، ۱۳۴۸
عنوان و نام پدیدآور	: آزمایشگاه پایگاه داده با ۲۰۱۲ SQL Server / تالیف رمضان عباس نژاد ورزی، فاطمه عبدی سقاواز، بهارک شاکری اسکی
مشخصات نشر	: مازندران: فن آوری نوین، ۱۳۹۲.
مشخصات ظاهری	: ۱۷۶ ص: مصور،
شابک	: ۹۷۸-۶۰۰-۹۲۲۵۴-۶-۰
وضعیت فهرست نویسی	: فیبا
موضوع	: سرور اس. کیو. ال
موضوع	: پایگاه های اطلاعاتی -- طراحی
شماره افزوده	: عبدی سقاواز، فاطمه، ۱۳۵۶ -
شماره افزوده	: شاکری اسکی، بهارک، ۱۳۶۲ -
رده بندی کنگره	: ۲۱۳۹۲ع۴/۷۶/۹QA
رده بندی دیویی	: ۰۰۵/۷۵۸۵
شماره کتابشناسی ملی	: ۳۰۶۷۸۳۶



فن آوری نوین

www.fanavarienovin.net

بابل، صندوق پستی ۷۳۴۴۸-۴۷۱۶۷

تلفن: ۰۱۱۱-۲۲۵۶۶۸۷

آزمایشگاه پایگاه داده با ۲۰۱۲ SQL Server

تالیف: مهندس رمضان عباس نژاد ورزی - مهندس فاطمه عبدی سقاواز - بهارک شاکری اسکی

ناشر: فن آوری نوین

چاپ اول: بهار ۱۳۹۲

جلد: ۱۰۰۰

شابک: ۹۷۸ - ۶۰۰ - ۹۲۲۵۴ - ۶ - ۰

قیمت: ۸۸۰۰ تومان

حروفچینی و صفحه آرای: فن آوری نوین

تلفکس: ۶۶۴۰۰۱۴۴-۶۶۴۰۰۲۲۰

تهران، خ

اردیبهشت، نبش وحید نظری، پلاک ۱۴۲

فصل اول: مراحل طراحی بانک اطلاعات

۷

۱-۱. مراحل طراحی بانک اطلاعاتی.....	۷
۱-۱-۱. تعیین کاربرد اصلی بانک اطلاعاتی.....	۷
۱-۱-۲. تعیین جداول مورد نیاز بانک اطلاعاتی.....	۸
۱-۱-۳. تعیین فیلدهای مورد نیاز بانک اطلاعاتی.....	۱۰
۱-۱-۴. تعریف رابطه‌های بین جداول.....	۱۱
۱-۱-۵. بهینه سازی طراحی.....	۱۲
۱-۲. بانک اطلاعاتی SQL Server.....	۱۳
۱-۲-۱. ورود به بانک اطلاعاتی SQL Server.....	۱۳
۱-۲-۲. تایپ و اجرای دستورات SQL.....	۱۴
۱-۳. ایجاد بانک اطلاعاتی.....	۱۴
۱-۳-۱. گروه‌های فایل.....	۱۷
۱-۳-۲. تغییر خواص بانک اطلاعاتی موجود.....	۱۷
۱-۳-۳. حذف بانک اطلاعاتی موجود.....	۱۸
۱-۴. ایجاد و تغییر ساختار جدول.....	۲۱
۱-۴-۱. ایجاد جدول.....	۲۴
۱-۴-۲. تغییر ساختار جدول با دستور SQL.....	۲۷
۱-۴-۳. حذف جدول با دستور SQL.....	۲۸
۱-۵. مسائل حل شده.....	۲۸
۱-۶. دستور کار آزمایشگاه.....	۳۸
۱-۷. جواب دستور کار آزمایشگاه.....	۴۰
۱-۸. تمرین‌ها.....	۴۳

فصل دوم: ورود، ویرایش، حذف و بازیابی اطلاعات

۴۵

۲-۱. دستور INSERT.....	۴۶
۲-۲. ویرایش رکوردهای جدول.....	۴۸
۲-۳. حذف رکوردهای جدول.....	۴۹
۲-۴. پرس و جوی بازیابی اطلاعات.....	۵۰
۲-۴-۱. عملگرها در SQL.....	۵۰
۲-۴-۲. بازیابی اطلاعات از جدول با دستور SELECT.....	۵۴
۲-۵. مرتب سازی رکوردها.....	۵۷
۲-۶. پرس و جوی مرکب.....	۵۸
۲-۶-۱. عملگر UNION.....	۵۸
۲-۶-۲. عملگر UNION ALL.....	۵۸
۲-۶-۳. عملگر INTERSECT.....	۵۹
۲-۶-۴. عملگر EXCEPT.....	۵۹
۲-۷. توابع تجمعی.....	۶۰
۲-۸. گروه بندی اطلاعات.....	۶۰
۲-۹. تست مقدار تهی فیلد.....	۶۲
۲-۱۰. پرس و جوی فرعی.....	۶۲
۲-۱۱. پیوند جداول (رابطه).....	۶۵
۲-۱۱-۱. پیوند ضربدری.....	۶۵
۲-۱۱-۲. پیوند متعادل.....	۶۶
۲-۱۱-۳. پیوند نامتعادل.....	۶۷
۲-۱۱-۴. پیوند درونی.....	۶۷
۲-۱۱-۵. پیوند بیرونی.....	۶۸
۲-۱۲. افزودن مقادیر از یک جدول به جدول دیگر.....	۷۰
۲-۱۳. حذف رکوردهای یک جدول از طریق جداول دیگر.....	۷۱
۲-۱۴. مسائل حل شده.....	۷۲
۲-۱۵. دستور کار آزمایشگاه.....	۸۳

۱۶-۲. پاسخ دستور کار آزمایشگاه..... ۸۴

۱۷-۲. تمرین‌ها..... ۸۶

فصل سوم: دیده‌ها، رویه‌های ذخیره شده و توابع..... ۹۰

۳-۱. دید..... ۹۰

۳-۱-۱. ایجاد دید..... ۹۰

۳-۱-۲. تغییر دید..... ۹۲

۳-۱-۳. حذف دید..... ۹۳

۳-۲. برنامه‌نویسی در SQL..... ۹۳

۳-۲-۱. متغیرها..... ۹۳

۳-۲-۲. توضیحات..... ۹۴

۳-۲-۳. دستور RETURN..... ۹۴

۳-۲-۴. دستور PRINT..... ۹۴

۳-۲-۵. دستور RAISERROR..... ۹۴

۳-۲-۶. دستور WAITFOR..... ۹۵

۳-۲-۷. ساختارهای تصمیم..... ۹۶

۳-۳. رویه‌های ذخیره شده..... ۹۸

۳-۳-۱. ایجاد رویه‌های ذخیره شده..... ۹۹

۳-۳-۲. انواع رویه‌های ذخیره شده..... ۱۰۰

۳-۳-۳. اجرای رویه ذخیره شده..... ۱۰۶

۳-۳-۴. تغییر رویه ذخیره شده..... ۱۱۰

۳-۴. توابع تعریف شده توسط کاربر..... ۱۱۱

۳-۴-۱. انواع توابع تعریف شده توسط کاربر..... ۱۱۲

۳-۴-۲. فراخوانی توابعی که یک مقدار اسکالر را برمی‌گردانند..... ۱۱۴

۳-۴-۳. توابعی که یک جدول را برمی‌گردانند..... ۱۱۶

۳-۴-۴. توابعی که یک خطی را برمی‌گردانند..... ۱۱۷

۶-۴-۳. توابع چند دستوری جدولی..... ۱۱۷

۵-۳. تغییر تابع..... ۱۱۹

۱-۵-۳. حذف تابع..... ۱۱۹

۶-۳. مسائل حل شده..... ۱۲۰

۷-۳. دستور کار آزمایشگاه..... ۱۲۸

۸-۳. پاسخ دستور کار آزمایشگاه..... ۱۳۰

۹-۳. تمرین..... ۱۳۵

فصل چهارم: مباحث پیشرفته در SQL Server..... ۱۳۸

۴-۱. بازیابی اطلاعات از کاتالوگ..... ۱۳۸

۴-۲. پشتیبان‌گیری از اطلاعات..... ۱۴۱

۱-۲-۴. پشتیبان‌گیری با دستور BACKUP..... ۱۴۳

۲-۲-۴. دستور DATABASE..... ۱۴۳

۲-۲-۴. دستور RESTORE DATABASE..... ۱۴۴

۳-۴. تریگرها..... ۱۴۶

۱-۳-۴. انواع تریگرها..... ۱۴۸

۲-۳-۴. ایجاد تریگر..... ۱۴۸

۳-۳-۴. مشاهده تریگرهای ایجاد شده بر روی جدول..... ۱۵۱

۴-۳-۴. پیاده‌سازی کاربردهای تریگر..... ۱۵۲

۵-۳-۴. تغییر تریگر..... ۱۵۵

۶-۳-۴. حذف تریگر..... ۱۵۶

۴-۴. کرسرها..... ۱۵۶

۱-۴-۴. معرفی کرسر..... ۱۵۷

۲-۴-۴. مقداردهی به کرسر..... ۱۵۸

۴-۴. مسائل حل شده..... ۱۶۱

۵-۴. دستور کار آزمایشگاه..... ۱۶۷

۶-۴. پاسخ دستور کار آزمایشگاه..... ۱۷۰

۷-۴. تمرین‌ها..... ۱۷۴

نام کتاب	لینک خرید الکترونیکی
مبانی رایانه و برنامه نویسی به زبان ++C	http://ktbr.ir/b30588
آشنایی با مبانی امنیت شبکه (امنیت اطلاعات)	http://ktbr.ir/b30327
اصول طراحی پایگاه داده ها	http://ktbr.ir/b29943
آموزش گام به گام برنامه نویسی پایتون	http://ktbr.ir/b29984
حل مسائل ++C (آزمایشگاه کامپیوتر مرجع کامل)	http://ktbr.ir/b29982
آموزش گام به گام LINQ با C#	http://ktbr.ir/b28451
ساختمان داده ها با ++C	http://ktbr.ir/b29676
طراحی سیستم های شی گرا با زبان C#	http://ktbr.ir/b29621
مدیریت استراتژیک فناوری اطلاعات	http://ktbr.ir/b29779
گرافیک رایانه ای با زبان برنامه نویسی C#	http://ktbr.ir/b29674
درس و کنکور پایگاه داده پیشرفته	http://ktbr.ir/b29644
فیزیک الکتریسته	http://ktbr.ir/b29680
تجارت الکترونیکی	http://ktbr.ir/b29619
راهنمای کاربردی کاربری OPNET برای شبکه های شبیه سازی کامپیوتر	http://ktbr.ir/b28504
درس و کنکور سیستم عامل پیشرفته	http://ktbr.ir/b28505
شبکه های کامپیوتری با رویکرد کاربردی، آزمایشگاه شبیه سازی شبکه	http://ktbr.ir/b28528
آزمایشگاه پایگاه داده با SQL Server ۲۰۱۲	http://ktbr.ir/b28503

http://ktbr.ir/b۲۸۴۵۰	کاربرد رایانه در مدیریت و حسابداری
http://ktbr.ir/b۲۸۴۴۹	آموزش گام به گام برنامه نویسی بانک اطلاعاتی با ویژال بیسیک نت
http://ktbr.ir/b۲۸۴۵۲	آموزش گام به گام برنامه نویسی به زبان ++C
http://ktbr.ir/b۲۸۴۴۸	دانلود کتاب آموزش گام به گام برنامه نویسی بانک اطلاعاتی با C#
http://ktbr.ir/b۲۸۳۹۸	حل مسائل پاسکال
http://ktbr.ir/b۲۸۴۰۱	حل مسائل ++C
http://ktbr.ir/b۲۸۳۹۹	دانلود کتاب حل مسائل C#
http://ktbr.ir/b۲۸۳۹۷	دانلود کتاب حل مسائل C

مقدمه

امروزه حجم زیادی از داده، ذخیره، پردازش و بازیابی می‌شوند. از طرف دیگر، در سیستم‌های امروزی از قبیل سیستم‌های تجارت الکترونیک، داده‌کاوی، مدیریت مشتریان و پشتیبانی تقسیم داده جایگاهی بسیار مهمی دارد، درس آزمایشگاه پایگاه داده یکی از دروس تخصصی رشته‌های مهندسی کامپیوتر و فناوری اطلاعات در نظر گرفته شده است. به همین دلیل سعی شده تا کتاب حاضر تدوین شود. تمام مطالب کتاب با توجه بر فصل مصوب وزارت علوم و تحقیقات، به زبان ساده و روان تدوین گردیده است. کتاب حاضر شامل ۴ فصل است. هر فصل آن از چهار بخش تشکیل شده است که عبارت‌اند از:

۱. تشریح مفاهیم به آن فصل همراه پرس‌و‌جوهای متعدد (در کل کتاب با بانک اطلاعاتی یک انتشارات بیان گردید).

۲. مسائل حل شده (در تمام فصل‌های کتاب با مثال خرید، فروش و توزیع کالا بیان گردید).

۳. دستور کار آزمایشگاه (در تمام فصل‌ها بانک اطلاعاتی آژانس حمل نقل بیان گردید).

۴. تمرین‌ها (در تمام فصل بانک اطلاعاتی سیستم حسابداری بیان شده).

از جمله ویژگی بسیار مهم کتاب حاضر، پیوستگی مطالب، مثال‌ها و اجرا مسائل آن بر روی بانک اطلاعاتی SQL Server می‌باشد. در فصل اول کتاب، فرآیند طراحی بانک اطلاعاتی، ایجاد بانک اطلاعاتی، ایجاد و تغییر ساختار جداول بیان گردیده است. در فصل دوم، مفاهیم افزودن، ویرایش، حذف و بازیابی رکوردهای جدول بحث شده است. در فصل سوم مفاهیم دید، رویه‌های ذخیره شده و توابع و پیاده‌سازی آن‌ها در SQL Server به طور کامل بیان گردید. در فصل چهارم، مفاهیم کاتالوگ، پشتیبان-گیری و بازیابی پشتیبان، تریگرها و کسررها و چگونگی پیاده‌سازی آن‌ها در SQL Server بیان گردیده است. کتابی که در پیش رو دارید، با بهره‌گیری از سال‌ها تجربه در امر تدریس در درس پایگاه داده به ویژه آموزش SQL Server تالیف شده است.

امیدواریم این اثر نیز مورد توجه اساتید و دانشجویان عزیز قرار گیرد.

در پایان از تمامی اساتید و دانشجویان عزیز تقاضا داریم، هر گونه اشکال، ابهام در متن کتاب و پیشنهاد و انتقادات را به آدرس پست الکترونیک www.fanavarienovin@Gmail.com ارسال نمایند.

۱-۱. مراحل طراحی بانک اطلاعاتی

طراحی هر سیستم نرم‌افزاری از جمله بانک اطلاعاتی از فرآیند منظمی تشکیل می‌گردد. در این بخش فرآیند ایجاد یک بانک اطلاعاتی را می‌آموزیم. گام‌های طراحی هر بانک اطلاعاتی در زیر آمده است:

✚ تعیین کاربرد اصلی بانک اطلاعاتی

✚ تعیین جداول مورد نیاز بانک اطلاعاتی

✚ تعیین فیلدهای مورد نیاز جداول

✚ تعریف رابطه‌های بین جداول

✚ بهینه‌سازی طراحی

۱-۱-۱. تعیین کاربرد اصلی بانک اطلاعاتی

این مرحله به شما کمک خواهد کرد تا نوع اطلاعاتی را که باید از بانک اطلاعاتی استخراج گردد، معین نمایید. در این صورت می‌توانید مشخص کنید که چه موضوعاتی را برای ثبت داده‌ها باید تفکیک نمایید (جداول تشکیل‌دهنده بانک اطلاعاتی تان چه هستند) و داده‌هایی که در جداول قرار می‌گیرند، چه نوع‌اند. فیلدهای تشکیل‌دهنده جداول تان دارای چه نوع هستند. برای نیل به این هدف باید با کاربران بانک اطلاعاتی مصاحبه کنید. فرم‌های اولیه را از آن‌ها دریافت نمایید و تعیین کنید این بانک اطلاعاتی باید به چه نیازهای آن‌ها پاسخ دهد. به عنوان مثال، بانک اطلاعاتی را در نظر بگیرید که برای پخش، خرید و فروش محصولات به کار می‌رود. در این بانک باید لیستی از پرسش‌های زیر پاسخ داده شوند:

۱. در طول ماه یا سال گذشته چه مقدار از کالاهای موجود فروخته شده است (آیا این موجودی‌ها به تفکیک کالا نیاز است)؟

۲. بهترین مشتریان در چه مناطقی زندگی می‌کنند؟

۳. تأمین‌کننده پرفروش‌ترین کالای ما چه شرکتی است؟

۴. چه اطلاعاتی از کالا، تأمین‌کنندگان و مشتریان باید نگهداری شود؟

۵. آیا فروش به صورت نسبه انجام می‌شود؟ اگر چنین است، آیا اعتبار هر مشتری با مشتری دیگر متفاوت است یا خیر؟

۶. سوالات دیگر

۲-۱-۱. تعیین جداول مورد نیاز بانک اطلاعاتی

شاید در طول فرآیند طراحی بانک اطلاعاتی، تعیین جداول یکی از مشکل‌ترین مراحل باشد. چون گزارش‌های قابل چاپی که از بانک اطلاعاتی می‌گیرید و پرسش‌هایی که مایلید پاسخ داده شوند، لزوماً ساختار جدولی که آن‌ها را تولید می‌کنند، تعیین نمی‌کنند. کاربران به شما می‌گویند چه می‌خواهند، اما مشخص نمی‌کنند که این اطلاعات چگونه باید در داخل جداول مختلف دسته‌بندی شوند. یعنی فرم‌هایی که از کاربران دریافت می‌کنید را نمی‌توانید به همین صوت داخل جداول بریزید. زیرا:

۱. ممکن است اطلاعات تکراری وارد شود (افزونگی داده داشته باشید). فرض کنید یک مشتری چند سفارش مختلف داده باشد. اگر برای هر سفارش مشتری، اطلاعاتش از قبیل نام، شماره تلفن و نشانی را دریافت کنید، ممکن است برای یک مشتری اطلاعات تکراری داشته باشید. این عمل علاوه بر ورود داده‌های تکراری و زائد (افزونگی داده)، امکان بروز خطا را در هنگام ورود داده چند برابر خواهد کرد.

۲. فرض کنید نشانی (آدرس) مشتری تغییر یابد. در این صورت باید تمام سفارشات مشتری را پیدا کرده، نشانی او را تغییر دهید. این عمل ممکن است موجب بی‌نظمی شود.

۳. از دست دادن اطلاعات مفید، فرض کنید یک مشتری سفارشی را داده اما بعداً آن را لغو نماید. اگر این سفارش را از جدولی پاک کنید که هم شامل اطلاعات مربوط به خود مشتری و هم اطلاعات سفارشات است، داده‌های مربوط به مشخصات مشتری نیز به همراه داده‌های سفارش حذف خواهد شد. در صورتی که ممکن است مایل باشید مشخصات مشتری جدید را در بانک اطلاعاتی خودتان ثبت کنید تا کاتالوگ محصولات جدید خود را برای او بفرستید.

برای رفع این مشکلات، بهتر است جدول مشتریان را جداگانه در نظر بگیرید. برای این منظور باید جداول بانک اطلاعاتی را نرمال کنید^۱. نرمال سازی موجب کاهش افزونگی داده، بی‌نظمی و کاهش داده‌های Null می‌شود.

بانک اطلاعاتی را در نظر بگیرید که اطلاعات یک انتشارات را نگهداری می‌کند. این کتاب بانک اطلاعاتی دارای جداول زیر است:

📚 **جدول Books**، اطلاعات کتاب‌ها از قبیل شابک کتاب، عنوان، تعداد صفحات و غیره را ذخیره می‌کند.

📚 **جدول Publishers**، کد ناشر، نام ناشر، شماره تلفن و غیره را برای ناشران نگهداری می‌کند.

📚 **جدول Authors**، اطلاعات مؤلفین از قبیل کد مؤلف، نام مؤلف، نام خانوادگی، سن و غیره را نگهداری می‌کند.

📚 **جدول AutBook**، اطلاعات شابک کتاب، مؤلف و مبلغ حق تالیف را نگهداری می‌کند.

📚 **جدول PubBook**، اطلاعات شابک، کد ناشر، تاریخ نشر و حق نشر را نگهداری می‌کند.

^۱. نرمال سازی جدول، یعنی شکستن یک جدول به چند جدول دیگر. مباحث مربوط به نرمال سازی را می‌توانید در کتاب اصول طراحی پایگاه داده از انتشارات فن آوری نوین مولفین رمضان عباس‌نژاد ورزی، علیرضا عظیمی و باقر رحیم پورکامی ببینید.

🚩 جدول GroupBook، اطلاعات کد گروه کتاب و نام گروه کتاب را نگهداری می کند.

جدول ۱ - ۱ جدول انتشارات.						
نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
Books (کتاب‌ها)	ISBN	کد کتاب یا شابک	Varchar(۱۰)	بله	نه	نه
	Title	عنوان کتاب	Varchar(۱۰)	نه	نه	نه
	Page	تعداد صفحات	int	نه	نه	نه
	Price	قیمت کتاب	Decimal(۱۵, ۰)	نه	نه	نه
	EditNo	شماره ویرایش	int	نه	بله	نه
	PrintNo	نوبت چاپ	int	نه	بله	نه
	groupID	کد گروه کتاب	Varchar(۶)	نه	نه	بله
Publishers (ناشران)	pubID	کد ناشر	Varchar(۱۰)	بله	نه	نه
	pName	نام ناشر	Varchar(۵۰)	نه	نه	نه
	Tel	شماره تلفن ناشر	Varchar(۱۵)	نه	بله	نه
	URL	آدرس وبسایت ناشر	Varchar(۱۰۰)	نه	بله	نه
	CityName	نام شهر ناشر	Varchar(۵۰)	نه	بله	نه
	bFname	نام مدیر انتشارات	Varchar(۲۰)	نه	نه	نه
	bLname	نام خانوادگی مدیر انتشارات	Varchar(۲۰)	نه	نه	نه
	CountBookPrint	تعداد کتاب‌های چاپ شده	int	نه	نه	نه
Authors (مؤلفان)	atID	کد مؤلف	Varchar(۱۰)	بله	نه	نه
	aFname	نام مؤلف	Varchar(۲۰)	نه	نه	نه
	aLname	نام خانوادگی مؤلف	Varchar(۲۰)	نه	نه	نه
	Age	سن مؤلف	int	نه	نه	نه
	Ranking	مدرک مؤلف	Varchar(۳۰)	نه	بله	نه
	Email	پست الکترونیک مؤلف	Varchar(۵۰)	نه	بله	نه
	Mobile	موبایل مؤلف	Varchar(۱۵)	نه	بله	نه
	SumPayment	مجموع حق تالیف	Decimal(۱۸, ۰)	نه	بله	نه
AutBook (کتاب‌های تالیف شده)	ISBN	کد کتاب	Varchar(۱۰)	بله	نه	بله
	atID	کد مؤلف	Varchar(۱۰)	بله	نه	بله
	Payment	حق تالیف	Decimal(۱۸, ۰)	نه	نه	نه
PubBook (کتاب‌های نشر یافته)	ISBN	شابک	Varchar(۱۰)	بله	نه	بله
	PubID	کد ناشر	Varchar(۱۰)	بله	نه	بله
	PubDate	تاریخ نشر	Varchar(۸)	نه	بله	نه
	Payment	حق نشر	Decimal(۱۸, ۰)	نه	بله	نه
GroupBook (گروه کتاب)	groupID	کد گروه کتاب	Varchar(۶)	نه	نه	نه
	groupName	نام گروه کتاب	Varchar(۳۰)	بله	نه	نه

۳-۱-۱. تعیین فیلدهای مورد نیاز بانک اطلاعاتی

برای تعیین فیلدهای یک جدول، باید نوع اطلاعات آن را مشخص کنید. یعنی، داده‌ها، اشخاص، اشیا یا رویدادهای ثبت شده در جدول را بدانید. می‌توانید فیلدها را به عنوان مشخصه‌های^۲ (خواص) یک جدول به حساب آورید. هر رکورد (سطر) در جدول، شامل فیلدها یا مشخصه‌ها است. به عنوان مثال، فیلد "آدرس" در جدول "مشتریان" شامل آدرس همه مشتریان است. به عبارت دیگر، هر رکورد در جدول مشتریان شامل داده‌هایی درباره یک مشتری است و فیلد آدرس شامل آدرس آن مشتری می‌باشد. در هنگام تعیین فیلدهای جداول به نکات زیر دقت کنید:

۱. هر فیلد را مستقیماً به موضوع آن جدول مربوط کنید، فیلدی که به موضوع جدول دیگری مربوط می‌گردد، جایش در همان جدول است. به عنوان مثال، فیلد "آدرس مشتری" مربوط به جدول مشتریان است. دیگر نیاز نیست در جدول سفارشات در نظر گرفته شود. در مراحل بعدی طراحی بانک اطلاعاتی (هنگامی که رابطه بین جداول را تعریف کنید)، خواهید دید که چگونه می‌توان داده‌های فیلدهای مختلف چندین جدول را با هم ترکیب نمود.

۲. داده‌های حاصل از محاسبات را ذخیره نکنید، یعنی فضای برای فیلدهای مشتق را در جدول در نظر نگیرید. به عنوان مثال، در هنگام خرید یا فروش محصولات، چنانچه قیمت و درصد تخفیف را داشته باشید، نیازی نیست مبلغ تخفیف را در جدول نگهداری کنید یا وقتی قیمت و تعداد اقلام کالا را داشته باشید، نیازی نیست مبلغ کل را نگهداری نمایید.

۳. همه اطلاعات لازم را در جدول قرار دهید، برای این منظور کافی است به اطلاعاتی که در مرحله کاربرد اصلی بانک اطلاعاتی جمع آوری کرده‌اید، رجوع کرده، به کاغذها، فرم‌ها و گزارش‌ها نگاهی کرده تا مطمئن شوید، تمام اطلاعات را در جدول در نظر گرفته‌اید یا می‌توانید از ترکیب چند فیلد به اطلاعات مورد نظر برسید.

۴. اطلاعات را در کوچک‌ترین واحدهای منطقی قرار دهید، ممکن است نام کامل اشخاص (نام و نام خانوادگی) یا آدرس (کشور، استان، شهر، خیابان، کوچه و ...) را در یک فیلد قرار داده باشید. اگر بیش از یک اطلاعات را در یک فیلد قرار دهید، بعداً ایجاد پرس‌وجو براساس تک تک آن‌ها دشوار خواهد شد. سعی کنید اطلاعات را به قطعات منطقی و مستقل کوچک‌تری بشکنید. یعنی، فیلدهای مرکب در جدول نداشته باشید (اولین شرط نرمال سازی فرم یک است).

۵. فیلدهای چند مقداری در جدول نداشته باشید. یعنی اگر جدولتان دارای فیلد چند مقداری مانند مدرک تحصیلی است، برای فیلد چند مقداری یک جدول مجزا در نظر بگیرید.

^۲. Properties (Attributes)

۶. **فیلدهای کلید اولیه**^۲ را تعیین کنید، باید هر جدول بانک اطلاعاتی دارای یک فیلد یا ترکیب گروهی از فیلدها باشد، که هر رکورد جدول را به صورت **منحصر به فرد** (یکتا)^۳ مشخص می کند. این فیلد معمولاً یک شماره هویت منحصر به فرد (ID) مانند شماره پرسنلی، شماره دانشجویی، کد کالا، شماره ملی و غیره است. اگر برای جدولی نمی توانید فیلد کلید اولیه پیدا کنید یا ترکیب همه فیلدهای جدول را به عنوان فیلد کلید اولیه در نظر بگیرید یا فیلدی به جدول اضافه کنید که به عنوان فیلد کلید اولیه جدول باشد. فیلدهای کلید اولیه جداول مختلف بانک اطلاعاتی انتشارات در زیر آمده اند:

🔗 **در جدول Books**، فیلد ISBN (شابک) کلید اولیه است. زیرا، هیچ دو کتابی شابک یکسان ندارند.
🔗 **در جدول Publishers**، فیلد PubID (کد ناشر) کلید اولیه است. زیرا، هیچ دو ناشری شماره پروانه یکسانی ندارند.

🔗 **در جدول Authors**، فیلد atID (کد مولف) کلید اولیه است. چون، هیچ دو مؤلفی کد یکسانی ندارند.

🔗 **در جدول AutBook**، ترکیب فیلدهای ISBN و atID کلید اولیه است.

🔗 **در جدول PubBook**، ترکیب فیلدهای ISBN و PubID کلید اولیه است.

🔗 **در جدول GroupBook**، فیلد groupID کلید اولیه است.

۴-۱-۱. تعریف رابطه های بین جداول

همان طور که بیان گردید، یک جدول را به چند جدول کوچک تر می شکنیم تا افزونگی، بی نظمی و داده های NULL کاهش یابد. به عنوان مثال، اطلاعات مشتریان و فروش های انجام شده در دو جدول مجزا و مستقل ذخیره می شوند. اکنون، اگر پرس و جویی داشته باشیم که بخواهد اطلاعات را از این دو جدول بازیابی نماید باید ارتباط بین این جداول را برقرار نماییم. برای این منظور، حداقل به یک فیلد مشترک بین این دو جدول نیاز است. این فیلد یا فیلدها باید در یک جدول کلید اولیه یا فرعی و در جدول دیگر کلید خارجی^۳ باشد. سه نوع رابطه را می توان بین جداول تعریف کرد:

🔗 رابطه یک به چند 🔗 رابطه چند به چند 🔗 رابطه یک به یک

🔗 **رابطه یک به چند**، متداول ترین نوع رابطه در یک بانک اطلاعاتی رابطه ای است. در یک رابطه یک به چند، یک رکورد (سطر) در یک جدول (نظیر A) می تواند بیش از یک رکورد متناظر در جدول دیگر (مانند B) داشته باشد. ولی یک رکورد در جدول B تنها یک رکورد متناظر در جدول A دارد. به عنوان مثال، جدول محصولات و خرید دارای ارتباط یک به چند هستند.

🔗 **رابطه چند به چند**، یک رکورد در یک جدول (مانند A) می تواند بیش از یک رکورد متناظر در جدول دیگر (نظیر B) باشد و بالعکس.

^۳.Primary Key

^۲.Unique

^۲.Foreign Key

❖ **رابطه یک به یک**، هر رکورد جدول A نمی تواند بیش از یک رکورد متناظر در جدول B داشته باشد و نیز هر رکورد جدول B نمی تواند بیش از یک رکورد متناظر در جدول A داشته باشد. به عنوان مثال، جدول رانندگان و خودروها را در بانک اطلاعاتی حمل و نقل در نظر بگیرید. هر راننده، فقط و فقط راننده یک ماشین است و هر خودرو فقط یک راننده دارد.

بین جداول Books و AutBook، فیلد ISBN ارتباط را برقرار می کند. در این ارتباط، فیلد ISBN در جدول Book کلید اولیه است. اما، ISBN در جدول AutBook کلید خارجی است. این ارتباط چند به چند است. چون، یک کتاب می تواند چند مؤلف داشته باشد و چند کتاب می تواند توسط یک مؤلف تألیف شود. بین جداول Books و GroupBook، فیلد groupID ارتباط را برقرار می کند. در این ارتباط، فیلد groupID در جدول GroupBook کلید اولیه است، ولی این فیلد در جدول Books کلید خارجی می باشد. ارتباط بین این دو جدول یک به چند است. زیرا، چند کتاب می توانند به یک گروه اختصاص یابند. فیلد pubID برای برقرار ارتباط بین جداول Publishers و PubBook به کار می رود. در این ارتباط، فیلد موجود در جدول Publishers کلید اولیه است. اما، این فیلد در جدول PubBook کلید خارجی است. این ارتباط چند به چند است. زیرا، یک ناشر می تواند چند کتاب را انتشار دهد و چند ناشر نیز می توانند یک کتاب را با همکاری هم چاپ کنند.

۱-۳-۱. گروه های فایل

در بانک اطلاعاتی می توان گروه هایی برای فایل ها تعریف نمود تا با قرار دادن فایل های داده و ایندکس ها روی دیسک های مختلف کارائی سیستم را افزایش داد. دو نوع گروه فایل در SQL Server وجود دارد که عبارت اند از:

۱. **گروه فایل اصلی**^۴، حاوی فایل های داده اصلی^۲ و فایل هایی است که گروه فایل آن ها مشخص نگردید. همه صفحات^۳ اختصاص داده شده به جداول سیستم در این گروه قرار می گیرند. هر بانک اطلاعاتی حداقل یک گروه فایل اصلی دارد.

۲. **گروه فایل های تعریف شده توسط کاربر**^۵، با پارامتر FILEGROUP از طریق دستورهای CREATE DATABASE و ALTER DATABASE تعریف می شوند. عملکرد این دستورات را در ادامه می بینید.

در هنگام ایجاد گروه های فایل به نکات زیر توجه کنید:

۱. هر بانک اطلاعاتی که ایجاد می کنید، یک گروه فایل پیش فرض برای آن ایجاد می گردد (گروه فایل پیش فرض، همان گروه فایل اصلی است).

^۴. Primary File Group ^۲. Primary Data File ^۳. Pages ^۵. User Defined File Group
^۱. Backup ^۶. Recovery

۲. می‌توان به جای پشتیبان‌گیری^۵ و ترمیم^۶ کل بانک اطلاعاتی از گروه فایل پشتیبان گرفت یا گروه فایل را ترمیم نمود.

۳. گروه فایل امکان توزیع اشیا مختلف بانک اطلاعاتی را بر روی درایوهای دیسک فراهم می‌نماید.

۴. یک گروه فایل می‌تواند شامل چند فایل باشد.

۵. یک فایل بانک اطلاعاتی فقط در یک گروه فایل قرار می‌گیرد (یک فایل بانک اطلاعاتی نمی‌تواند در چند گروه فایل قرار گیرد).

۶. فایل کارنامه در هیچ گروه فایلی قرار نمی‌گیرد. زیرا، همان‌طور که بیان گردید، فایل کارنامه در فایل جداگانه‌ای قرار می‌گیرد.

۷. جداول، ایندکس‌ها و داده‌های نظیر text، image می‌توانند به یک گروه فایل، متناظر گردند.

۲-۳-۱. تغییر خواص بانک اطلاعاتی موجود

در بخش ۱-۳-۱ روش ایجاد بانک اطلاعاتی را دیدید. گاهی نیاز است خواص فایل‌های بانک اطلاعات موجود از قبیل اندازه، حداکثر اندازه و غیره را تغییر دهید. برای این منظور می‌توانید از دستور ALTER DATABASE به صورت زیر استفاده کنید:

ALTER DATABASE database_name

ADD FILE (ویژگی‌های فایل داده)

ADD FILEGROUP = نام گروه فایل

MODIFY FILE (ویژگی‌های فایل موجود)

MODIFY NAME = نام جدید

ADD LOG FILE (ویژگی‌های فایل کارنامه)

REMOVE FILE نام فایل منطقی

REMOVE FILEGROUP نام گروه فایل

database_name، نام بانک اطلاعاتی است که می‌خواهید ویژگی‌های آن را تغییر دهید.

 **مثال ۳-۱.** پرس‌وجویی که اندازه و حداکثر اندازه فایل Publish_data از بانک اطلاعاتی PublishDB را به ترتیب به ۱۵۰ و ۲۰۰ مگابایت تغییر می‌دهد.

ALTER DATABASE PublishDB

MODIFY FILE (NAME= Publish_data, SIZE= ۱۵۰, MAXSIZE= ۲۰۰)

۳-۳-۱. حذف بانک اطلاعاتی موجود

همان‌طور که دیدید، فایل‌های داده و کارنامه بانک اطلاعات فضای روی دیسک را اشغال می‌کنند. بنابراین، اگر بانک اطلاعاتی را نیاز نداشته باشید باید آن را حذف کنید تا فضای اشغال شده توسط آن به سیستم عامل برگردد. برای حذف بانک اطلاعات دستور DROP DATABASE به صورت زیر استفاده می‌شود:

```
DROP DATABASE database_name [۱,...,n];
```

database_name[۱,...,n] نام بانک‌های اطلاعاتی هستند که می‌خواهید حذف شوند. اگر بانک اطلاعاتی را حذف کنید، کلیه فایل‌های داده و کارنامه متعلق به آن حذف خواهند شد.

 **مثال ۴-۱.** پرس‌وجویی که بانک اطلاعات به نام myDatabase با ویژگی‌های زیر ایجاد می‌کند:

این بانک دارای یک فایل به نام my_data۱ است که در درایو C پوشه classDatabase با نام my_data۱.mdf قرار می‌گیرد. اندازه این فایل ۵MB تا حداکثر ۱۰۰MB به صورت ۵ مگابایت، ۵ مگابایت رشد می‌یابد.

```
CREATE DATABASE myDatabase
ON
( NAME = my_data۱,
  FILENAME = 'c:\classDatabase\my_data۱.mdf',
  SIZE = ۵MB, MAXSIZE = ۱۰۰MB, FILEGROWTH = ۵MB
)
GO
```

 **مثال ۵-۱.** پرس‌وجویی که فایلی به نام my_data۲ در پوشه classDatabase درایو C با نام my_data۲.mdf به بانک اطلاعاتی myDatabase اضافه می‌کند. اندازه این فایل ۵ مگابایت است و تا حداکثر ۱۰۰ مگابایت به صورت ۵ مگابایت، ۵ مگابایت می‌تواند رشد یابد.

```
ALTER DATABASE myDatabase
ADD FILE ( NAME = my_data۲, FILENAME='c:\classDatabase\my_data۲.mdf', SIZE=۵MB, MAXSIZE=۱۰۰MB, FILEGROWTH = ۵MB
)
GO
```

 **مثال ۶-۱.** پرس‌وجویی که گروه فایلی به نام myFileGroup را به بانک اطلاعاتی myDatabase اضافه می‌کند.

```
ALTER DATABASE myDatabase
ADD FILEGROUP myFileGroup
GO
```

 **مثال ۷-۱.** پرس‌وجویی که فایل my_data۳ در پوشه classDatabase درایو C با نام my_data۳.mdf را به گروه myFileGroup بانک اطلاعاتی myDatabase اضافه می‌کند. این فایل ۴MB است که می‌تواند تا ۵۰MB با گام ۵MB رشد کند.

```
ALTER DATABASE myDatabase
ADD FILE (NAME=my_data۳, FILENAME=N'c:\classdatabase\my_data۳.mdf', SIZE=۴MB, MAXSIZE=۵۰MB, FILEGROWTH=۵MB)
TO FILEGROUP myFileGroup
GO
```

 **مثال ۸-۱.** پرس‌وجویی که فایل my_data۲ را از بانک اطلاعاتی myDatabase حذف می‌کند.

```
ALTER DATABASE myDatabase
  REMOVE FILE my_data۲
GO
```

 مثال ۹-۱. پرس‌وجویی که اندازه فایل my_data۱ از بانک اطلاعاتی myDatabase را به ۱۰ مگا بایت تغییر می‌دهد.

```
ALTER DATABASE myDatabase
  MODIFY FILE (NAME = my_data۱, SIZE = ۱۰MB)
GO
```

۱-۴-۱. ایجاد جدول

برای ایجاد جداول بانک اطلاعاتی می‌توانید از دستور CREATE TABLE به صورت زیر استفاده کنید:

```
CREATE TABLE table_name
  (Column۱ type۱ Constraint۱,
   Column۲ type۲ Constraint۲,
   .
   .
   .
   Columnn typen Constraintn)
```

در این ساختار table_name، نام جدولی است که می‌خواهید ایجاد کنید. Column۱ تا Columnn، نام فیلدهای (ستون‌های) جدول را تعیین می‌کنند. type۱ تا typen، نوع ستون‌های Column۱ تا Columnn را مشخص می‌کنند و Constraint۱ تا Constraintn به ترتیب محدودیت‌هایی را تعیین می‌کنند که باید بر روی ستون‌های Column۱ تا Columnn اعمال شوند. اگر ستونی محدودیت نداشته باشد، می‌توانید بخش Constraint آن ستون را حذف کنید.

 مثال ۲۵-۱. پرس‌وجویی که جدول GroupBook را ایجاد می‌کند (مشخصات فیلدهای جدول GroupBook در جدول ۱-۱ آمده است).

```
USE PublishDB
GO
CREATE TABLE GroupBook
(
  groupID    varchar(۱) PRIMARY KEY,
  groupName  varchar(۳۰) NOT NULL
)
GO
```

همان‌طور که در این دستورات می‌بینید، فیلد groupID کلید اولیه تعریف شده است و groupName به صورت NOT NULL تعریف گردید تا مقدار تهی را نپذیرد.

 مثال ۲۹-۱. پرس‌وجویی که جدول AutBook را ایجاد می‌کند.

```
CREATE TABLE AutBook
(
```

```

ISBN      varchar(۱۰) ,
atID      varchar(۱۰) ,
Payment   decimal(۱۸, ۰),
PRIMARY KEY (ISBN, atID),
CONSTRAINT FK_ISBN FOREIGN KEY (ISBN) REFERENCES Books (ISBN) ,
CONSTRAINT FK_atID FOREIGN KEY (atID) REFERENCES
Authors (atID)
)
GO

```

۵- ۱. مسائل حل شده

از آنجایی که نام این کتاب آزمایشگاه پایگاه داده است. لذا در این بخش یک بانک اطلاعاتی واقعی تر بیان گردیده، تمام مسائل حل شده در کل کتاب از این بانک اطلاعاتی استفاده می کند. این بانک اطلاعاتی برای خرید، فروش و بخش محصولات مختلف به کار می رود. برخی از مراحل بیان شده طراحی پایگاه داده در درس در زیر آمده است:

۱. **تعیین جداول مورد نیاز**، این بانک اطلاعاتی اطلاعات مشتریان، حساب ها و غیره را نگهداری می کند. جداول در نظر گرفته شده برای این بانک در زیر آمده اند:

🚩 **جدول Customer**، اطلاعات مشتریان از قبیل کد مشتری، نام مشتری، نام خانوادگی مشتری، کد ملی و غیره را نگهداری می کند (جدول ۳- ۱).

🚩 **جدول City**، اطلاعات شهرها از قبیل کد شهر، نام شهر و کد کاربر را نگهداری می نماید (جدول ۳- ۱).

جدول ۳- ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات.						
نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
Customer (مشتریان)	ID	کد مشتری	Varchar(۶)	بله	نه	نه
	firstName	نام مشتری	Varchar(۳۰)	نه	نه	نه
	lastName	نام خانوادگی مشتری	Varchar(۳۰)	نه	نه	نه
	nationalCode	کد ملی	Varchar(۱۰)	نه	نه	نه
	Tel	شماره تلفن	Varchar(۱۵)	نه	بله	بله
	Mobile	شماره موبایل	Varchar(۱۵)	نه	بله	نه
	Address	آدرس	Varchar(۵۰)	نه	بله	نه
	activityType	نوع فعالیت	Varchar(۵۰)	نه	بله	نه
	cityID	کد شهر	Varchar(۶)	نه	بله	بله

نه	بله	نه	Varchar(٦)	کد استان	stateID	
نه	بله	نه	Varchar(٥٠)	ایمیل	Email	
نه	بله	نه	Bit	خوش حسابی	goodAccount	
نه	بله	نه	Decimal(١٨,٠)	میزان اعتبار	Credit	
بله	بله	نه	Varchar(٦)	کد کاربر	userCode	
نه	نه	بله	Varchar(٦)	کد کاربر	<u>userCode</u>	Login ^١ (کاربران)
نه	نه	نه	Varchar(٥٠)	نام کاربر	userName	
نه	بله	نه	Varchar(٥٠)	کلمه عبور	password	
نه	نه	بله	Varchar(٦)	کد شهر	<u>cityID</u>	City (شهرها)
نه	نه	نه	Varchar(٥٠)	نام شهر	cityName	
بله	نه	نه	Varchar(٦)	کد کاربر	userCode	

ادامه جدول ۳ - ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات.						
نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
State (استان ها)	stateID	کد استان	Varchar(۶)	بله	نه	نه
	stateName	نام استان	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
Store (انبار)	storeID	کد انبار	Varchar(۶)	بله	نه	نه
	storeName	نام انبار	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	نه
Merchandise (کالا)	merchandiseID	کد کالا	Varchar(۶)	بله	نه	نه
	merchandiseGroupID	کد گروه کالا	Varchar(۶)	نه	نه	بله
	merchandiseName	نام کالا	Varchar(۶)	نه	نه	نه
	consumerPrice	قیمت مصرف کننده	Numeric(۱۸, ۰)	نه	بله	نه
	buyPrice	قیمت خرید	Numeric(۱۸, ۰)	نه	بله	نه
	pocketSellingPrice	قیمت فروش بسته ای	Numeric(۱۸, ۰)	نه	بله	نه
	changeSellingPrice	قیمت تعویض	Numeric(۱۸, ۰)	نه	بله	نه
	hav	موجودی	Numeric(۱۸, ۰)	نه	بله	نه
	peritionAmount	حد سفارش	Numeric(۱۸, ۰)	نه	بله	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
merchandiseGroup (گروه کالا)	merchandiseGroupID	کد گروه کالا	Varchar(۶)	بله	نه	نه
	merchandiseGroupName	نام گروه کالا	Varchar(۵۰)	نه	نه	نه
	unit	واحد	Varchar(۵۰)	نه	بله	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
BankSave (بانک)	fishNo	شماره فیش	Varchar(۱۵)	بله	نه	نه
	hesabNo	شماره حساب	Varchar(۱۵)	نه	نه	نه
	Date	تاریخ	Varchar(۸)	نه	نه	نه
	mablagTake	مبلغ برداشتی	Decimal(۱۸, ۰)	نه	نه	نه
	mablagSave	مبلغ واریزی	Decimal(۱۸, ۰)	نه	نه	نه
	sellerID	کد فروشنده	Varchar(۶)	نه	نه	بله
	comment	توضیحات	Varchar(۶)	نه	بله	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
BuyInvoice (فاکتور خرید)	invoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	بله	نه	نه
	Date	تاریخ فاکتور	Varchar(۸)	نه	نه	نه
	ID	شماره مشتری	Varchar(۶)	نه	نه	بله

ادامه جدول ۳ - ۱ بانک اطلاعاتی پخش، خرید و فروش محصولات.

نام جدول	نام فیلد	هدف	نوع	کلید اصلی	تهی	کلید خارجی
BuyInvoice (فاکتور خرید)	buyType	نوع خرید	Varchar(۵۰)	نه	بله	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
buyInvoice۱ (ریز فاکتور خرید)	invoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(۶)	بله	نه	بله
	Price	قیمت	Numeric(۱۸, ۰)	نه	نه	نه
	Discount	تخفیف	Numeric(۱۸, ۰)	نه	نه	نه
	Amount	تعداد	Numeric(۱۸, ۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
sellInvoice (فاکتور فروش)	invoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	بله	نه	نه
	Date	تاریخ فاکتور	Varchar(۸)	نه	نه	نه
	ID	شماره مشتری	Varchar(۶)	نه	نه	بله
	sellType	نوع فروش	Varchar(۵۰)	نه	بله	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
sellInvoice۱ (ریز فاکتور فروش)	invoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(۶)	بله	نه	بله
	Price	قیمت	Numeric(۱۸, ۰)	نه	نه	نه
	Discount	تخفیف	Numeric(۱۸, ۰)	نه	نه	نه
	Amount	تعداد	Numeric(۱۸, ۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
Hessab (حساب بانکی)	hessabNo	شماره حساب	Varchar(۱۵)	بله	نه	نه
	hessabName	نام حساب	Varchar(۵۰)	نه	نه	نه
	bankName	نام بانک	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
Seller (فروشندهگان)	sellerID	شماره فروشنده	Varchar(۶)	بله	نه	نه
	sellerName	نام فروشنده	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
Receipt (رسید)	recepitID	شماره رسید	Varchar(۶)	بله	نه	نه
	storeID	کد انبار	Varchar(۶)	نه	نه	بله
	Date	تاریخ	Varchar(۸)	نه	نه	نه
	InvoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	نه	نه	بله
	Amount	تعداد	Numeric(۱۸, ۰)	نه	نه	نه

ادامه جدول ۳-۱ بانک اطلاعاتی پخش، خرید و فروش محصولات.						
نام جدول	نام فیلد	هدف	نوع	کلید اصلی	تهی	کلید خارجی
Recipt (رسید)	Render	تحويل گیرنده	Varchar(۵۰)	نه	نه	نه
	Teller	تحويل دهنده	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
Receipt۲ (ریز کالا رسید شده)	receiptID	شماره رسید	Varchar(۶)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(۶)	بله	نه	بله
	Amount	تعداد کالا	Numeric(۱۸,۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
Order (حواله)	receiptID	شماره حواله	Varchar(۶)	بله	نه	نه
	storeID	کد انبار	Varchar(۶)	نه	نه	بله
	Date	تاریخ	Varchar(۸)	نه	نه	نه
	invoiceNumeral	شماره فاکتور	Decimal(۱۸,۰)	نه	نه	بله
	Amount	تعداد	Numeric(۱۸,۰)	نه	نه	نه
	Render	تحويل گیرنده	Varchar(۵۰)	نه	نه	نه
	Teller	تحويل دهنده	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله
	receiptID	شماره حواله	Varchar(۶)	بله	نه	بله
Order۲ (ریز کالای حواله شده)	merchandiseID	کد کالا	Varchar(۶)	بله	نه	بله
	Amount	تعداد کالا	Numeric(۱۸,۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(۶)	نه	نه	بله

🚩 **جدول Login**، اطلاعات کاربران از قبیل کد کاربر، نام کاربر و کلمه عبور را نگهداری می کند (جدول ۳-۱).

🚩 **جدول State**، اطلاعات استان ها از قبیل کد استان، نام استان و کد کاربر را نگهداری می کند (جدول ۳-۱).

🚩 **جدول Store**، اطلاعات انبار از قبیل کد انبار، نام انبار و کد کاربر را نگهداری می کند (جدول ۳-۱).

🚩 **جدول Merchandise**، اطلاعات کالا از قبیل کد کالا، نام کالا، کد گروه کالا، قیمت مصرف کننده، قیمت خرید و غیره را نگهداری می کند (جدول ۳-۱).

🚩 **جدول MerchandiseGroup**، اطلاعات گروه کالا از قبیل کد گروه، نام گروه، واحد و کد کاربر را نگهداری می نماید (جدول ۳-۱).

🚩 **جدول BankSave**، اطلاعات برداشت و واریزها به بانک از قبیل شماره فیش، شماره حساب، تاریخ، مبلغ برداشت، مبلغ واریز و غیره را نگهداری می نماید (جدول ۳ - ۱).

🚩 **جدول buyInvoice**، اطلاعات فاکتور از قبیل شماره فاکتور، تاریخ فاکتور، شماره مشتری، نوع خرید و کد کاربر را ذخیره می کند (جدول ۳ - ۱).

🚩 **جدول ۱ buyInvoice**، اطلاعات ریز فاکتورها از قبیل شماره فاکتور، کد کالا، قیمت، تخفیف، تعداد و کد کاربر را ذخیره می کند (جدول ۳ - ۱).

🚩 **جدول SellInvoice**، اطلاعات فاکتور فروش از قبیل شماره فاکتور، تاریخ، کد مشتری و غیره را نگهداری می کند (جدول ۳ - ۱).

🚩 **جدول ۱ SellInvoice**، اطلاعات ریز فاکتور فروش از قبیل شماره فاکتور، کد کالا، قیمت، تخفیف، تعداد و کد کاربر را نگهداری می نماید (جدول ۳ - ۱).

🚩 **جدول Hessab**، اطلاعات حساب ها از قبیل شماره حساب، نام بانک، نام حساب و کد کاربر را ذخیره می نماید (جدول ۳ - ۱).

🚩 **جدول Seller**، اطلاعات فروشندگان از قبیل شماره فروشنده، نام فروشنده و کد کاربر را ذخیره می نماید (جدول ۳ - ۱).

🚩 **جدول Receipt**، اطلاعات رسیده ها از قبیل شماره سند، کد انبار، تاریخ، شماره فاکتور و غیره را ذخیره می نماید (جدول ۳ - ۱).

🚩 **جدول ۲ Receipt**، اطلاعات ریز رسیده ها از قبیل شماره رسید، کد کالا، تعداد و کد کاربر را ذخیره می کند (جدول ۳ - ۱).

🚩 **جدول Order**، اطلاعات حواله ها از قبیل شماره حواله، کد انبار، تاریخ و غیره را نگهداری می کند (جدول ۳ - ۱).

🚩 **جدول ۲ Order**، اطلاعات ریز حواله ها از قبیل شماره حواله، کد کالا، تعداد و کد کاربر را نگهداری می کند (جدول ۳ - ۱).

۲. تعیین فیلدهای کلید اولیه

همان طور که بیان گردید، در هر جدول باید یک یا چند فیلد آن به عنوان کلید اولیه باشد. کلیدهای اولیه بانک اطلاعاتی خرید، فروش و پخش محصولات در زیر آمده است:

🚩 **در جدول Customer**، فیلد ID کلید اولیه است. زیرا، هیچ دو مشتری نمی توانند شماره مشتری یکسان داشته باشند.

🚩 **در جدول Login**، فیلد userCode کلید اولیه است. چون، هیچ دو کاربری کد یکسان ندارند.

🚩 **در جدول City**، فیلد cityID کلید اولیه است. چون هیچ دو شهری کد یکسان ندارند.

🚩 **در جدول State**، فیلد stateID کلید اولیه است. زیرا، کد استان نمی تواند تکراری باشد.

🚩 **در جدول Store**، فیلد storeID کلید اولیه است. چون، کد انبار یکتا است.

🚩 **در جدول Merchandise**، فیلد merchandiseID کلید اولیه است. چون، هیچ دو کالایی نمی توانند کد یکسان داشته باشند.

🚩 **در جدول MerchandiseGroup**، فیلد merchandiseGroupID کلید اولیه است. زیرا، کد گروه کالا نمی تواند تکراری باشد.

🚩 **در جدول BankSave**، فیلد fishNo کلید اولیه است. زیرا، هیچ دو فیشی نمی توانند شماره تکراری داشته باشند.

🚩 **در جداول buyInvoice و SellInvoice**، فیلد invoiceNumeral کلید اولیه است. زیرا، هیچ دو فاکتور خرید و فروشی نباید شماره تکراری داشته باشند.

🚩 **در جداول buyInvoice^۱ و SellInvoice^۱**، ترکیب فیلدهای merchandiseID، invoiceNumeral کلید اولیه است.

🚩 **در جدول Hesab**، فیلد hesabNo کلید اولیه است، چون هیچ دو حسابی نمی توانند شماره تکراری داشته باشند.

🚩 **در جدول Seller**، فیلد sellerID کلید اولیه است. چون کد هیچ دو فروشنده نمی تواند یکی باشد.

🚩 **در جدول Order و Receipt**، به ترتیب receiptID و orderID کلید اولیه هستند.

🚩 **در جدول Receipt^۲**، ترکیب فیلدهای receiptID و merchandiseID کلید اولیه است.

🚩 **در جدول Order^۲**، ترکیب فیلدهای orderID و merchandiseID کلید اولیه است.

۳. تعیین ارتباط بین جداول

یکی از بخش های بسیار مهم بانک اطلاعاتی، تعیین فیلد ارتباطی بین جداول و نوع ارتباط است. ارتباط بین جداول در بانک اطلاعاتی خرید، فروش و بخش محصولات در زیر آمده است:

🚩 بین جداول Customer و Login^۱، فیلد ارتباط userCode است. این فیلد در جدول Customer کلید خارجی، ولی در جدول Login کلید اولیه است.

🚩 بین جداول Customer و City، فیلد ارتباط cityID است. این فیلد در جدول Customer کلید خارجی است.

🚩 بین جداول Customer و State، فیلد ارتباط stateID است که در جدول Customer کلید خارجی است. چون در جدول State کلید اولیه می باشد.

🚩 بین جداول Login^۱ و تمام جدول، userCode ارتباط را برقرار می کند که در جدول Login^۱، فیلد userCode کلید اولیه است، اما، در بقیه جداول کلید خارجی می باشد.

🚩 بین جداول Store، Receipt و Order، فیلد ارتباط storeID است. این فیلد در جدول Store کلید اولیه است. اما در جدول Receipt و Order کلید خارجی می باشد.

🌈 بین جدول Merchandise و جداول Order، Receipt، buyInvoice و SellInvoice فیلد merchandiseID ارتباط را برقرار می‌کند. این فیلد در جدول Merchandise کلید اولیه است. ولی، در بقیه جداول کلید خارجی می‌باشد.

🌈 بین جداول Merchandise و MerchandiseGroup فیلد merchandiseGroupID ارتباط را برقرار می‌کند. این فیلد در جدول Merchandise کلید خارجی است.

🌈 بین جداول BankSave و Hesab فیلد HesabNo ارتباط را برقرار می‌کند. فیلد HesabNo در جدول BankSave کلید خارجی است.

۶-۱. دستور کار آزمایشگاه

قبل این که دستور کار آزمایشگاه را شروع کنیم، بانک اطلاعاتی آژانس حمل و نقل را در نظر بگیرد. در این بانک اطلاعاتی، مشتریان (Customers) زنگ می‌زنند، سفارشی (انتقال وسائل یا خودشان از یک مکان به مکان دیگر) را می‌دهند. سپس خودرو یا خودروهایی که هر یک از آن خودروها یک راننده دارند، این سفارشات را انجام می‌دهند. جداول و فیلدهای این بانک اطلاعاتی در جدول ۴-۱ آمده است. اکنون به سوالات زیر پاسخ دهید.

۱. ارتباط بین جداول رانندگان (Drivers) و جدول ماشین‌ها (Cars) از چه نوعی است؟

۲. ارتباط بین مشتریان (Customers) و سفارشات (Orders) از چه نوعی است؟

۳. ارتباط بین جداول رانندگان (Drivers) و سفارشات (Orders) از چه نوعی است؟

۴. ارتباط بین جداول Cars و Orders از چه نوعی است؟

جدول ۴-۱ بانک اطلاعاتی آژانس حمل و نقل.						
نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
Cars (ماشین‌ها)	carID	شماره ماشین	Varchar(۱۰)	بله	نه	نه
	carType	نوع ماشین	Varchar(۳۰)	نه	نه	نه
	capacity	ظرفیت	Decimal(۱۸, ۰)	نه	نه	نه
	unit	واحد	Varchar(۳۰)	نه	نه	نه
	drID	شماره راننده	Varchar(۱۰)	نه	نه	بله
Drivers (رانندگان)	drID	شماره راننده	Varchar(۱۰)	بله	نه	نه
	drFname	نام راننده	Varchar(۲۰)	نه	نه	نه
	drLname	نام خانوادگی راننده	Varchar(۲۰)	نه	نه	نه
	degree	درجه گواهی نامه	Varchar(۲۰)	نه	بله	نه
Customers (مشتریان)	cID	شماره مشتری	Varchar(۱۰)	بله	نه	نه
	cFname	نام مشتری	Varchar(۲۰)	نه	نه	نه
	cLname	نام خانوادگی مشتری	Varchar(۲۰)	نه	نه	نه

نه	بله	نه	Varchar(۱۵)	شماره تلفن	Tel	Orders (سفارشات)
نه	بله	نه	Varchar(۳۰)	شهر مشتری	City	
نه	نه	بله	Varchar(۱۰)	شماره سفارش	oID	
بله	بله	نه	Varchar(۱۰)	شماره راننده	drID	
بله	بله	نه	Varchar(۱۰)	شماره مشتری	clID	
نه	نه	نه	Varchar(۳۰)	شهر مقصد	desCity	
نه	نه	نه	Decimal(۱۸, ۰)	کرایه	Freight	

۵. بانک اطلاعاتی آژانس حمل و نقل را با نام TransportationAgency ایجاد کنید. این بانک دارای فایل‌های زیر باشد:

🚩 فایل **t_agency_data.mdf**، دارای حجم سه مگابایت و حداکثر تا ۱۰ مگابایت رشد می‌کند که درصد رشد آن ۲۰ است.

🚩 فایل **t_agency_log.ldf**، دارای حجم سه مگابایت و حداکثر تا ۱۰ مگابایت رشد می‌کند که درصد رشد آن ۵۰ است.

تکته: فایل t_agency_data.mdf در مسیر database درایو C و فایل t_agency_log.ldf در مسیر classDatabase درایو C ایجاد می‌شود.
--

۶. فایل کارنامه **t_agency۱_log.ldf** را با اندازه ۵MB به بانک اطلاعاتی TransportationAgency در پوشه Database درایو C اضافه کنید، به طوری که حداکثر تا ۱۰۰MB بتواند رشد داشته باشد و رشد فایل ۵MB در هر مرتبه باشد.

۷. فایل داده **t_agency۱_data.mdf** را با اندازه ۵MB به بانک اطلاعاتی TransportationAgency در پوشه classDatabase درایو C اضافه کنید، به طوری که حداکثر اندازه آن ۵۰MB شود و رشد آن ۵MB در هر مرحله باشد.

۸. فایل **t_agency۱_data.mdf** را از بانک اطلاعاتی TransportationAgency حذف کنید.

۹. جدول Drivers را ایجاد کنید.

۱۰. فیلدهای Email (از نوع varchar(۲۰)) و Photo (از نوع image) را به جدول Drivers اضافه کنید.

۱۱. فیلد Email را از جدول Drivers حذف کنید.

۱۲. جدول Cars را ایجاد کنید.

۱۳. ارتباط بین جداول Drivers و Cars را ایجاد کنید.

۱۴. فیلد Photo را از جدول Drivers حذف کنید.

۱۵. جدول Customers را ایجاد کنید.

۱۶. فایل **t_agency۱_log.ldf** را از بانک اطلاعاتی TransportationAgency حذف کنید.

۱۷. جدول Orders را ایجاد کنید.

۱۸. ارتباط بین Orders با جداول Drivers و Customers را برقرار کنید.

۱۹. در بانک اطلاعاتی که اطلاعات را به زبان فارسی ذخیره می کند، برای ذخیره اطلاعات فیلدهای زیر از چه انواعی استفاده می شوند:

- الف: شماره دانشجویی ب: نمره ج: مدرک استاد د: کد ملی
پ: عکس دانشجو ت: اطلاعات XML خ: تاریخ فارغ التحصیلی چ: قیمت کالا
۲۰. جدولی به نام Test با فیلدهای x1 و x2 از نوع int به بانک TransportationAgency اضافه کنید.
۲۱. فیلد x2 را در جدول Test کلید اولیه تعیین کنید.
۲۲. جدول Test را حذف کنید.

۸-۱. تمرین ها

جداول موجود در جدول ۵-۱ را در نظر بگیرید. اکنون به سوالات زیر پاسخ دهید:

۱. بانک اطلاعاتی به نام Accounting ایجاد کنید که دارای یک فایل داده و یک فایل کارنامه باشد. مسیر فایل داده و کارنامه تفاوت داشته باشد.
۲. یک فایل داده جدید به بانک Accounting اضافه کنید.
۳. کلیدهای خارجی جداول موجود در جدول ۵-۱ را تعیین کنید.
۴. ارتباط بین جداول را تعیین کرده و نوع ارتباط را مشخص نمایید.
۵. جداول را به همراه ارتباط بین آنها در بانک Accounting ایجاد کنید.

جدول ۵-۴ توزیع فیلدهای جداول سیستم حسابداری.				
نام جدول	نام فیلد	نوع فیلد	اندازه	اولیه
جدول کل	کد کل	varchar	۳	بله
	نام کل	varchar	۵۰	نه
	کد نوع حساب	varchar	۱	نه
جدول معین	کد کل	varchar	۳	بله
	کد معین	varchar	۳	بله
	نام معین	varchar	۵۰	نه
جدول تفضیلی	کد کل	varchar	۳	بله
	کد معین	varchar	۳	بله
	کد تفضیلی	varchar	۶	بله
	نام تفضیلی	varchar	۵۰	نه

سربلگ سند	شماره سند تاریخ سند شرح سند تعداد ضمایم سند کد نوع سند	decimal varchar varchar decimal varchar	۵ ۱۰ ۲۵۵ ۳ ۱	بله نه نه نه نه
جدول ریز سند	شماره سند کد کل کد معین کد تفصیلی شرح ردیف سند مبلغ بدهکار مبلغ بستانکار کد هزینه کد پروژه	decimal varchar varchar varchar varchar decimal decimal varchar varchar	۵ ۳ ۳ ۶ ۲۵۵ ۱۸ ۱۸ ۶ ۶	بله بله بله بله نه نه نه نه نه
جدول هزینه	کد هزینه نام هزینه	varchar varchar	۶ ۵۰	بله نه
جدول پروژه	کد پروژه نام پروژه	varchar varchar	۶ ۵۰	بله نه
جدول نوع حساب	کد نوع حساب نام نوع حساب	varchar varchar	۱ ۵۰	بله نه
جدول نوع سند	کد نوع سند نام نوع سند	varchar varchar	۱ ۵۰	بله نه
جدول گروه حساب	کد گروه حساب نام گروه حساب	varchar varchar	۲ ۵۰	بله نه

ورود، ویرایش، حذف و بازیابی اطلاعات

در فصل اول، با چند مثال ساده روش ایجاد بانک اطلاعات و جداول آن را با استفاده از دستور SQL دیدید. همان طور که بیان گردید، بعد از ایجاد جداول در بانک اطلاعات باید بتوان داده‌ها را در آن وارد نمود، آن‌ها را ویرایش کرد، یا داده‌های اضافی را حذف نمود و در صورت نیاز به اطلاعات خاصی، آن‌ها را بازیابی کرد. بنابراین، در این فصل دستورات INSERT، UPDATE، DELETE و SELECT که برای دست کاری داده‌های جدول به کار می‌روند را می‌آموزیم.

قبل از این که به ورود، ویرایش و بازیابی اطلاعات بپردازیم، به جداول بانک اطلاعات PublishDB اطلاعات اولیه می‌دهیم تا خروجی‌هایی که از دستورات گرفته می‌شود، همان خروجی باشد که شما با اجرای دستورات می‌گیرید. بنابراین، ابتدا اطلاعات جداول بانک اطلاعاتی PublishDB را با توجه به شکل ۱-۲ در نظر بگیرید. این اطلاعات را در ادامه وارد جداول خواهیم کرد. سپس بازیابی اطلاعات را از این جداول بیان می‌کنیم.

اطلاعات جدول Publishers							
pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
۰۱	فناوری نوین	۰۱۱۱۲۵۶۶۸۷	www.fanavarienovin.net	بابل	رمضان	عباس نژاد	۱۵
۰۲	دانش نگار	۰۲۱۶۶۴۰۰۲۲۰	www.danshiran.ir	تهران	محمد رضا	حاتمی	۱۰۰
۰۳	دانشگاه مازندران			بابلسر	رضا	امینی	

اطلاعات جدول Authors							
atID	atFname	atLname	Age	Ranking	Email	Mobile	SumPayment
۰۱	رمضان	عباس نژاد	۴۲	فوق لیسانس	fanavarienovin@yahoo.com	۰۹۱۱۱۱۶۲۱۲	۲۸۰۰۰۰۰۰
۰۲	یوسف	عباس نژاد	۲۴	لیسانس	yousef_dssht@yahoo.com	۰۹۱۱۲۱۸۵۴۲۶	
۰۳	باقر	رحیم پور	۳۲	فوق لیسانس			۱۰۰۰۰۰۰۰
۰۴	مرتضی	پویان	۲۷	فوق لیسانس			
۰۵	سیدحجت	جلیلی	۳۵	لیسانس			۶۰۰۰۰۰۰۰

						اله	
--	--	--	--	--	--	-----	--

اطلاعات کتاب‌ها (Books).						
groupID	printNO	editNO	Price	page	Title	ISBN
۰۲	۱	۱	۷۳۰۰۰	۲۵۶	اصول طراحی پایگاه داده	۶۰۰۹۱۴۱۳۹۵
۰۱	۱	۱	۱۳۰۰۰۰	۴۳۲	طراحی سیستم‌های شی گرا	۶۰۰۹۲۲۵۴۲۲
۰۱	۲	۱	۵۷۰۰۰	۲۱۶	حل مسائل C++	۶۰۰۹۱۴۱۳۰۲
	۲	۱	۶۲۰۰۰	۲۰۰	حل مسائل C#	۶۰۰۹۱۴۱۳۴۰
۰۳	۱	۱	۶۵۰۰۰	۱۹۲	امنیت شبکه	۶۰۰۹۱۴۱۳۸۸
۰۴	۱	۱	۷۵۰۰۰	۲۷۲	مدیریت استراتژیک	۶۰۰۹۲۲۵۴۳۹
۰۱	۱	۱	۴۰۰۰۰	۳۰۴	رهیافت C++	۷۰۰۹۲۲۵۴۳۱

اطلاعات جدول GroupBook.	
groupName	groupID
برنامه‌نویسی	۰۱
پایگاه داده	۰۲
شبکه‌های رایانه‌ای	۰۳
فناوری اطلاعات	۰۴
گرافیک رایانه‌ای	۰۵

اطلاعات جدول AutBook.		
Payment	atID	ISBN
۵۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۹۵
۵۰۰۰۰۰۰	۰۳	۶۰۰۹۱۴۱۳۹۵
۵۰۰۰۰۰۰	۰۱	۶۰۰۹۲۲۵۴۲۲
۵۰۰۰۰۰۰	۰۳	۶۰۰۹۲۲۵۴۲۲
۶۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۰۲
۶۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۴۰
۶۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۸۸
۶۰۰۰۰۰۰	۰۵	۶۰۰۹۲۲۵۴۳۹
۶۰۰۰۰۰۰	۰۱	۷۰۰۹۲۲۵۴۳۱

اطلاعات جدول PubBook.		
Payment	PubID	ISBN
۶۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۹۵
۸۰۰۰۰۰۰	۰۱	۶۰۰۹۲۲۵۴۲۲
۷۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۴۰
۶۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۸۸
۵۰۰۰۰۰۰	۰۱	۶۰۰۹۲۲۵۴۳۹
۳۵۰۰۰۰۰	۰۱	۷۰۰۹۲۲۵۴۳۱
۳۵۰۰۰۰۰	۰۲	۷۰۰۹۲۲۵۴۳۱
۶۰۰۰۰۰۰	۰۱	۶۰۰۹۱۴۱۳۰۲

۲ - ۱

شکل

اطلاعات جدول بانک اطلاعاتی PublishDB.

۲-۱. دستور INSERT

این دستور برای اضافه کردن رکورد به جداول بانک اطلاعات به کار می‌رود و به صورت زیر استفاده می‌شود:

```

INSERT [INTO] {table_name|view_name}
list) ] {VALUES ({DEFAULT|NULL|expression} [,... ]_ {[(column
table_|derived
|execute_statement
} }
|DEFAULT VALUES

```

پارامترهای این دستور در زیر آمده‌اند:

table_name نام جدولی را تعیین می‌کند که اطلاعات باید وارد آن شود.

view_name نام دیدی را تعیین می‌کند که اطلاعات باید وارد آن شود.

column_list لیست فیلدهایی را تعیین می‌کند که اطلاعات بخش VALUES باید وارد آن‌ها شود.

DEFAULT مقدار پیش‌فرض فیلد جدول یا دید را مشخص می‌نماید. اگر مقدار فیلد از طریق دستور INSERT وارد نگردد، این مقدار در فیلد قرار می‌گیرد.

NULL مقدار تهی را وارد فیلد جدول یا دید می‌نماید.

expression حاصل یک عبارت را وارد فیلد جدول یا دید می‌نماید (برای مقدار دادن به فیلدهای محاسباتی مفید است).

derived_table خروجی یک دستور SELECT را وارد جدول می‌نماید. در ادامه دستور SELECT را می‌آموزیم.

execute_statement خروجی یک دستور SELECT یا READTEXT را به جدول یا دید اضافه می‌کند.

عملگرهای بیتی

عملگرهایی هستند که بین دو عملوند از نوع صحیح قرار گرفته، عملیات بیتی از قبیل «و» بیتی، «یا» بیتی و «یا انحصاری» بیتی را انجام می‌دهند (جدول ۲-۲).

عملگرهای رابطه‌ای

عملگرهایی هستند که دو عبارت (عملوند) را با هم مقایسه کرده، نتیجه مقایسه را به صورت منطقی برمی‌گردانند (جدول ۲-۳).

جدول ۲-۲ عملگرهای بیتی در SQL				
عملگر	نام	مثال	نتیجه	هدف
&	«و» بیتی	۱۷۰ & ۷۵	۱۰	عملوند اول و دوم را بیت به بیت «و» منطقی می‌نماید.
	«یا» بیتی	۱۷۰ ۷۵	۲۳۵	عملوند اول و دوم را بیت به بیت «یا» منطقی می‌نماید.
^	«یا بیتی» انحصاری (XOR)	۱۷۰ ^ ۷۵	۲۲۵	عملوند اول و دوم را بیت به بیت با هم XOR می‌نماید.

عملگرهای منطقی

عملگرهایی هستند که بین دو عبارت منطقی قرار گرفته، نتیجه دو عبارت منطقی را بررسی می‌کنند. عبارات منطقی دارای یکی از سه ارزش درستی^۵، نادرستی^۶ و یا تهی^۷ هستند. این عملگرها در جدول ۲-۴ آمده‌اند.

^۵.True

^۶.False

^۷.Null

جدول ۳-۲ عملگرهای رابطه‌ای در SQL				
عملگر	نام	مثال	نتیجه	هدف
=	تساوی	$۱۰ = ۵$	نادرست	دو عملوند را با یکدیگر مقایسه می‌کند، در صورتی که با هم برابر باشند، نتیجه درست است.
>	بزرگ‌تر	$۱۰ > ۵$	درست	اگر عملوند اول بزرگ‌تر از عملوند دوم باشد، نتیجه درست است.
<	کوچک‌تر	$۱۰ < ۵$	نادرست	اگر عملوند اول کوچک‌تر از عملوند دوم باشد، نتیجه درست است.
>=	بزرگ‌تر مساوی	$۱۰ >= ۷$	درست	اگر عملوند اول بزرگ‌تر یا مساوی عملوند دوم باشد، نتیجه درست است.
<=	کوچک‌تر مساوی	$۱۰ <= ۱۵$	درست	اگر عملوند اول کوچک‌تر یا مساوی عملوند دوم باشد، نتیجه درست است.
< >	نامساوی	$۱۰ < > ۱۵$	درست	اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است.
!=	نامساوی	$۱۰ != ۱۰$	نادرست	اگر عملوند اول مخالف عملوند دوم باشد، نتیجه درست است.
!<	کوچک‌تر نیست	$۱۰ !< ۱۲$	نادرست	اگر عملوند اول کوچک‌تر از عملوند دوم نباشد، نتیجه درست است.
!>	بزرگ‌تر نیست	$۱۰ !> ۱۲$	درست	اگر عملوند اول بزرگ‌تر از عملوند دوم نباشد، نتیجه درست است.

جدول ۴-۲ عملگرهای منطقی در SQL				
X	Y	X AND Y	X OR Y	Not X
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

جدول ۵-۲ عملگرهای یکانی در SQL				
عملگر	نام	مثال	نتیجه	هدف
+	مثبت	+۵	۵	مقدار مثبت عملوند را برمی‌گرداند.
-	منفی	-۱۵	-۱۵	مقدار منفی عملوند را برمی‌گرداند.
~	متمم یک	~۱۰۱	۰۱۰	متمم یک عملوند را برمی‌گرداند.

عملگرهای یکانی

بر روی یک عملوند صحیح عمل می‌کنند. جدول ۵-۲ عملکرد این عملگرها را نمایش می‌دهد.

عملگر اتصال رشته‌ای

عملگر است که بین دو عملوند از نوع رشته‌ای قرار گرفته آن‌ها را به هم الحاق می‌کند.

مثال ۷-۲. دستوراتی که رشته «Database with SQL» را در متغیر S۳ قرار می‌دهند.

```
SET @S۱ = "Database with "
SET @S۲ = "SQL"
SET @S۳ = @S۱ + @S۲
```

عملگرهای ویژه

عملگرهایی هستند که در SQL کاربردهای خاصی دارند. بعضی از این عملگرها در زیر آمده‌اند:

۶-۲. پرس‌وجوهای مرکب

این پرس‌وجوها از چندین پرس‌وجو تشکیل شده‌اند. عملگرهایی که برای ایجاد پرس‌وجوهای مرکب به کار می‌روند، عبارت‌اند از:

۵. عملگر UNION (همان عملگر $\cup$ جبر رابطه‌ای).

۶. عملگر INTERSECT (همان عملگر $\cap$ جبر رابطه‌ای).

۷. عملگر EXCEPT (همان عملگر $-$ جبر رابطه‌ای).

۱-۶-۲. عملگر UNION

این عملگر نتیجه دو یا چند دستور SQL را با هم ترکیب کرده، رکوردهای تکراری را فقط یک بار نمایش می‌دهد.

 مثال ۲۳-۲. پرس‌وجویی که تمام افراد موجود در بانک اطلاعات انتشارات را نمایش می‌دهد.

در این پرس‌وجو، ابتدا مؤلفین را بازایی می‌کنیم. سپس، رواسای انتشارات را بازایی کرده، آن‌ها را با هم ترکیب می‌کنیم. یعنی:

```
SELECT aFname, aLname FROM Authors
UNION
SELECT bFname, bLname FROM Publishers
```

	aFname	aLname
1	باقر	رحیم پور
2	رضا	امینی
3	رمضان	شهاب نژاد
4	سید حجت اله	جنیدی
5	محمود رضا	حاتمی
6	مرتضی	یویان
7	یوسف	شهاب نژاد

همان‌طور که در این خروجی می‌بینید، نام و نام خانوادگی مؤلفین و رواسای انتشارات نمایش داده شده است.

۲-۶-۲. عملگر UNION ALL

این عملگر همانند عملگر UNION عمل می‌کند. با این تفاوت که تکرار رکوردها را حذف نمی‌کند.

 مثال ۲۴-۲. پرس‌وجویی که کلیه افراد موجود در بانک اطلاعاتی انتشارات را نمایش می‌دهد (این پرس‌وجو، نام و نام خانوادگی تکراری را چند بار نمایش می‌دهد).

```
SELECT aFname, aLname FROM Authors
UNION ALL
SELECT bFname, bLname FROM Publishers
```

	aFname	aLname
1	رمضان	شهاب نژاد
2	یوسف	شهاب نژاد
3	باقر	رحیم پور
4	مرتضی	یویان
5	سید حجت اله	جنیدی
6	رمضان	شهاب نژاد
7	محمود رضا	حاتمی
8	رضا	امینی

همان‌طور که در این خروجی می‌بینید، برخی از نام و نام خانوادگی‌ها تکرار شده‌اند.

۳-۶-۲. عملگر INTERSECT

این عملگر از نتیجه چند پرس وجو اشتراک می گیرد (مانند عملگر $\cap$ جبر رابطه ای) و نتیجه اشتراک را برمی گرداند.

مثال ۲۵-۲. پرس وجویی که کد مؤلفینی که کتاب تألیف کرده اند را نمایش می دهد.

```
SELECT atID FROM Authors
INTERSECT
SELECT atID FROM AutBook
```

	atID
1	01
2	03
3	05

مثال ۲۶-۲. پرس وجویی که نام و نام خانوادگی افرادی که هم مؤلف و هم مدیر انتشارات هستند را نمایش می دهد.

```
SELECT aFname, aLname FROM Authors
INTERSECT
SELECT bFname, bLname FROM Publishers
```

	aFname	aLname
1	رضا	تباس نژاد

۴-۶-۲. عملگر EXCEPT

این عملگر تفاضل بین نتیجه پرس وجوی اول و دوم را برمی گرداند (مانند عملگر - در جبر رابطه ای). یعنی، رکوردهایی را برمی گرداند که در پرس وجوی اول باشد، ولی در نتیجه پرس وجوی دوم نباشد.

مثال ۲۷-۲. پرس وجویی که کد مؤلفینی که کتاب چاپ شده در این بانک ندارند، را نمایش می دهد.

```
SELECT atID FROM Authors
EXCEPT
SELECT atID FROM AutBook
```

	atID
1	02
2	04

مثال ۲۸-۲. پرس وجویی که نام و نام خانوادگی رواسای انتشارات را نمایش می دهد که مؤلف نیستند.

```
SELECT bFname, bLname FROM Publishers
EXCEPT
SELECT aFname, aLname FROM Authors
```

	bFname	bLname
1	رضا	امینی
2	محمد رضا	حاتمی

نکته: عملگرهای UNION، INTERSECT و EXCEPT، رکوردهای تکراری را یک بار نمایش می دهند. برای نمایش رکوردهای تکراری می توان از عملگرهای UNION ALL، INTERSECT ALL و EXCEPT ALL استفاده کرد.

نکته: همان طور که بیان گردید، عبارت ORDER BY برای مرتب سازی رکوردها به کار می رود. در هنگام به کارگیری ORDER BY در پرس وجوهای مرکب باید دقت کنید که این عبارت را فقط در یک پرس وجو تایپ کنید.

مثال ۲۹-۲. پرس وجویی که شماره مؤلفینی را نمایش می دهد که در جدول AutBook حق تألیف نگرفته اند (اطلاعات را بر اساس شماره مؤلف به صورت نزولی مرتب کرده، نمایش می دهد).

```
SELECT atID FROM Authors
EXCEPT
SELECT atID FROM AutBook ORDER BY atID DESC
```

	atID
1	04
2	02

جدول ۲-۷ توابع تجمعی (خلاصه سازی).

تابع	ورودی	هدف
------	-------	-----

AVG	عددی	میانگین مقادیر را برمی گرداند.
SUM	عددی	مجموع مقادیر را برمی گرداند.
MAX	عددی یا غیر عددی	بزرگ ترین مقدار را برمی گرداند.
MIN	عددی یا غیر عددی	کوچک ترین مقدار را برمی گرداند.
COUNT	عددی یا غیر عددی	تعداد رکوردها یا مقادیر غیر تهی را برمی گرداند.
STDEV	عددی	انحراف معیار یک فیلد یا چند مقدار را برمی گرداند.
STDEVP	عددی	انحراف معیار توزیعی یک فیلد یا چند مقدار را برمی گرداند.
VAR	عددی	واریانس مقادیر یک یا چند فیلد را برمی گرداند.
VARP	عددی	واریانس توزیعی مقادیر یک فیلد یا چند مقدار را برمی گرداند.

۷-۲. توابع تجمعی

توابع تجمعی برای خلاصه سازی اطلاعات به کار می روند (یعنی، مجموعه ای (کلکسیون) از مقادیر را به عنوان ورودی دریافت کرده، یک مقدار را برمی گرداند). با توابع تجمعی می توانید مجموع مقادیر، بزرگ ترین مقدار، کوچک ترین مقدار، میانگین مقادیر یک یا چند فیلد را برگردانید. نمونه ای از این توابع در جدول ۷-۲ آمده اند.

۱۰-۲. پرس وجوی فرعی

پرس وجویی که نتیجه آن توسط پرس وجوی دیگر مورد استفاده قرار می گیرد، پرس وجوی فرعی^۲ (متداخل^۳) نام دارد. در پرس وجوهای فرعی، ممکن است چندین پرس وجو به صورت پشت سرهم قرار گیرند. در این صورت، نتیجه پرس وجوی سطح بعدی در سطح قبلی استفاده می شود و این روند تا اولین سطح پرس وجو ادامه می یابد. پرس وجوهای فرعی را می توان در عبارت WHERE و HAVING مربوط به پرس وجوی اصلی استفاده کرد. در هنگام استفاده از پرس وجوهای فرعی باید نکات زیر را رعایت کنید:

از عملگر BETWEEN نمی توان در پرس وجوی فرعی استفاده کرد. ولی، در پرس وجوی اصلی می توان از این عملگر استفاده نمود.

پرس وجوهای فرعی باید در داخل پرانتز قرار گیرند.

پرس وجوهای فرعی که بیش از یک رکورد (تاپل) را برمی گرداند، می توانند با عملگرهایی مانند IN مورد استفاده قرار گیرند.

در یک پرس وجوی فرعی نمی توان از گزینه ORDER BY استفاده کرد. برای این منظور، می توان از گزینه GROUP BY استفاده نمود تا عمل ORDER BY نیز انجام شود.

پرس وجوی فرعی به صورت زیر به کار می رود:

دستور پرس وجوی اصلی

...

(دستور پرس وجوی فرعی) عملگر [لیست فیلدها] WHERE

برخی از مثال‌های پرس‌وجوی فرعی را در ادامه می‌بینید.

مثال ۳۶-۲. پرس‌وجویی که ناشرانی را نمایش می‌دهد که حداقل برای یک کتاب حق نشر دریافت کرده‌اند.

برای حل این مثال، ابتدا شماره ناشرانی که در جدول PubBook وجود دارند را مشخص می‌کنیم، یعنی:

```
SELECT pubID FROM PubBook
```

سپس، از جدول Publishers، ناشرانی را نمایش می‌دهیم که شماره آن‌ها در نتیجه پرس‌وجوی قبلی موجود باشد. یعنی:

```
SELECT * FROM Publishers
WHERE pubID IN (SELECT pubID FROM PubBook)
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	01	فناوری نوین	01122566687	fanavarienovin.net	بابل	رمضان	عباس نژاد	15
2	02	دانش نگار	02166400220		تهران	محمد رضا	حاتمی	100

مثال ۳۷-۲. پرس‌وجویی که ناشرانی را نمایش می‌دهد که هیچ حق نشری دریافت نکردند (برای انجام این پرس‌وجو از عملگر NOT IN استفاده می‌شود).

```
SELECT * FROM Publishers
WHERE pubID NOT IN (SELECT pubID FROM PubBook)
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	03	دانشگاه مازندران	01125226686		بابلسر	رضا	امینی	10

مثال ۳۸-۲. پرس‌وجویی که لیست گروه کتاب‌های را بازایی می‌کند که حداقل یک کتاب از آن گروه در جدول Books وجود دارد.

```
SELECT * FROM GroupBook g
WHERE EXISTS (SELECT * FROM Books WHERE groupID = g.groupID)
```

همان‌طور که در این پرس‌وجو می‌بینید، از عملگر EXISTS استفاده گردید.

این عملگر معنی شامل شدن را می‌دهد که به جای این دستور می‌توان از عملگر

IN به صورت زیر استفاده کرد:

```
SELECT * FROM GroupBook
WHERE groupID IN (SELECT groupID FROM Books)
```

مثال ۳۹-۲. پرس‌وجویی که لیست مؤلفانی را نمایش می‌دهد که تاکنون حق تألیف دریافت نکردند.

```
SELECT * FROM Authors a
WHERE NOT EXISTS (SELECT * FROM AutBook WHERE atID = a.atID )
```

	atID	aFname	aLname	Age	Ranking	Email	Mobile	sumPayment
1	02	یوسف	عباس نژاد	26	لیسانس	yousef_dssht@yahoo.com	NULL	NULL
2	04	مرتضی	پویان	27	NULL	NULL	NULL	NULL

همان طور که در این پرس وجو می بینید، عملگر NOT EXISTS مولفینی که هیچ حق تالیفی نگرفته اند را تعیین می کند. به جای این دستور می توانید از عملگر NOT IN به صورت زیر استفاده کنید:

```
SELECT * FROM Authors
WHERE atID NOT IN (SELECT atID FROM AutBook)
```

 مثال ۴۰-۲. پرس وجویی که گروه هایی که اصلاً کتاب ندارند را نمایش می دهد (برای حل این پرس وجو از عملگر EXCEPT استفاده کنید).

```
SELECT * FROM GroupBook
WHERE groupID IN
(SELECT groupID FROM GroupBook
EXCEPT
SELECT groupID FROM Books )
```

	groupID	groupName
1	05	نگرافیک رایانه ای

پرس وجوی فرعی کد گروه هایی را برمی گرداند که کتاب ندارند (با عملگر EXCEPT) و پرس وجوی اصلی اطلاعات این گروه ها را نمایش می دهد.

 مثال ۴۱-۲. پرس وجویی که کتاب های را نمایش می دهد که حق تألیف برای آن ها دریافت نگردید (با عملگر INTERSECT).

```
SELECT * FROM Books
WHERE ISBN IN
(SELECT ISBN FROM Books
INTERSECT
SELECT ISBN FROM AutBook)
```

	ISBN	Title	Page	Price	editNo	printNo	groupID
1	6009141302	حل مسائل C++	216	57000	1	2	01
2	6009141340	حل مسائل C#	256	62000	1	2	01
3	6009141388	امنیت شبکه	192	65000	1	1	03
4	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02
5	6009225422	اصول طراحی سیستم های شی نگرا	432	130000	1	1	01
6	6009225439	مدیریت استراتژیک	272	75000	1	1	04
7	7009225431	رهیافت C++	320	43000	1	1	01

Query executed successfully. | 1-VAIO\YOUSEF (10.0 RTM) | 1-VAIO\1 (54) | PublishDB | 00:00:00 | 7 rows

پرس وجوی فرعی، کد کتاب هایی را برمی گرداند که در جدول AutBook هستند. سپس، پرس وجوی اصلی، اطلاعات کتاب ها را نمایش می دهد.

 مثال ۴۲-۲. پرس وجویی که کتاب های با کد گروه کتاب '۰۲' را نمایش می دهد که تاکنون برای آن ها حق تألیف دریافت نگردید (با عملگر EXCEPT).

```
SELECT * FROM Books
WHERE ISBN IN
(SELECT ISBN FROM Books
WHERE groupID = '۰۲')
EXCEPT
SELECT ISBN FROM AutBook)
```

ISBN	Title	Page	Price	editNo	printNo	groupID
------	-------	------	-------	--------	---------	---------

پرس وجوی فرعی شایبک کتاب های با کد

گروه '۰۱' که هیچ حق تألیفی برای آن ها دریافت نگردید را برمی گرداند و پرس وجوی اصلی، کلیه اطلاعات این کتاب ها را نمایش می دهد.

۲-۱۱-۲. پیوند متعادل

پیوند متعادل، رایج ترین پیوند است که با استفاده از فیلد مشترک بین دو جدول انجام می شود. این پیوند همان ضرب دکارتی بین دو جدول است، به طوری که شرط برابری فیلد مشترک آن ها باید در ضرب دکارتی ذکر گردد. الحاق بین دو جدول به صورت زیر انجام می شود:

```
SELECT * FROM t۱, t۲
WHERE t۱.relationField = t۲.relationField
```

مثال ۴۶-۲. پرس و جویی که اطلاعات نام کتاب و نام گروه را نمایش می دهد.

```
SELECT Title, groupName
FROM GroupBook AS g ,Books AS b
WHERE b.groupID = g.groupID
```

	Title	groupName
1	حل مسائل C++	برنامه نویسی
2	حل مسائل C#	برنامه نویسی
3	امنیت شبکه	شبکه رابانه ای
4	اصول طراحی پایگاه داده	پایگاه داده
5	اصول طراحی سیستم های شی گرا	برنامه نویسی
6	مدیریت استراتژیک	فناوری اطلاعات
7	رهیافت C++	برنامه نویسی

همان طور که در این پرس و جو می بینید، شرط WHERE، عبارت b.groupID = g.groupID است. (فیلد groupID، فیلد مشترک بین جداول Books و GroupBook می باشد).

مثال ۴۷-۲. پرس و جویی که نام ناشر، نام خانوادگی مدیر انتشارات، نام کتاب و میزان حق نشر دریافت شده برای هر کتاب را نمایش می دهد.

```
SELECT pName, bFname, bLname, Title, Payment
FROM Publishers p ,Books AS b, PubBook AS pu
WHERE b.ISBN = pu.ISBN AND p.pubID = pu.pubID
```

	pName	bFname	bLname	Title	Payment
1	فناوری نوین	رمضان	عباس نژاد	حل مسائل C++	6000000
2	فناوری نوین	رمضان	عباس نژاد	حل مسائل C#	7000000
3	فناوری نوین	رمضان	عباس نژاد	امنیت شبکه	6000000
4	فناوری نوین	رمضان	عباس نژاد	اصول طراحی پایگاه داده	6000000
5	فناوری نوین	رمضان	عباس نژاد	اصول طراحی سیستم های شی گرا	8000000
6	فناوری نوین	رمضان	عباس نژاد	مدیریت استراتژیک	5000000
7	فناوری نوین	رمضان	عباس نژاد	رهیافت C++	3500000
8	دانش نگار	محمد رضا	حاتمی	رهیافت C++	3500000

همان طور که در این پرس و جو می بینید، برای برقراری ارتباط بین سه جدول، ارتباط بین دو به دو با هم برقرار گردید. یعنی، ارتباط جداول Publishers و PubBook از طریق فیلد pubID برقرار گردید و پیوند جدول PubBook و Books، از طریق فیلد ISBN برقرار شد.

۲-۱۱-۳. پیوند نامتعادل

در پیوند نامتعادل دیدید که برای برقراری ارتباط از عملگر = در شرط WHERE استفاده شده است. اگر به جای عملگر =، از عملگرهای دیگر (نظیر >، <، <=، >=، < و >) در شرط WHERE استفاده کنید، پیوند نامتعادل ایجاد خواهد شد.

مثال ۴۸-۲. پرس و جویی که دو جدول Books و GroupBook را از طریق عملگر > پیوند می زند.

```
SELECT * FROM Books AS b, GroupBook
WHERE b.groupID > GroupBook.groupID
```

	ISBN	Title	Page	Price	editNo	printNo	groupID	groupID	groupName
1	6009141388	امنیت شبکه	192	65000	1	1	03	01	برنامه نویسی
2	6009141388	امنیت شبکه	192	65000	1	1	03	02	پایگاه داده
3	6009141395	اصول طراحی پایگاه داده	256	73000	1	1	02	01	برنامه نویسی
4	6009225439	مدیریت استراتژیک	272	75000	1	1	04	01	برنامه نویسی
5	6009225439	مدیریت استراتژیک	272	75000	1	1	04	02	پایگاه داده
6	6009225439	مدیریت استراتژیک	272	75000	1	1	04	03	شبکه رایانه ای

۴-۱۱-۲. پیوند درونی

الحاق‌هایی که تاکنون ایجاد شده‌اند، شرط الحاق در بخش WHERE دستور SELECT آمده است. اگر در هنگام ایجاد پیوند، شرط معمولی نیز وجود داشته باشد، شرط پیوند با شرط معمولی، "و" منطقی (AND) می‌گردد. در این حالت، اگر تعداد شرط‌ها زیاد شود، تشخیص شرط الحاق (پیوند) و شرط معمولی مشکل خواهد بود. برای رفع این مشکل، پیوند درونی مطرح گردید. پیوند درونی به صورت زیر به کار می‌رود:

```
SELECT لیست فیلدها FROM t۱
      شرط پیوند ۱ ON t۲ INNER JOIN
      شرط پیوند ۲ ON t۳ INNER JOIN
      ...
      WHERE شرط معمولی
```

مثال ۴۹-۲. پرس‌وجویی که نام، نام خانوادگی مؤلف، نام کتاب و میزان حق تألیف دریافتی برای هر کتاب را نمایش می‌دهد.

```
SELECT aFname, aLname, Title, Payment
FROM Authors AS a
INNER JOIN AutBook au ON a.atID = au.atID
INNER JOIN Books b ON au.ISBN = b.ISBN
```

	aFname	aLname	Title	Payment
1	رمضان	عباس نژاد	حل مسائل C++	6000000
2	رمضان	عباس نژاد	حل مسائل C#	6000000
3	رمضان	عباس نژاد	امنیت شبکه	6000000
4	رمضان	عباس نژاد	اصول طراحی پایگاه داده	5000000
5	باقر	رحیم پور	اصول طراحی پایگاه داده	5000000
6	رمضان	عباس نژاد	اصول طراحی سیستم‌های شی گرا	5000000
7	باقر	رحیم پور	اصول طراحی سیستم‌های شی گرا	5000000
8	نسید حجت اله	جلیلی	مدیریت استراتژیک	6000000
9	رمضان	عباس نژاد	رهیافت C++	6000000

مثال ۵۰-۲. پرس‌وجویی که نام ناشر، نام کتاب و مبلغ حق نشر کتابی که حق نشر آن زیر ۴۰۰۰۰۰۰ ریال باشد را نمایش می‌دهد.

```
SELECT pName, Title, Payment
FROM Publishers AS p
INNER JOIN PubBook pu
ON p.pubID = pu.pubID
INNER JOIN Books b
ON pu.ISBN = b.ISBN
WHERE Payment < ۴۰۰۰۰۰۰
```

	pName	Title	Payment
1	فناوری نوین	رهیافت C++	3500000
2	دانش نگار	رهیافت C++	3500000

مثال ۵۱-۲. پرس‌وجویی که لیست ناشرانی را بازایی می‌کند که برای هیچ کتابی حق نشر بیش از ۴۰۰۰۰۰۰ ریال دریافت نکردند.

```
SELECT p.*
FROM Publishers AS p
INNER JOIN PubBook pu ON p.pubID = pu.pubID
WHERE p.pubID NOT IN (SELECT pubID FROM PubBook WHERE
Payment > ۴۰۰۰۰۰۰)
```

	pubID	pName	Tel	URL	cityName	bFname	bLname	countBookPrint
1	02	دانش نگار	02166400220		تهران	محمد رضا	حاتمی	100

مثال ۵۲-۲. پرس وجویی که لیست مؤلفانی که برای هیچ کتابی حق تألیف بیش از ۵۰۰۰۰۰۰ ریال دریافت نکردند، را نمایش می دهد.

```
SELECT DISTINCT a.*
FROM Authors AS a
INNER JOIN AutBook at ON a.atID = at.atID
WHERE a.atID NOT IN (SELECT atID FROM AutBook WHERE
Payment > ۵۰۰۰۰۰۰ )
```

	atID	aFname	aLname	Age	Ranking	Email	Mobile	sumPayment
1	03	باقر	رحیم پور	32	خوبی لیسانس		NULL	10000000

۱۴ - ۲. مسائل حل شده

مثال ۱. دستوراتی که رکوردهایی را به برخی از جداول بانک اطلاعاتی Buy_Sell اضافه می کنند تا خروجی دستورات یک شکل باشند.

```
/****** اضافه کردن کاربر *****/
INSERT INTO Login\ VALUES ('۰۱', 'Admin', '۱۳۵۷۹۲۴۶۸')
INSERT INTO Login\ VALUES ('۰۲', 'test', '۸۷۶۵۴۳۲۱۰۹')
/****** اضافه کردن شهر *****/
INSERT INTO City VALUES ('۰۲۱', '۰۱', 'تهران')
INSERT INTO City VALUES ('۰۱۱۱', '۰۲', 'بابل')
INSERT INTO City VALUES ('۰۱۲۱', '۰۲', 'آمل')
INSERT INTO City VALUES ('۰۱۵۱', '۰۲', 'ساری')
INSERT INTO City VALUES ('۰۲۱۱', '۰۱', 'اصفهان')
INSERT INTO City VALUES ('۰۵۱۱', '۰۱', 'مشهد')
/****** اضافه کردن استان *****/
INSERT INTO State VALUES ('۰۱', '۰۱', 'تهران')
INSERT INTO State VALUES ('۰۲', '۰۱', 'اصفهان')
INSERT INTO State VALUES ('۰۳', '۰۱', 'مشهد')
/****** اضافه کردن فروشنده *****/
INSERT INTO Seller VALUES ('۰۱', '۰۱', 'احمدی')
INSERT INTO Seller VALUES ('۰۲', '۰۱', 'رضائی')
INSERT INTO Seller VALUES ('۰۳', '۰۲', 'محمدی')
/****** اضافه کردن انبار *****/
INSERT INTO Store VALUES ('۰۱', '۰۱', 'انبار مرکزی')
INSERT INTO Store VALUES ('۰۲', '۰۱', 'انبار شماره ۱')
/****** اضافه کردن گروه کالا *****/
INSERT INTO merchandiseGroup VALUES ('۰۱', '۰۲', 'کیلو',
'حبوبات')
INSERT INTO merchandiseGroup VALUES ('۰۲', '۰۱', 'نوشیدنی',
'باتری')
INSERT INTO merchandiseGroup VALUES ('۰۳', '۰۲', 'متر', 'کابل')
/****** اضافه کردن مشتری *****/
INSERT INTO Customer VALUES ('۰۱', 'رضا', 'احمدی',
'۲۱۵۲۰۶۴۴۴۳', '۰۲۱۶۴۰۰۲۲۰', '۰۹۱۲۲۲۰۹۹۹', 'تهران خیابان',
'ولیعصر', NULL, '۰۲۱', '۰۱', NULL, ۱, ۱۰۰۰۰۰۰, '۰۱')
INSERT INTO Customer VALUES ('۰۲', 'علی', 'رضوی',
'۰۵۱۱۶۴۰۰۲۲۰', '۰۹۱۵۲۲۰۹۹۹', 'مشهد خیابان امام رضا', NULL,
'۰۵۱۱', '۰۳', NULL, ۱, ۱۰۰۰۰۰۰, '۰۱')
/****** اضافه کردن کالا *****/
INSERT INTO Merchandise VALUES ('۰۱', '۰۲', 'نوشابه بزرگ',
'۶۰۰۰', '۴۰۰۰', '۵۰۰۰', '۵۵۰۰', '۱۰۰۰', '۰۱')
```

```

INSERT INTO Merchandise VALUES ('۰۲','۰۲', 'دوغ بزرگ', ۶۰۰۰,
۴۰۰۰, ۵۰۰۰, ۵۷۰۰, ۵۰۰, '۰۱')
INSERT INTO Merchandise VALUES ('۰۳','۰۲', 'برنج', ۳۲۰۰۰,
۲۶۰۰۰, ۲۵۰۰۰, ۲۶۰۰۰, ۱۵۰۰۰, '۰۲')
/***** اضافه کردن فاکتور خرید *****/
INSERT INTO BuyInvoice VALUES (۹۷۵۰۰,'۹۱/۰۳/۳۱','۰۱','۰۲', 'نقد')
INSERT INTO BuyInvoice VALUES (۹۷۵۰۱,'۹۱/۰۴/۰۱','۰۲','۰۲', 'چک')
INSERT INTO BuyInvoice VALUES (۹۷۵۰۲,'۹۱/۰۴/۱۵','۰۱','۰۲', 'نقدی')
/***** اضافه کردن ریز فاکتور خرید *****/
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۰,'۰۱',۴۰۰۰,۵۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۰,'۰۲',۵۰۰۰,۳۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۰,'۰۳',۱۵۰۰۰,۳۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۱,'۰۱',۴۰۰۰,۲۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۱,'۰۲',۵۰۰۰,۱۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۱,'۰۳',۱۵۰۰۰,۳۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۲,'۰۱',۴۰۰۰,۲۰۰,۰۲')
INSERT INTO BuyInvoice\ VALUES (۹۷۵۰۲,'۰۳',۱۵۰۰۰,۰,۱۰۰,۰۲')

```

مثال ۲. پرس وجویی که استان 'مازندران' با کد استان '۰۴' را کاربری با کد '۰۲' به جدول State اضافه می کند.

```
INSERT INTO State VALUES ('۰۴', '۰۲', 'مازندران')
```

مثال ۳. پرس وجویی که اطلاعات جدول State را به جدول State اضافه می کند.

```
SELECT * INTO State\ FROM State
```

	stateID	stateName	userCode
1	01	تهران	01
2	02	اصفهان	01
3	03	مشهد	01
4	04	مازندران	02

مثال ۴. پرس وجویی که اطلاعات جدول State را نمایش می دهد.

```
SELECT * FROM State\
```

مثال ۵. پرس وجویی که مقدار فیلدهای کد استان (stateID) و نام استان (stateName) جدول State را به جدول State اضافه می کند.

```
INSERT INTO State\ ( stateID, stateName)
SELECT stateID, stateName FROM State
```

	stateID	stateName	userCode
1	01	تهران	NULL
2	02	اصفهان	NULL
3	03	مشهد	NULL
4	04	مازندران	NULL
5	01	تهران	01
6	02	اصفهان	01
7	03	مشهد	01
8	04	مازندران	02

مثال ۶. پرس وجویی که اطلاعات جدول State را نمایش می دهد.

```
SELECT * FROM State\
```

همان طور که در این خروجی می بینید، برخی از رکوردها مقدار فیلد

userCode آن ها NULL است. چون، فقط مقدار فیلدهایی کد استان و نام استان از جدول State به جدول State اضافه شدند.

مثال ۷. پرس وجویی که رکوردهایی از جدول State را حذف می کند که مقدار فیلد userCode آن ها NULL نیست. سپس رکوردهای باقی مانده را نمایش می دهد.

```
DELETE FROM State\ WHERE userCode IS NOT NULL
SELECT * FROM State\
```

	stateID	stateName	userCode
1	01	تهران	NULL
2	02	اصفهان	NULL
3	03	مشهد	NULL
4	04	مازندران	NULL

همان طور که در خروجی مشاهده می گردد، رکوردهایی که مقدار userCode آن ها NULL نبود، حذف شدند.

مثال ۸. پرس وجویی که رکوردهایی از جدول State1 را حذف می کند، که مقدار فیلد userCode آن ها NULL باشد و سپس اطلاعات جدول State1 را نمایش می دهد.

```
DELETE FROM State\ WHERE userCode IS NULL
SELECT * FROM State\
```

مثال ۳۵. پرس وجویی که کاربرانی که هم اطلاعات استان ها و هم اطلاعات مشتریان را وارد کرده اند یا تغییر داده اند، نمایش می دهد.

```
SELECT userName FROM Login\
WHERE userCode IN( SELECT userCode FROM Customer
INTERSECT
SELECT userCode FROM City )
```

userName
1 Admin

مثال ۳۶. پرس وجویی که نام و نام خانوادگی مشتریانی که از آن ها خرید نداشته ایم را بازایی می کند.

```
SELECT firstName, lastName FROM Customer
WHERE ID NOT IN(SELECT ID FROM BuyInvoice\)
```

firstName	lastName
-----------	----------

مثال ۳۷. پرس وجویی که گروه های کالایی را بازایی می کند که حداقل یک کالا از آن گروه خریداری شده است.

```
SELECT * FROM merchandiseGroup
WHERE merchandiseGroupID IN(SELECT merchandiseGroupID FROM
Merchandise
WHERE merchandiseID IN ( SELECT merchandiseID FROM
BuyInvoice\ ) )
```

این پرس وجوی از سه پرس وجوی تودرتو تشکیل شده است. داخلی ترین پرس وجو، کالاهایی که تاکنون از آن ها خرید انجام شده را بازایی می کند:

```
( SELECT merchandiseID FROM BuyInvoice\ )
```

پرس وجو داخلی دوم، کد گروه کالاهایی را بازایی می کند که از آن ها خرید انجام شده است:

```
(SELECT merchandiseGroupID FROM Merchandise
WHERE merchandiseID IN ( SELECT merchandiseID FROM
BuyInvoice\ ) )
```

و در نهایت پرس وجو کلی، گروه های کالا را برمی گرداند که از آن گروه کالا خرید انجام شده است.

مثال ۳۸. پرس وجویی که گروه کالایی که از آن‌ها خرید نشده است را بازیابی می‌کند.

```
SELECT * FROM merchandiseGroup
WHERE merchandiseGroupID NOT IN(SELECT merchandiseGroupID
FROM Merchandise
WHERE merchandiseID IN ( SELECT merchandiseID FROM
BuyInvoice\ ) )
```

	merchandiseGroupID	merchandiseGroupName	unit	userCode
1	03	کابل	پتر	02

این پرس وجو مانند پرس وجوی قبلی است. با این تفاوت که در بخش WHERE پرس وجوی خارجی به جای IN، عملگر NOT آمده است.

مثال ۳۹. پرس وجویی که شماره فاکتور، نام مشتری، نام خانوادگی مشتری، نام کالاهای خریداری شده از آن مشتری را بازیابی می‌کند (مرتب شده براساس نام خانوادگی و نام مشتری).

```
SELECT b۲.invoiceNumeral, firstName, lastName, merchandiseName
FROM BuyInvoice b۱, BuyInvoice\ b۲, Customer c, Merchandise m
WHERE b۱.invoiceNumeral = b۲.invoiceNumeral
AND b۲.merchandiseID = m.merchandiseID
AND c.ID = b۱.ID
ORDER BY lastName, firstName
```

	invoiceNumeral	firstName	lastName	merchandiseName
1	97500	رضا	احمدی	نوشابه بزرگ
2	97500	رضا	احمدی	دوغ بزرگ
3	97500	رضا	احمدی	برنج
4	97502	رضا	احمدی	نوشابه بزرگ
5	97502	رضا	احمدی	برنج
6	97501	علی	رضوی	نوشابه بزرگ
7	97501	علی	رضوی	دوغ بزرگ
8	97501	علی	رضوی	برنج

۱۵-۲. دستور کار آزمایشگاه

- یکسری اطلاعات به جدول رانندگان (Drivers)، ماشین (Cars)، مشتریان (Customers) و سفارشات (Orders) اضافه کرده، سپس به پرس وجوهای زیر پاسخ دهید:
- پرس وجویی بنویسید که تمام ماشین‌هایی را بازیابی کند که واحد آن‌ها تن است.
- پرس وجوی بنویسید تا نام و نام خانوادگی رانندگانی را بازیابی کند که در گواهی‌نامه آن‌ها کلمه 'پایه ۱' آمده است.
- پرس وجویی بنویسید تا اطلاعات رانندگانی را بازیابی کند که سفارشات شهر شیراز یا تهران را انجام داده‌اند.
- پرس وجویی بنویسید تا اطلاعات ماشین‌هایی را بازیابی کند که سفارشات شهر آمل یا بابل را انجام داده‌اند (مقصد مشتریان آن‌ها آمل یا بابل بوده است).

۶. پرس وجویی بنویسید تا اطلاعات ماشین هایی را بازیابی کند که سفارشات شهر آمل یا بابل را انجام نداده- اند (مقصد مشتریان آن ها آمل یا بابل نبود).

۷. پرس وجویی بنویسید که اطلاعات مشتریانی را بازیابی کند که اصلاً سفارش نداشتند.

۸. پرس وجویی بنویسید تا اطلاعات رانندگانی که فقط یک سفارش را انجام داده اند، بازیابی کند.

۹. پرس وجویی بنویسید تا اطلاعات رانندگانی که حداقل دو سفارش را انجام داده اند، بازیابی کند.

۱۰. پرس وجویی بنویسید تا ضرب دکارتی جداول ماشین ها (Cars) و رانندگان (Drivers) را انجام داده، بازیابی کند.

۱۱. پرس وجویی بنویسید تا نام، نام خانوادگی رانندگان و نام ماشین های مربوط به آن ها را بازیابی کند.

۱۲. پرس وجویی بنویسید تا نام، نام خانوادگی رانندگان و نام ماشین های آن ها را بازیابی کند، به طوری که برای جداول Cars و Drivers پیوند بیرونی چپ با عملگر > انجام دهد به شرطی که واحد ماشین نفر باشد.

۱۳. پرس وجویی بنویسید تا نام، نام خانوادگی مشتریان و شماره سفارش های را بازیابی کند به طوری که بین جداول Customers و Orders پیوند بیرونی کامل با عملگر > انجام دهد و اطلاعات را بر اساس نام خانوادگی مشتریان مرتب نماید. چنانچه نام خانوادگی مشتریان یکی باشد، اطلاعات را بر اساس شماره فاکتور مرتب کند.

۱۴. پرس وجویی بنویسید تا نام و نام خانوادگی رانندگان و مجموع کرایه هایی که دریافت کرده اند را بازیابی کند.

۱۵. پرس وجویی بنویسید تا اطلاعات ماشینی را بازیابی کند که نام خانوادگی راننده آن "احمدی" است.

۱۶. پرس وجویی بنویسید تا اطلاعات رانندگانی را بازیابی کند که واحد ماشین آن ها "تن" است.

۱۱۷- ۲. تمرین ها

۱. اطلاعات جداول ۹-۲، ۱۰-۲، ۱۱-۲ و ۱۲-۲ را در جدول مربوطه به بانک حسابداری وارد کنید.

۲. پرس وجویی بنویسید تا نام کل، نام معین و نام تفصیلی را نمایش دهد.

۳. پرس وجویی بنویسید تا نام گروه حساب و نام کل حساب ها را نمایش دهد.

۴. چند رکورد به جداول سربرگ سند و ریز سند اضافه کنید.

۵. پرس وجویی بنویسید تا اسنادی را نمایش دهد که در آن اسناد یکی از اقلام بدهکار یا بستانکار بالای ۲۰۰۰۰۰۰۰ ریال باشد.

۶. پرس وجویی بنویسید تا اسنادی را بازیابی کند که در نام تفصیلی آن ها، علی الحساب وجود داشته باشد.

۷. پرس وجویی بنویسید تا کلیه حساب های کل، معین، تفصیلی را بازیابی کند.

۸. پرس وجویی بنویسید تا که حساب های معین گروه حساب دارایی ها را بازیابی کند.

۹. پرس وجویی بنویسید تا سند خاصی را بازیابی کند.

۱۰. پرس وجویی بنویسید تا اسنادی که مجموع بدهکار و بستانکار آن‌ها یکی نیست را بازیابی کند (اسنادی که بالانس نیستند).

جدول ۹ - ۲ اطلاعات جدول گروه حساب.	
کد گروه حساب	نام گروه حساب
۱	دارایی‌ها
۲	بدهی‌ها
۳	سرمایه
۴	درآمدها
۵	هزینه‌ها

جدول ۱۰ - ۲ اطلاعات جدول تفضیلی.			
نام تفضیلی	کد تفضیلی	کد معین	کد کل
صندوق	۰۰۰۰۱	۰۰۰۰۱	۱۰۰
بانک ملی	۱۵۰۰۰	۰۰۰۰۲	۱۰۰
بانک صادرات	۱۶۰۰۰	۰۰۰۰۲	۱۰۰
بانک ملت	۱۷۰۰۰	۰۰۰۰۲	۱۰۰
...	...	...	...
بانک ملی	۱۵۰۰۰	۰۰۰۰۳	۱۰۰
بانک اقتصاد نوین	۳۲۰۰۰	۰۰۰۰۳	۱۰۰
بانک کشاورزی	۲۸۰۰۰	۰۰۰۰۳	۱۰۰
...	...	...	..
نام اداره دریافت کننده	۰۰۰۰۱	۰۰۰۰۴	۱۰۰
کارکرد تلفن ثابت	۱۰۰۰۱	۱۱۰۰۱	۱۲۰
سایر خدمات	۱۰۱۰۰	۱۱۰۰۱	۱۲۰
...	...	...	...
مشترکین تلفن ثابت	۱۰۱۰۱	۱۱۰۰۲	۱۲۰
مشترکین سلب امتیازی	۱۰۱۰۴	۱۱۰۰۲	۱۲۰
...	...	...	...
شرح کالا، نام انبار	۰۰۰۰۱	۱۲۰۰۰	۱۴۰
شرح کالا، نام انبار	۰۰۰۰۱	۲۲۰۰۰	۱۴۰
شرح کالا، نام انبار	۰۰۰۰۱	۲۲۰۰۰	۱۴۰
علی الحساب یا پرداختی	۱۰۰۰۱	۱۰۰۰۰	۱۵۰
پیش پرداخت مالیات	۱۰۰۰۱	۱۰۰۰۰	۱۵۰
...	...	...	...
علی الحساب خرید زمین	۲۲۲۰۰	۲۲۰۰۰	۱۵۰
علی الحساب خرید ساختمان	۲۲۲۱۰	۲۲۰۰۰	۱۵۰
...	...	...	...
سرمایه ساختمان	۱۰۰۱۰	۱۰۰۰۰	۱۷۰
درآمد نگهداری	۲۰۰۱۷	۱۰۰۰۰	۱۷۰
...	...	...	...
وام	۲۰۰۰۲	۲۰۰۰۰	۱۷۰
هزینه امکان رفاهی	۲۰۰۰۳	۲۰۰۰۰	۱۷۰
هزینه بیمارستان	۲۰۰۰۴	۲۰۰۰۰	۱۷۰
مسکونی و اداری	۱۱۰۰۰	۰۰۰۰۲	۲۰۰

۲۰۰	۰۰۰۰۲	۱۲۰۰۰	اداری و سایر
...	...	...	...
۲۱۰	۰۰۰۰۲	۱۱۰۰۰	مسکونی و انبار
۲۱۰	۰۰۰۰۲	۱۲۰۰۰	اداری و سایر
۲۴۰	۰۰۰۰۳	۱۱۰۰۰	مسکونی و انبار
۲۴۰	۰۰۰۰۳	۱۲۰۰۰	اداری و سایر
۲۵۰	۰۰۰۰۱	۰۰۰۰۱	مراکز سوئیچ

جدول ۱۱ - ۲ اطلاعات جدول کل.		
نام کل	کد کل	کد گروه حساب
موجودی نقدی	۱۰۰	۱
حساب‌ها و اسناد دریافتی تجاری	۱۲۰	۱
طلب/بدهی شرکت‌های وابسته	۱۷۰	۲
زمین	۲۰۰	۱
ساختمان	۲۱۰	۱
حرارت مرکزی	۲۴۰	۲

جدول ۱۲ - ۲ اطلاعات جدول معین.		
نام معین	کد معین	کد کل
صندوق	۰۰۰۰۱	۱۰۰
حساب‌های بانکی قابل برداشت	۰۰۰۰۲	۱۰۰
حساب‌های بانکی غیر قابل پرداخت	۰۰۰۰۳	۱۰۰
تنخواه	۰۰۰۰۴	۱۰۰
اشخاص وابسته تلفن ثابت	۱۱۰۰۱	۱۲۰
سایر اشخاص تلفن ثابت	۱۱۰۰۲	۱۲۰
سایر اشخاص تلفن همراه	۱۲۰۰۲	۱۲۰
موجودی کالای مصرفی در انبار	۱۲۰۰۰	۱۴۰
موجودی کالای سرمایه‌ای در انبار	۲۲۰۰۰	۱۴۰
پیش‌پرداخت‌های جاری	۱۰۰۰۰	۱۵۰
پیش‌پرداخت‌های سرمایه	۲۲۰۰۰	۱۵۰
تجاری	۱۰۰۰۰	۱۷۰
غیر تجاری	۲۰۰۰۰	۱۷۰
زمین	۰۰۰۰۱	۲۰۰
ساختمان	۰۰۰۰۲	۲۱۰
حرارت مرکزی	۰۰۰۰۳	۲۴۰
مرکز سوئیچ	۰۰۰۰۱	۲۵۰
کابل	۰۰۰۰۳	۲۵۰
...	...	...

۱۱. پرس‌وجویی بنویسید تا بیشترین مبلغ بدهکار، بیشترین مبلغ بستانکار را در جدول ریز اسناد بازایی کند.
۱۲. پرس‌وجویی بنویسید تا سندی را بازایی کند که تعداد ردیف‌های ریز آن‌ها از همه اسناد بیشتر است.
۱۳. پرس‌وجویی بنویسید تا مجموع بستانکار و مجموع بدهکار اسناد را به تفکیک شماره سند بازایی نماید.
۱۴. پرس‌وجویی بنویسید تا کد معین‌هایی را بازایی کند که هنوز در ردیف‌های اسناد استفاده نشده‌اند (با عملگر NOT IN).
۱۵. پرس‌وجویی بنویسید تا نام معین و تفصیلی‌هایی که در ردیف اسناد استفاده شده‌اند را بازایی کند.

۱۶. پرس وجویی بنویسید تا نام کل هایی که در ردیف اسناد استفاده نشده اند را بازیابی کند (با عملگر NOT EXISTS).

۱۷. پرس وجویی بنویسید تا نام معین هایی که هنوز در ردیف های اسناد استفاده نشده اند، را بازیابی کند (با عملگر EXCEPT).

۱۸. پرس وجویی بنویسید تا در دفتر معین اطلاعات، از یک تاریخ تا تاریخ دیگر را بازیابی کند (دفتر معین، تمام کد معین که در این بازه دارای مبلغ بدهکار یا بستانکار هستند را بازیابی می کند. اطلاعاتی که در دفتر معین ظاهر می شوند، عبارت اند از: ردیف، شماره سند، تاریخ، کد کل، نام کل، کد معین، نام معین، مبلغ بدهکاری و مبلغ بستانکار).

۱۹. پرس وجویی بنویسید تا شماره اسنادی که در شرح ردیف اسناد آن ها عبارت فناوری نوین وجود دارد را بازیابی کند (دقت کنید که شماره اسناد تکراری را فقط یک بار نمایش دهد).

۲۰. پرس وجویی بنویسید تا شماره اسنادی را بازیابی کند که مبلغ بدهکار یا بستانکار آن ها بین ۱۰۰۰۰۰۰۰ تا ۵۰۰۰۰۰۰۰ ریال باشد (شماره اسناد تکراری فقط یک بار نمایش داده شود).

۲۱. پرس وجویی بنویسید تا شماره و نام کل هایی را بازیابی کند که مجموع مبلغ بدهکار یا بستانکار آن ها در جدول ریز اسناد دقیقاً برابر ۱۰۰۰۰۰۰۰۰ ریال باشد.

۲۲. پرس وجویی بنویسید تا پیوند بیرونی چپ بین جدول گروه حساب و جدول کل را با عملگر < نمایش دهد.

۲۳. پرس وجویی بنویسید تا پیوند بیرونی راست بین جداول گروه حساب و کل را با عملگر > نمایش دهد.

۲۴. پرس وجویی بنویسید تا پیوند بیرونی کامل بین جداول گروه حساب و کل را با عملگر < نمایش دهد.

۲۵. پرس وجویی بنویسید تا اطلاعات تفصیلی هایی را بازیابی کند که نام تفصیلی آن ها با عبارت "مشترکین" شروع شده باشد.

۲۶. پرس وجویی بنویسید که اطلاعات تفصیلی هایی را بازیابی کند که نام تفصیلی آن ها با عبارت "ثابت" خاتمه یابد.

۲۷. پرس وجویی بنویسید تا شماره اسنادی را بازیابی کند که بالانس هستند (مجموع مبلغ بدهکار و بستانکار آن ها یکی است).

۲۸. پرس وجویی بنویسید تا شماره کل هایی را بازیابی کند که مجموع بدهکار آن ها بیشتر از مجموع بستانکار است (اختلاف بین بدهکار و بستانکار را نیز نمایش دهد).

۲۹. پرس وجویی بنویسید تا تمام تفصیلی که از یک شماره سند خاص تا شماره سند دیگر گردش دارند را بازیابی کند.

دیدها، رویه‌های ذخیره شده و توابع

ایجاد دید، دستورات SQL، رویه‌های ذخیره شده و توابع از بخش‌های بسیار مهم SQL هستند. دستورات و پرس‌وجوها را می‌توان به شکل دید^۱ در بانک اطلاعاتی ذخیره نمود. اما برنامه را می‌توان به صورت رویه‌های ذخیره شده^۲ یا توابع^۳ ذخیره کرد. در این فصل به شرح مفاهیمی از قبیل، دید، دستورات SQL، رویه‌های ذخیره شده و توابع پرداخته، ایجاد، ویرایش و حذف این اشیا را می‌آموزیم.

۱-۳. دید

همان‌طور که در فصل دوم دیدید، دستور SELECT، برای بازیابی اطلاعات از بانک اطلاعات به کار می‌رود. یکی از معایب این دستور این بود که در بانک اطلاعات به عنوان یک شیء ذخیره نمی‌گردید. یعنی، اگر بخواهید یک دستور SELECT را چندین بار اجرا نمایید باید آن را چندین بار تایپ کرده، سپس اجرا کنید. این امر موجب صرف وقت زیادی خواهد شد. از طرف دیگر، در SQL Server به طور مستقیم نمی‌توان تعیین کرد هر کاربر به چه فیلدها یا رکوردهایی از یک یا چند جدول دسترسی داشته باشد. برای رفع این مشکلات، می‌توانید دید را ایجاد نموده، از آن استفاده نمایید. چون دید به عنوان یک شیء در بانک اطلاعات ذخیره می‌گردد. برخی از دلایل استفاده از دید به طور خلاصه در زیر آمده است:

۱. امکان ذخیره سازی پرس‌وجو بر روی بانک اطلاعاتی با استفاده از دید.
۲. فراهم نمودن دیدهای مختلف برای کاربران از داده‌های فیلدهای بانک اطلاعاتی (پیاده‌سازی دید خارجی)
۳. ایجاد محدودیت دسترسی به داده از طریق دسترسی به دید به جای دسترسی به جدول.

۱-۱-۳. ایجاد دید

دید نه تنها در بانک اطلاعات ذخیره می‌شود، بلکه امکانات امنیتی را برای کاربران فراهم می‌نماید تا کاربران بتوانند به فیلدها و رکوردهای خاص خودشان دسترسی داشته باشند. دید، جدول مجازی^۴ با قالب‌های جدید ایجاد می‌کند. در جدول فاکتور، فاکتورهای خرید ثبت شده کاربران مختلف نگهداری می‌شود. فرض کنید کاربر خاصی بخواهد فاکتورهای ثبت شده مربوط به خودش را ببیند. در حالت عادی کل فاکتورها را مشاهده می‌کند. بنابراین، کاربر خاص باید با دستور SELECT که دارای شرط خاصی است، اطلاعات و

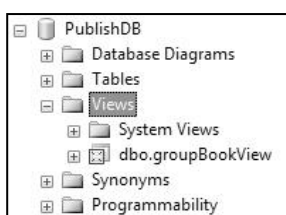
^۱.View^۲.Stored Procedures^۳.Functions^۴.Virtual table

رکوردهای فاکتورهای خودش را بازیابی نماید. اگر تعداد دفعات اجرای این دستور زیاد باشد (به دفعات زیاد بخواهد این کار را انجام دهد)، تایپ دستور نه تنها کسل کننده است، بلکه زمان بر نیز خواهد بود. برای رفع این مشکل می توان برای هر یک از کاربران نام برده، دیدی ایجاد کرد تا کاربران فقط بتوانند اطلاعات خودشان را بازیابی کنند. از طرف دیگر، چون در دید، کاربران به طور مستقیم به جداول بانک اطلاعات دسترسی ندارند، امنیت افزایش می یابد. برای ایجاد دید از دستور CREATE VIEW به صورت زیر استفاده می شود:

CREATE VIEW view_name(fields_list) **AS** SQL دستور
در این ساختار، view_name، نام دیدی است که می خواهید ایجاد کنید، fields_list، لیست فیلدهای دید را تعیین می کند.

 مثال ۱ - ۳. پرس و جویی که دیدی به نام groupBookView ایجاد می کند. این دید نام و نام گروه کتاب را بازیابی می کند.

```
CREATE VIEW groupBookView AS
SELECT Title, groupName
FROM GroupBook AS g , Books AS b
WHERE b.groupID = g.groupID
```



نکته: دیدها در پوشه Views مربوط به بانک اطلاعاتی نگهداری می شوند. برای این که ببینید آیا دید ایجاد شده است یا خیر. پوشه PublishDB را باز کرده، پوشه Views را کلیک مضاعف کنید تا دید ایجاد شده را مشاهده نمایید (شکل مقابل).

 مثال ۲ - ۳. پرس و جویی که دید groupBookView را اجرا می کند.

```
SELECT * FROM groupBookView
```

	Title	groupName
1	حل مسائل C++	برنامه نویسی
2	حل مسائل C#	برنامه نویسی
3	امنیت شبکه	شبکه رایانه ای
4	اصول طراحی پایگاه داده	پایگاه داده
5	اصول طراحی سیستم های گرا	برنامه نویسی
6	مدیریت استراتژیک	فناوری اطلاعات
7	زهیافت C++	برنامه نویسی

در هنگام ایجاد دید باید به نکات زیر دقت کنید:

- همانند جدول و اشیای دیگر، اگر دیدی را قبلاً ایجاد کرده باشید و بخواهید مجدداً آن را ایجاد نمایید، با پیام خطا مواجه خواهید شد.

۲-۳. برنامه‌نویسی در SQL

یکی از امکانات جالب SQL Server، برنامه‌نویسی از طریق T-SQL^۹ است. برای برنامه‌نویسی ابتدا باید دستورات مختلف T-SQL را بی‌آموزیم. در ادامه به این دستورات می‌پردازیم.

۱-۲-۳. متغیرها

همان‌طور که بیان گردید، متغیرها، نام‌هایی برای محل‌های از حافظه هستند که در طول اجرای برنامه می‌توان محتویات آن‌ها را تغییر داد. برای استفاده از متغیرها باید آن‌ها را تعریف کرده و مقداری را در آن قرار داد. برای تعریف متغیر در SQL می‌توان از دستور DECLARE به صورت زیر استفاده کرد:

[مقدار پیش‌فرض =] نوع متغیر نام متغیر @ DECLARE

رویه‌های ذخیره شده

رویه‌های ذخیره شده، تعدادی از دستورات SQL هستند که با هم در یک مجموعه قرار گرفته کامپایل می‌شوند و با یک دستور SQL قابل اجرا می‌باشند. رویه‌های ذخیره شده مزایای زیادی دارند که برخی از آن‌ها عبارت‌اند از:

۱. **سرعت اجرای رویه‌های ذخیره شده بالاست.** زیرا، رویه‌های ذخیره شده به صورت کامپایل شده در حافظه نهان^۱ بانک اطلاعات نگهداری می‌شوند. بنابراین، دستیابی به آن‌ها سریع‌تر انجام خواهد شد و از طرف دیگر، چون کامپایل شده‌اند، در هنگام اجرا نیاز به کامپایل شدن ندارند. پس، سرعت اجرای آن‌ها افزایش می‌یابد.

۲. **رویه‌های ذخیره شده برنامه‌نویسی ماژولی^۲ را امکان‌پذیر می‌سازند.** زیرا، یک بار رویه‌های ذخیره شده را می‌نویسید، کامپایل می‌کنید و چند بار آن‌ها را فراخوانی می‌نمایید تا اجرا شوند.

۳. **رویه‌های ذخیره شده ترافیک شبکه را کاهش می‌دهند.** زیرا، ممکن است رویه‌های ذخیره شده از چندین سطر تشکیل شوند. در حالت عادی، وقتی سرویس‌گیرنده^۳ چند خط را به عنوان پرس‌وجو برای سرویس‌دهنده ارسال می‌کند، ترافیک شبکه افزایش می‌یابد. در حالتی که تعداد سرویس‌گیرنده‌ها (کاربران) در محیط شبکه زیاد باشد، این موضوع بیشتر خودش را نشان می‌دهد. ولی، اگر از رویه‌های ذخیره استفاده کنید، جهت اجرا فقط کافی است یک خط را بفرستید تا رویه ذخیره شده اجرا گردد. چون بدنه رویه ذخیره شده در سمت سرویس‌دهنده^۴ قرار دارد.

۴. **افزایش قابلیت نگهداری و اعمال تغییرات.** یکی از مزایایی که برنامه‌نویسان و تولیدکنندگان سیستم‌های اطلاعاتی علاقه بیشتری به استفاده از رویه ذخیره شده دارند، مستقل بودن کد نوشته شده از برنامه

^۹.Transaction SQL

^۱.Cache Memory

^۲.Modular Programming

^۳.Client

^۴. Server

اجرایی ایجاد شده است. چون برای انجام کوچک‌ترین تغییر در الگوریتم اجرای برنامه نیاز به تولید مجدد سیستم و استقرار دوباره آن خواهد بود. لذا، نگهداری سیستم‌ها از این نظر دچار مشکلاتی خواهد شد. با استفاده از رویه‌های ذخیره شده می‌توان این تغییرات را ساده‌تر انجام داد تا نگهداری سیستم راحت‌تر شود. زیرا، کد رویه‌های ذخیره شده در اختیار برنامه‌نویسان است (باینری یا exe نیست)، بنابراین به راحتی قابل ویرایش و تغییر خواهد بود و نیازی نیست که مجدداً برنامه را کامپایل نماید و فایل اجرایی آن را دوباره بسازید.

۵. رویه‌های ذخیره شده را می‌توان به طور خودکار پس از راه‌اندازی SQL Server اجرا نمود. نام رویه‌های ذخیره شده در دید sys.objects و دستورات آن در دید sys.comments ذخیره می‌شوند.

جدول ۱-۳ پارامترهای دستور CREATE PROCEDURE	
پارامتر	هدف
procedure_name	نام رویه ذخیره شده‌ای را تعیین می‌کند که ایجاد می‌شود.
number	شماره‌ای را مشخص می‌کند که می‌توانید به رویه ذخیره شده تخصیص دهید تا برخی از رویه‌های ذخیره شده که در یک گروه قرار می‌گیرند را با این شماره بتوانید فراخوانی کنید.
@parameter	پارامترهایی را تعیین می‌کند که می‌توانید مقادیر را از طریق آن‌ها برای رویه ذخیره شده ارسال کنید.
data_type	نوع هر یک از پارامترهای رویه ذخیره شده را تعیین می‌کند.
VARYING	مجموعه نتایج را به عنوان پارامتر خروجی مشخص می‌کند.
DEFAULT	مقدار پیش‌فرض پارامتر رویه ذخیره شده را تعیین می‌کند. اگر رویه ذخیره شده را بدون مقدار فراخوانی کنید، این مقدار برای پارامتر رویه ذخیره شده در نظر گرفته می‌شود.
OUTPUT	نوع پارامتر را خروجی در نظر می‌گیرد.
WITH RECOMPILE	با هر فراخوانی رویه ذخیره شده دوباره کامپایل می‌گردد.
WITH ENCRYPTION	دستورات رویه ذخیره شده را رمزگذاری می‌کند تا قابل بازیابی نباشد.
FOR REPLICATION	رویه ذخیره شده را برای تکثیر ایجاد می‌کند که توسط سرویس گیرنده قابل اجرا نیست.
sql_Statement	دستوراتی هستند که رویه ذخیره شده را ایجاد می‌کنند.

۱-۳-۳. ایجاد رویه‌های ذخیره شده

رویه‌های ذخیره شده به عنوان یک شیء در بانک اطلاعات ذخیره می‌شوند که می‌توان با دستور

CREATE PROCEDURE آن‌ها را ایجاد نمود. این دستور به صورت زیر به کار می‌رود:

```
CREATE PROC[EDURE ] procedure_name [ ; number ]
[ { @parameter data_type }
  VARYING ] [ = DEFAULT ] [ OUTPUT ]
[ , ...n ]
[ WITH { RECOMPILE | ENCRYPTION } ]
[ FOR REPLICATION ]
```

AS sql_statement [...n]

پارامترهای این دستور در جدول ۱-۳ آمده‌اند.

در هنگام ایجاد رویه‌های ذخیره شده به نکات زیر دقت کنید:

- رویه‌های ذخیره شده می‌توانند تعداد ۲۱۰۰ پارامتر داشته باشند.
- پارامترهای رویه ذخیره شده دو نوع‌اند. پارامترهای ورودی و پارامترهای ورودی/خروجی. اگر بعد از نوع پارامتر چیزی ذکر نشود، پارامتر ورودی در نظر گرفته می‌شود. اما، اگر بعد از نوع پارامتر output آورده شود، پارامتر ورودی/خروجی منظور خواهد شد.
- در رویه‌های ذخیره شده امکان ارجاع به اشیا بانک اطلاعاتی وجود دارد.
- در هنگام صدا زدن رویه ذخیره شده می‌توان از نام کامل آن که شامل نام بانک اطلاعاتی، نام مالک و نام رویه ذخیره شده است، استفاده کرد.
- در رویه‌های ذخیره شده می‌توان اشیا جدیدی را ایجاد کرد، از آن‌ها استفاده نمود. اشیا یی که رویه ذخیره شده می‌تواند ایجاد کند، دو نوع‌اند:

۱. **اشیا موقت**، اشیا یی هستند که نام آن‌ها با # شروع می‌شود و با خاتمه رویه ذخیره از بین خواهند رفت.

۲. **اشیا موقت دائمی**، اشیا یی هستند که نام آن‌ها با ## شروع می‌شود که با خاتمه رویه ذخیره شده حذف نمی‌شوند.

- در رویه‌های ذخیره شده می‌توان متغیرهای را تعریف کرد. این متغیرها محلی منظور خواهند شد. چون با خاتمه رویه ذخیره شده از بین خواهند رفت.
- نام رویه ذخیره شده می‌تواند با کاراکترهای # یا ## شروع شود. در این صورت رویه ذخیره شده موقت خواهد بود.

➤ با دستور RETURN از طریق نام رویه ذخیره شده می‌توان یک مقدار عددی صحیح را برگرداند. چنانچه جلوی دستور RETURN مقداری ذکر نشود، مقدار پیش فرض صفر برگشت داده می‌شود که بیان می‌کند رویه ذخیره شده بدون خطا اجرا شد.

➤ در رویه ذخیره شده نمی‌توانید از دستورات CREATE، CREATE VIEW، CREATE RULE، CREATE TRIGGER و CREATE DEFAULT استفاده کرد.

➤ رویه‌های ذخیره شده در بانک فعلی ایجاد می‌گردند. یعنی، در هنگام ایجاد رویه ذخیره شده نمی‌توان نام بانک اطلاعاتی را تعیین کرد.

➤ دستورات رویه ذخیره شده حداکثر می‌تواند ۱۲۸ مگابایت باشد.

➤ اگر نام رویه ذخیره شده‌ای را با so_ شروع نمایید و آن را در بانک اطلاعاتی master ایجاد کنید، این رویه ذخیره شده به صورت سراسری تعریف خواهد شد. یعنی، از هر بانک اطلاعاتی می‌توان آن را اجرا نمود.

۲-۳-۳. انواع رویه‌های ذخیره شده

رویه‌های ذخیره شده چهار نوع‌اند:

۱. رویه‌های ذخیره شده افزودن رکورد به جدول

۲. رویه‌های ذخیره شده حذف رکورد از جدول

۳. رویه‌های ذخیره شده تغییر رکوردهای جدول

۴. رویه‌های ذخیره شده بازبینی اطلاعات جدول

رویه‌های ذخیره شده افزودن رکورد به جدول

این رویه ذخیره شده معمولاً اطلاعات فیلدهای جدول را به عنوان پارامتر دریافت کرده، اگر شرایط خاصی برقرار باشد، اطلاعات پارامتر را در جدول اضافه می‌کند، وگرنه از طریق پارامتر دیگری خطا رخ داده شده را به برنامه‌ای که رویه ذخیره شده را فراخوانی کرده است، برمی‌گرداند. در هنگام افزودن رکوردی به جدول باید تمام محدودیت‌ها و قیود تعریف شده، در جدول بررسی شود. برخی از مهم‌ترین این قیود در زیر آمده‌اند:

۱. اگر جدول کلید اولیه داشته باشد، باید بررسی شود که مقدار فیلد کلید اولیه در جدول از قبل وجود نداشته باشد. اگر این مقدار وجود داشته باشد، پارامتر خطا مقدار بگیرد. این عمل به صورت زیر انجام می‌شود:

```
IF پارامتر متناظر با فیلد کلیدی  
IN (نام جدول FROM (SELECT نام فیلد کلید  
BEGIN  
SET 'مقدار خطا' = پارامتر خروجی @  
END
```

۲. اگر جدول دارای کلید خارجی باشد، باید بررسی شود، مقدار فیلد کلید خارجی این جدول که از طریق این فیلد با جدول پایه ارتباط دارد، وجود داشته باشد. اگر وجود نداشته باشد، پارامتر خطا مقدار بگیرد. این عمل به صورت زیر انجام می‌شود:

```
IF نام پارامتر متناظر با کلید خارجی  
NOT IN (فیلد کلید اولیه متناظر کلید خارجی (SELECT  
FROM (جدول ارتباط  
BEGIN  
SET 'مقدار خطا' = پارامتر خروجی @  
END
```

مثال ۱۸-۳. پرس‌وجویی که رویه ذخیره شده‌ای به نام InsertBooks را در بانک اطلاعات PublishDB ایجاد می‌کند تا از طریق آن بتوان رکوردهایی را به جدول Books اضافه نمود.

```
CREATE PROC InsertBooks  
@ISBN varchar(۱۰), @Title varchar(۴۰),  
@Page int, @Price decimal(۱۵,۰),  
@editNo int, @printNo int, @groupID varchar(۶),  
@errorCode varchar(۱) output  
AS  
IF @ISBN IN (SELECT ISBN FROM Books)  
BEGIN  
SET @errorCode = '۱'  
RETURN ۱  
END  
ELSE IF @groupID NOT IN (SELECT groupID FROM GroupBook)  
BEGIN  
SET @errorCode = '۲'  
RETURN ۲  
END
```

```

ELSE IF @Page < ۰
BEGIN
SET @errorCode = '۳'
RETURN ۳
END
ELSE IF @Price < ۰
BEGIN
SET @errorCode = '۴'
RETURN ۴
END
ELSE IF @printNo < ۰
BEGIN
SET @errorCode = '۵'
RETURN ۵
END
ELSE IF @editNo < ۰
BEGIN
SET @errorCode = '۶'
RETURN ۶
END
ELSE
BEGIN
INSERT INTO Books (ISBN, Title, Page, Price, editNo,
printNo, groupID)
VALUES (@ISBN, @Title, @Page, @Price, @editNo,
@printNo, @groupID)
SET @errorCode = '۰'
RETURN ۰
END
GO

```

این دستورات، رویه ذخیره شده InsertBooks را ایجاد می کنند. در این رویه ذخیره شده اعمال زیر انجام می شود:

۱. در سطر اول، دستور CREATE PROC آورده شده است.
۲. در سطرهای ۲، ۳ و ۴، پارامترهای ورودی (همان مقادیر فیلدهای جدول Books آورده شده اند).
۳. در سطر ۵، پارامتر @errorCode از نوع ورودی / خروجی (output) تعریف شده است که کد خطای رخ داده شده را نگهداری می کند. اگر خطای رخ ندهد، این پارامتر مقدار '۰' را در خودش نگهداری می کند.
۴. شرط IF @ISBN چک می کند که اگر شایبک کتاب در جدول Books وجود دارد، مقدار پارامتر @errorCode برابر '۱' شود و رویه ذخیره شده مقدار ۱ را برگرداند.
۵. شرط ELSE IF @groupID چک می کند که اگر کد گروه کتاب در جدول GroupBook وجود ندارد، مقدار پارامتر @errorCode برابر '۲' شود و رویه ذخیره شده ۲ را برگرداند.
۶. شرط ELSE IF @Page < ۰ بررسی می کند که اگر در تعداد صفحات کتاب مقداری منفی وارد شده است، مقدار پارامتر @errorCode برابر '۳' شود و رویه ذخیره شده ۳ را برگرداند.
۷. شرط ELSE IF @Price < ۰ چک می کند که اگر برای قیمت کتاب منفی وارد گردید، پارامتر @errorCode برابر '۴' شود و رویه ذخیره شده ۴ را برگرداند.

۸. شرط ELSE IF @ printNO بررسی می کند که اگر برای فیلد نوبت چاپ مقدار منفی وارد شده باشد، مقدار پارامتر @errorCode برابر '۵' گردد و رویه ذخیره شده مقدار ۵ را برگرداند.

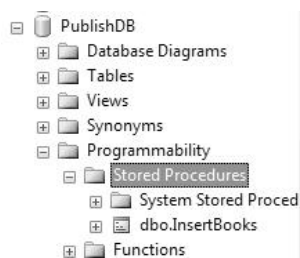
۹. شرط ELSE IF @editNo < ۰ بررسی می کند که اگر مقدار فیلد نوبت ویرایش منفی وارد شده باشد، مقدار پارامتر @errorCode برابر '۶' شود و رویه ذخیره شده ۶ را برگرداند.

۱۰. اگر هیچ یک از شرط های بیان شده برقرار نباشد، رکورد با دستور INSERT به جدول Books اضافه می شود و مقدار پارامتر @errorCode برابر '۰' خواهد شد. در پایان، رویه ذخیره شده ۰ را برمی گرداند.

Stored ایجاد خواهد شد در پوشه InsertBooks اگر این دستورات را اجرا کنید، رویه ذخیره شده قرار خواهد گرفت. برای PublishDB بانک اطلاعاتی Programmability موجود در پوشه Procedures دیدن این رویه ذخیره شده، مراحل زیر را انجام دهید:

۱. پوشه Databases را کلیک مضاعف کنید تا بانک های اطلاعاتی موجود در سرویس دهنده سیستم تان را ببینید.

۲. پوشه PublishDB را کلیک مضاعف کرده تا اشیای موجود در این بانک را مشاهده کنید.



۳. پوشه Programmability را کلیک مضاعف کرده تا اشیای برنامه - ای بانک اطلاعاتی PublishDB را مشاهده نمایید.

۴. پوشه Stored Procedures را کلیک مضاعف کرده تا پوشه رویه های ذخیره شده، ایجاد شده را ببینید (شکل مقابل):

۴ - ۳. انواع توابع تعریف شده توسط کاربر

توابع تعریف شده توسط کاربر بر اساس مقدار خروجی که می توانند برگردانند، به دو دسته زیر تقسیم می شوند:

۱. **توابعی که یک مقدار برمی گردانند**، این توابع فقط یک مقدار را برمی گردانند. اطلاعاتی که این توابع برمی گردانند، اسکالر (غیر جدولی) نام دارند. این توابع نمی توانند داده های نوع timestamp، انواع داده های غیر اسکالر مانند Table، Text، ntext، image، Cursor و انواع تعریف شده توسط کاربر را برگردانند. تعریف این توابع به صورت زیر است:

```
CREATE FUNCTION [schema_name.]function_name
([{@parameter_name}[AS] [type_schema_name.]parameter_data_type
[ = default ] [ READONLY ] ]
[ , ...n ]
)
RETURNS return_data_type
[ WITH <function_option> [ , ...n ] ]
```

```

[ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
[ ; ]

```

در این ساختار، schema_name، نام شیما و function_name، نام تابع را تعیین می‌کند، parameter_name، نام پارامتر، type_schema_name، نوع شیما و parameter_data_type، نوع پارامتر را مشخص می‌کند. پارامترهای default و READONLY به ترتیب مقدار پیش‌فرض و فقط خواندنی بودن پارامتر را تعیین می‌کنند. return_data_type، نوع داده‌ای که تابع برمی‌گرداند، می‌باشد. function_body بدنه (دستورات) تابع را تعیین می‌کند و scalar_expression، عبارتی است که نتیجه آن یک مقدار اسکالر می‌باشد و دستور RETURN آن را برمی‌گرداند.

در هنگام تعریف توابع که فقط یک مقدار را برمی‌گردانند، به نکات زیر توجه کنید:

- این توابع فقط یک مقدار را برگشت می‌دهند. پس باید حداقل یک دستور RETURN در بدنه آن‌ها وجود داشته باشد.

۲. توابع تعریف شده توسط کاربر حداکثر می‌توانند ۱۰۲۴ پارامتر داشته باشند.

 مثال ۳۵-۳. پرس‌وجویی که تابعی به نام getPubName ایجاد می‌کند تا نام و نام خانوادگی مدیر انتشارات را به عنوان پارامتر دریافت کرده، نام انتشارات را برگشت دهد.

```

CREATE FUNCTION getPubName(@bFname varchar(۱۰), @bLname
    varchar(۲۰))
RETURNS varchar(۳۰)
AS
BEGIN
    RETURN (SELECT pName FROM Publishers WHERE bFname=@bFname and
        bLname = @bLname)
END

```

 مثال ۳۶-۳. پرس‌وجویی که تابعی به نام getBookPrice ایجاد می‌کند تا نام کتاب را به عنوان پارامتر دریافت کرده، قیمت آن را برگرداند.

```

CREATE FUNCTION getBookPrice(@Title varchar(۲۰))
RETURNS int
AS
BEGIN
    RETURN (SELECT Price FROM Books WHERE Title =@Title)
END

```

۵-۴-۳. فراخوانی توابع خطی که یک جدول را برمی گردانند

از آنجایی که این توابع یک جدول را برمی گردانند، از طریق دستور SELECT می توان این توابع را فراخوانی کرد. نمونه های از فراخوانی این توابع را در پرس وجوهای بعدی خواهید دید.

 مثال ۴۳-۳. پرس وجویی که با فراخوانی تابع getBookAut() نام و قیمت کتاب های تألیف شده توسط رمضان عباس نژاد را نمایش می دهد.

```
SELECT * FROM getBookAut('رمضان', 'عباس نژاد')
```

	Title	Price
1	حل مسائل C++	57000
2	حل مسائل C#	62000
3	امنیت شبکه	65000
4	اصول طراحی پایگاه داده	73000
5	اصول طراحی سیستم های شی نگرا	130000
6	رہبافت C++	43000

همان طور که در این فراخوانی می بینید، در توابع خطی (الزامی dbo که جدولی را برمی گردانند، ذکر نام مالک) به جای نام جدول getBookAut() نیست (در این جا نام تابع قرار گرفته است). SELECT یا دید در دستور

نکته: البته در هنگام فراخوانی تابع خطی که جدولی را

می توان فیلدهای خاصی را بازایی کرد (پرس وجویی بعدی را ببینید). *برمی گرداند، به جای علامت

 مثال ۲۷. پرس وجویی که تابع MerchandiseGroup() را فراخوانی می کند.

```
SELECT * FROM dbo.MerchandiseGroup()
```

با اجرای این دستور خطای زیر ظاهر می شود:

```
Msg ۳۱۳, Level ۱۶, State ۳, Line ۱
An insufficient number of arguments were supplied for
the procedure or function dbo.MerchandiseGroup.
```

زیرا، چنانچه تابعی دارای پارامتری با مقدار پیش فرض باشد، برای فراخوانی تابع اگر نخواهید مقداری برای پارامتر پیش فرض ارسال کنید، حتماً باید کلمه DEFAULT را ذکر نمایید. پرس وجویی بعدی این موضوع را بیان می کند.

 مثال ۲۸. پرس وجویی که تابع MerchandiseGroup() را فراخوانی می کند تا اطلاعات کالاهای نوشیدنی را بازایی کند.

```
SELECT * FROM dbo.MerchandiseGroup(DEFAULT)
```

همان طور که در فراخوانی این تابع می بینید، کلمه DEFAULT به جای نام پارامتر ذکر شده است. با اجرای این دستور خروجی زیر ظاهر می شود:

	merchandiseID	merchandiseGroupID	merchandiseName	consumerPrice	buyPrice	pocketSellingPrice	chequeSellingPri
1	01	02	نوشابه بزرگ	6000	4000	5000	5500
2	02	02	دوغ بزرگ	7000	5000	5700	6000

همان طور که در این خروجی می بینید، فقط کالاهای نوشیدنی نمایش داده شده اند. چون، مقدار پیش فرض

نوشیدنی بود. @groupName پارامتر

 مثال ۲۹. پرس وجویی که با فراخوانی تابع MerchandiseGroup() لیست کالاهایی که نام گروه آنها حبوبات باشد را برمی گرداند.

```
SELECT * FROM dbo.MerchandiseGroup('حبوبات')
```

با اجرای این دستور خروجی زیر ظاهر می گردد:

	merchandiseID	merchandiseGroupID	merchandiseName	consumerPrice	buyPrice	pocketSellingPrice	chequeSellingPri
1	03	01	برنج	32000	25000	26000	26000

همان طور که در این خروجی مشاهده می گردد، فقط کالاهایی که در گروه حبوبات قرار دارند، نمایش

داده شده اند.

 مثال ۳۰. پرس وجویی که تابعی به نام GetPrice() ایجاد می کند تا نام کالا را به عنوان پارامتر دریافت کرده، قیمت خرید آن را برگرداند.

```
CREATE FUNCTION GetPrice(@merchandiseName varchar(۳۰) )
RETURNS DECIMAL(۱۸, ۰) AS
BEGIN
    RETURN (SELECT buyPrice FROM Merchandise WHERE
            merchandiseName=@merchandiseName) END
```

۷-۳. دستور کار آزمایشگاه

۱. پرس وجویی بنویسید که دیدی به نام DriversView ایجاد کند تا اطلاعات رانندگانی را بازیابی نماید که برای شهرهای آمل یا بابل سفارش حمل کردند.
۲. پرس وجویی بنویسید که دید DriversView را اجرا کند.
۳. پرس وجویی بنویسید که با استفاده از دید DriversView، نام و نام خانوادگی رانندگانی که سفرهای برای شهرهای بابل یا آمل حمل نموده اند را بازیابی کند.
۴. پرس وجویی بنویسید که دید DriversView را طوری تغییر دهد تا اطلاعات رانندگانی که برای شهر بابل و شیراز سفرهای حمل کردند، بازیابی کند.
۵. پرس وجویی بنویسید که دیدی به نام CusOrderView ایجاد کند تا اطلاعات مشتریانی که حداقل یک سفارش داشته اند را بازیابی کند.
۶. پرس وجویی بنویسید که دید CusOrderView را طوری تغییر دهد تا اطلاعات مشتریانی که اصلاً سفرهای نداشته اند، بازیابی کند.
۷. پرس وجویی بنویسید که دیدی به نام DrvCountOrderView ایجاد کند تا اطلاعات رانندگانی که بیش از ۲ سفارش انجام داده اند، بازیابی کند.
۸. پرس وجویی بنویسید که دیدی به نام sumFreightCusView ایجاد کند تا نام، نام خانوادگی و مجموع کرایه پرداخت شده توسط مشتریان را بازیابی کند.
۹. پرس وجویی بنویسید که دیدهای DrvCountOrderView و CusOrderView را حذف کند.
۱۰. پرس وجویی بنویسید که رویه ذخیره شده ای به نام InsertCars ایجاد کند تا اطلاعات ماشین را به عنوان پارامتر ورودی دریافت کرده، اطلاعات آن ماشین را به جدول Cars اضافه کند. در این رویه شرطهای

از قبیل تکراری نبودن شماره ماشین، موجود بودن شماره راننده در جدول رانندگان و منفی نبودن مقدار فیلد ظرفیت ماشین چک شود.

۱۱. پرس وجویی بنویسید که رویه ذخیره شده InsertCars را اجرا کرده و پیغام "شماره ماشین تکراری است" را نمایش دهد.

۱۲. پرس وجویی بنویسید که رویه ذخیره شده InsertCars را اجرا کرده و پیغام "شماره راننده موجود نیست" را نمایش دهد.

۱۳. پرس وجویی بنویسید که رویه ذخیره شده InsertCars را اجرا کرده و پیغام "ظرفیت منفی وارد شده است" را نمایش دهد.

۱۴. پرس وجویی بنویسید که رویه ذخیره شده InsertCars را اجرا کرده، رکوردی را به جدول Cars اضافه کرده، پیغام "رکورد درج شده" را نمایش دهد.

۱۵. پرس وجویی بنویسید که رویه ذخیره شده DeleteCars را ایجاد کند. در این رویه ذخیره شده اگر شماره ماشین در جدول Cars وجود نداشت، از طریق پارامتر خروجی یک (۱) را برگرداند، وگرنه ماشین را از جدول Cars حذف کرده، صفر (۰) را برگرداند.

۱۶. پرس وجویی بنویسید که شماره ماشین '۱۰' را به عنوان پارامتر ورودی دریافت کرده، آن ماشین را از طریق اجرای رویه ذخیره DeleteCars از جدول Cars حذف کند.

۱۷. پرس وجویی بنویسید که رویه ذخیره شده DeleteCustomer را ایجاد کرده، تا شماره مشتری را به عنوان دریافت کرده، اگر آن شماره مشتری در جدول Customers وجود دارد، آن را حذف کرده، مقدار صفر (۰)، وگرنه مقدار یک (۱) را برگرداند.

۱۸. پرس وجویی بنویسید که رویه ذخیره شده DeleteCustomer را طوری تغییر دهد تا اگر شماره مشتری در جدول Orders موجود بود، دو (۲) را برگرداند.

۱۹. پرس وجویی بنویسید که تابعی به نام CountDrvOrder ایجاد کند تا نام و نام خانوادگی راننده‌ای را به عنوان پارامتر دریافت کرده، تعداد سفارش‌های انجام شده توسط آن راننده را برگرداند.

۲۰. پرس وجویی بنویسید که با دستور SELECT تابع CountDrvOrder() را فراخوانی کند تا تعداد سفارش‌های انجام شده توسط احمد احمدی را نمایش دهد.

۲۱. پرس وجویی بنویسید که با دستور EXEC تابع CountDrvOrder() را فراخوانی کند تا تعداد سفارش‌های انجام شده توسط راننده رضا رضازاده را نمایش دهد.

۲۲. پرس وجویی بنویسید که تابعی به نام drvName ایجاد کند تا نام و نام خانوادگی رانندگانی را برگرداند که حداقل یک سفارش انجام داده‌اند (با گزینه INNER JOIN).

۲۳. پرس وجویی بنویسید که تابع drvName() را فراخوانی کند. به خروجی دقت کنید، رکوردهای تکراری می‌بینید.

۲۴. پرس وجویی بنویسید که تابع `drvName()` را طوری تغییر دهد تا نام و نام خانوادگی های تکراری را فقط یک بار برگرداند.
۲۵. پرس وجویی بنویسید که تابع `drvName()` را فراخوانی کند. به خروجی دقت نمایید، رکوردهای تکراری فقط یک بار ظاهر گردیدند.
۲۶. پرس وجویی بنویسید که تابعی به نام `CusName()` ایجاد کند تا نام و نام خانوادگی مشتریانی را بازیابی کند که سفارش داشته اند.
۲۷. پرس وجویی بنویسید که با استفاده از فراخوانی تابع `CusName()` تعداد مشتریانی که سفارش داشته اند را نمایش دهد.
۲۸. پرس وجویی بنویسید که با استفاده از فراخوانی تابع `CusName()`، مشتریانی که نام آن ها رضا است و سفارش داشته اند را بازیابی کند.
۲۹. پرس وجویی بنویسید که تابعی به نام `keyReturns()` ایجاد کند تا نام جدول را به عنوان پارامتر دریافت کرده، محدودیت هایی تعریف شده در آن جدول را برگرداند.
۳۰. پرس وجویی بنویسید تا تابع `keyReturns()` را برای جدول `Orders` فراخوانی کند.
۳۱. پرس وجویی بنویسید تا تابع `keyReturns()` را طوری فراخوانی کند تا کلیدهای خارجی جدول `Orders` را نمایش دهد.
۳۲. پرس وجویی بنویسید تا تابع `keyReturns()` را طوری فراخوانی کند تا کلید اولیه جدول `Customers` را برگرداند.
۳۳. پرس وجویی بنویسید تا تابع `CountDrvOrder()` را حذف کند.

۹-۳. تمرین

۱. دیدی ایجاد کنید تا نام کل و نام معین را نمایش دهد.
۲. دیدی ایجاد کنید تا نام معین و نام تفضیلی را نمایش دهد.
۳. دیدی ایجاد کنید تا شماره سند، تاریخ سند، نام کل، مبلغ بدهکار و مبلغ بستانکار کلیه اسناد را نمایش دهد.
۴. با استفاده از دید ایجاد شده در سوال ۳، دستوری بنویسید تا اطلاعات سند شماره ۱۰۰ را بازیابی کند.
۵. دیدی ایجاد کنید تا نام هزینه، نام پروژه، تاریخ، مبلغ بدهکار و مبلغ بستانکار را بازیابی کند.
۶. دیدی ایجاد کنید تا نام پروژه، مجموع بدهکار و بستانکار آن پروژه را بازیابی کند.
۷. دیدی ایجاد کنید تا نام کل، مجموع بدهکار و مجموع بستانکار هر کل را نمایش دهد.
۸. دیدی ایجاد کنید تا نام کل، نام معین، مجموع بدهکار و مجموع بستانکار هر کل و معین را نمایش دهد.
۹. دیدی ایجاد کنید تا نام گروه حساب، مجموع بدهکار و بستانکار هر گروه حساب را نمایش دهد.
۱۰. دیدی ایجاد کنید تا تعداد رکوردهای هر سند را بازیابی کند.
۱۱. دیدی ایجاد کنید تا مجموع بدهکار و بستانکار هر سند را بازیابی کند.

۱۲. دیدی ایجاد کنید تا نوع حساب، نام کل، تاریخ، مبلغ بدهکار و بستانکار کلیه اسناد را نمایش دهد.
۱۳. دید ایجاد شده سوال ۱۲ را طوری تغییر دهید تا علاوه بر اطلاعات بیان شده، نام معین و نام تفضیلی را نیز بازیابی کند.
۱۴. دیدهای ایجاد شده در سوال‌های ۱ و ۷ را حذف کنید.
۱۵. دیدی ایجاد کنید تا کلیه اسناد ضرب دری را بازیابی کند (اسنادی ضرب دری هستند که مجموع مبلغ بدهکار و بستانکار آن‌ها برابر نباشد).
۱۶. پرس‌وجویی بنویسید تا مقادیر کد نوع حساب و نام نوع حساب را عنوان پارامتر ورودی دریافت کرده، رکوردی به جدول نوع حساب اضافه کند. در این پرس‌وجو شرط‌های زیر بررسی شود:
- 🚩 اگر کد نوع حساب قبلاً در جدول نوع حساب وجود داشت، رویه ذخیره شده، مقدار یک (۱) را برگرداند.
- 🚩 وگرنه، اگر نام نوع حساب قبلاً در جدول نوع حساب وجود داشت، رویه ذخیره شده دو (۲) را برگشت دهد.
- 🚩 وگرنه، یک رکورد به جدول نوع حساب اضافه کرده، مقدار صفر (۰) را برگرداند.
۱۷. پرس‌وجویی بنویسید که رویه ذخیره شده‌ای ایجاد کرده تا کد کل را به عنوان پارامتر ورودی دریافت کرده، آن کل را از جدول کل حذف کند. در این رویه ذخیره شده تست شود، اگر کد کل در جدول کل وجود نداشت، رویه ذخیره شده یک (۱) را برگرداند.
۱۸. پرس‌وجویی بنویسید که رویه ذخیره شده در سوال ۱۷ را طوری تغییر دهد تا علاوه بر شرایط بیان شده، شرط‌های زیر را نیز تست کند.
- 🚩 اگر کد کل در جدول معین وجود داشت، رویه ذخیره شده، دو (۲) را برگشت دهد.
- 🚩 اگر کد کل در جدول تفضیلی موجود بود، رویه ذخیره شده، سه (۳) را برگرداند.
- 🚩 اگر کد کل در جدول سند وجود داشت، رویه ذخیره شده، چهار (۴) را برگرداند.
۱۹. پرس‌وجویی بنویسید که رویه ذخیره شده‌ای ایجاد کند تا پارامترهای کد کل، کد معین، کد تفضیلی و نام تفضیلی را به صورت ورودی دریافت کرده، نتیجه اجرای رویه ذخیره شده را از طریق یک پارامتر خروجی برگرداند. در این رویه ذخیره شده شرایط زیر بررسی شود:
- 🚩 اگر کد کل در جدول کل وجود نداشت، پارامتر خروجی مقدار عددی یک (۱) را بگیرد.
- 🚩 وگرنه، اگر کد معین در جدول معین وجود نداشت، پارامتر خروجی مقدار عددی دو (۲) را بگیرد.
- 🚩 وگرنه، اگر کد کل، کد تفضیلی و کد معین در جدول تفضیلی وجود دارند، رویه ذخیره شده مقدار عددی سه (۳) را در پارامتر خروجی قرار دهد.

🌈 وگرنه رکورد مورد نظر را به جدول تفضیلی اضافه کرده، مقدار صفر (۰) را در پارامتر خروجی قرار دهد.

۲۰. پرس وجویی بنویسید که رویه ذخیره شده‌ای ایجاد کند تا جدول پروژه را حذف کند.

۲۱. پرس وجویی بنویسید که رویه ذخیره شده ایجاد شده در سوال ۲۰ را طوری تغییر دهد تا اگر رکوردی از جدول پروژه در جدول سند، موجود بود، رویه ذخیره شده مقدار یک را برگرداند، وگرنه جدول پروژه را حذف کرده، مقدار صفر را برگرداند (با دستور ALTER PROC).

۲۲. رویه ذخیره شده ایجاد شده در سوال‌های ۱۹ و ۲۱ را حذف کنید (با دستور DROP PROC).

۲۳. پرس وجویی بنویسید که تابعی ایجاد کند تا شماره سند را به عنوان پارامتر دریافت کرده، تعداد رکوردهای آن سند را برگرداند.

۲۴. پرس وجویی بنویسید که تابعی ایجاد کند تا شماره سند، مجموع بدهکار، مجموع بستانکار اسنادی را برگرداند که مجموع بدهکار آن‌ها کمتر از مجموع بستانکار آن‌ها می‌باشد.

۲۵. پرس وجویی بنویسید که تابعی ایجاد کند تا تاریخی را به عنوان پارامتر ورودی دریافت کرده، تعداد اسناد ثبت شده در آن تاریخ را برگرداند.

۲۶. پرس وجویی بنویسید که تابعی ایجاد کند تا شماره سند، نام کل، نام معین، نام تفضیلی اسنادی را برگرداند که در شرح سند آن‌ها عبارت خاصی وجود داشته باشد (این عبارت را از طریق یک پارامتر ورودی برای تابع ارسال کنید).

۲۷. پرس وجویی بنویسید که تابعی ایجاد کند تا نام کل‌هایی را بازیابی کند که اختلاف مجموع بدهکار و مجموع بستانکار آن‌ها از مقدار خاصی بزرگ‌تر است (مقدار که اختلاف مجموع بدهکار و بستانکار با آن مقایسه می‌شود، از طریق پارامتر ورودی برای تابع ارسال می‌گردد، چنانچه چیزی ذکر نشود، به طور پیش فرض صفر در نظر گرفته شود).

۲۸. پرس وجویی بنویسید که تابع بیان شده در سوال ۲۷ را یک بار با دستور EXEC و بار دیگر بدون دستور EXEC فراخوانی کند.

۲۹. پرس وجویی بنویسید تا تابع ایجاد شده در سوال ۲۷ را حذف کند (با دستور DROP FUNCTION).

۳۰. پرس وجویی بنویسید که تابعی ایجاد کند تا مقدار رکوردهای سند با نوع خاص را برگرداند (نام نوع سند به عنوان پارامتر برای تابع ارسال می‌شود و مقدار پیش فرض این پارامتر 'اسناد موقت' است).

۳۱. پرس وجویی بنویسید تا تابع ایجاد شده در سوال ۳۰ را یک مرتبه بدون دستور EXEC و مرتبه دیگر با دستور EXEC فراخوانی کند.

تاکنون با مراحل ایجاد بانک اطلاعاتی، ایجاد و دست کاری اشیای آن از قبیل جداول، دیدها، رویه‌های ذخیره شده و توابع آشنا شدید. در این فصل به مفاهیم پیشرفته زیر در SQL Server می‌پردازیم:

۱. بازیابی اطلاعات از کاتالوگ

۲. پشتیبان گیری و بازیابی پشتیبان

۳. تریگرها، کاربرد و ایجاد آنها

۴. کرسرها، کاربرد و ایجاد آنها

۴-۱. بازیابی اطلاعات از کاتالوگ

کاتالوگ^{۱۱} یا فرهنگ داده را می‌توان یک بانک اطلاعات در نظر گرفت (یک بانک سیستمی، نه بانک اطلاعات کاربر) که حاوی داده‌هایی راجع به اشیاء موجود در بانک اطلاعات است. داده‌هایی که در کاتالوگ نگهداری می‌شوند، شبه داده یا توصیف نامیده می‌شوند (یعنی سایر اشیای بانک اطلاعات را توصیف می‌کنند). هر کاتالوگ اطلاعاتی از قبیل ساختار جداول، ساختار دیدها، ساختار شاخص، نام بانک‌های اطلاعات، اندازه اشیاء، حساب کاربران، حقوق دست‌یابی و ایمنی کاربران و اطلاعات دیگر را نگهداری می‌کند. سیستم کاتالوگ توسط سیستم مدیریت بانک اطلاعاتی^{۱۲} ایجاد و دست کاری می‌شود.

جدول ۱ - ۴ برخی از اشیای موجود در دید INFORMATION_SCHEMA	
نام دید	هدف
SCHEMATA	همه بانک‌های اطلاعات قابل دست‌یابی توسط کاربر را نگهداری می‌کند.
TABLES	هر جدول تعریف شده برای بانک اطلاعاتی فعلی را نگهداری می‌کند.
VIEW_TABLE_USAGE	جداول استفاده شده برای دیدها را مشخص می‌کند.
COLUMNS	فیلدهای تعریف شده در هر جدول بانک اطلاعات فعلی که توسط کاربر قابل دست‌یابی است.
VIEWS	دیدهای قابل دست‌یابی توسط کاربر برای بانک اطلاعات فعلی را نگهداری می‌کند.
ROUTINES	رویه‌های ذخیره شده یا توابع که در بانک اطلاعات تعریف شده‌اند را نگهداری می‌کند.
PARAMETERS	پارامترهای تعریف شده برای توابع و رویه‌های ذخیره شده بانک اطلاعات فعلی را نگهداری می‌کند.
و غیره	

^{۱۱}. Catalog

^{۱۲}. DBMS(Database Management System)

هر بانک اطلاعات SQL Server دیدی به نام INFORMATION_SCHEMA دارد. اشیای زیادی در این دید قرار دارند. برخی از این اشیاء در جدول ۱-۴ آمده‌اند.

هر یک از این دیدها دارای فیلدهای مختلفی هستند که بحث در باره همه آنها در این کتاب امکان‌پذیر نمی‌باشد. به عنوان مثال، دید TABLES دارای فیلدهایی به نام TABLE_CATALOG (نام بانک اطلاعات جدول)، TABLE_SCHEMA (شما جدول)، TABLE_NAME (نام جدول) و TABLE_TYPE (نوع جدول) است.

مثال ۱-۴. پرس‌وجویی که نام جداول و دیدهای بانک اطلاعاتی فعلی را نمایش می‌دهد.

```
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
```

به جای دید INFORMATION_SCHEMA می‌توانید از شیء sys.all_objects استفاده کنید. این شیء فیلدهای متعددی دارد (فیلد name (نام شیء)، فیلد type (نوع شیء) را تعیین می‌کند). نوع شیء می‌تواند مقادیر زیادی داشته باشد که برخی از آنها عبارت‌اند از:

۱. مقدار U (جدول تعریف شده توسط کاربر)
۲. مقدار V (شیء از نوع دید است)
۳. مقدار P (شیء از نوع رویه ذخیره شده است)
۴. و غیره.

مثال ۲-۴. پرس‌وجویی که نام جداول و دیدهای بانک اطلاعاتی فعلی را نمایش می‌دهد.

```
SELECT sys.all_objects.name FROM sys.all_objects
WHERE sys.all_objects.Type='U' OR sys.all_objects.Type='V'
```

علاوه بر اشیای بیان شده، دیدهای دیگری وجود دارند که از طریق آنها می‌توان اشیای بانک اطلاعات را بازیابی کرد. برخی از مهم‌ترین این اشیاء در جدول ۲-۴ آمده‌اند.

مثال ۳-۴. پرس‌وجویی که لیست رویه‌های ذخیره شده بانک اطلاعاتی PublishDB را نمایش می‌دهد.

```
SELECT * FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE = 'PROCEDURE'
```

	SPECIFIC_CATALOG	SPECIFIC_SCHEMA	SPECIFIC_NAME	ROUTINE_CATALOG	ROUTINE_SCHEMA	ROUTINE_NAME	ROUTINE_TYPE
1	PublishDB	dbo	InsertBooks	PublishDB	dbo	InsertBooks	PI
2	PublishDB	dbo	GroupBookP	PublishDB	dbo	GroupBookP	PI
3	PublishDB	dbo	DeleteBooks	PublishDB	dbo	DeleteBooks	PI
4	PublishDB	dbo	UpdateBooks	PublishDB	dbo	UpdateBooks	PI

۴-۲. پشتیبان‌گیری از اطلاعات

یکی از بخش‌های مهم نرم‌افزار حفظ و نگهداری اطلاعات است. زیرا ممکن است اطلاعات حذف شوند. برخی از دلایل حذف اطلاعات عبارت‌اند از:

۱. دیسک سخت توسط افراد غیرمجاز مورد دست‌یابی قرار گیرد و آنها اعمالی مثل فرمت یا حذف اطلاعات را انجام دهند.
۲. دیسک سخت خراب شود و قابل بازیابی نباشد.
۳. اطلاعات نادرست بر روی اطلاعات فعلی کپی گردد (اشتباه).

۴. ویروس‌های رایانه‌ای، اطلاعات را تخریب کنند.

بنابراین در این بخش به پشتیبان‌گیری و بازیابی اطلاعات پشتیبان از بانک اطلاعات می‌پردازیم.

جدول ۳-۴ پارامترهای دستور BACKUP DATABASE	
پارامتر	هدف
database_name	نام بانک اطلاعاتی است که باید از آن پشتیبان گرفته شود.
@database_name_var	نام بانک اطلاعاتی که باید از آن پشتیبان گرفته شود، در این متغیر قرار دارد.
backup_device	نام دستگاهی را تعیین می‌کند که پشتیبان باید در آن قرار داده شود.
DESCREPTION	توصیف پشتیبان را مشخص می‌نماید.
BLOKSIZE	اندازه فیزیکی بلوک را به بایت تعیین می‌کند.
DIFFERENTIAL	از اطلاعاتی که تغییر یافته‌اند، پشتیبان می‌گیرد.
EXPIREDATE	تاریخ و زمان انقضای پشتیبان را تعیین می‌کند.
RETAIN DAYS	تعیین می‌نماید چند روز بعد می‌توان بر روی این مجموعه پشتیبان، بازنویسی (رونویسی) کرد.
PASSWORD	کلمه عبور بر روی پشتیبان قرار می‌دهد تا هر کاربری نتواند آن را بازیابی کند.
FORMAT	تعیین می‌کند سرآیند ^{۱۲} رسانه باید بر روی تمام Volume‌هایی که برای عمل پشتیبان استفاده شده‌اند، نوشته شود.
NOFORMAT	تعیین می‌کند سرآیند رسانه بر روی تمام Volume‌هایی که برای عمل پشتیبان گیری استفاده شده‌اند، نوشته نشود.
INIT	تمام مجموعه پشتیبان به جز سرآیند رسانه باید بازنویسی شود.
NOINIT	مجموعه پشتیبان را به یک دستگاه دیسک یا نوار اضافه می‌کند و مجموعه پشتیبان‌های قبلی را حفظ می‌کند.
MEDIADESCRIPTION	توصیف رسانه را مشخص می‌کند.
MEDIANAME	نام رسانه را تعیین می‌کند که حداکثر ۱۲۸ کاراکتر است.
MEDIAPASSWORD	کلمه عبور را بر روی رسانه قرار می‌دهد تا هر کاربری نتواند آن رسانه را باز کند.
NAME	نام پشتیبان را مشخص می‌کند که حداکثر ۱۲۸ کاراکتر می‌باشد.
NOSKIP	سرآیند رسانه را می‌خواند و تست‌های از قبیل کلمه عبور را انجام می‌دهد و اگر پارامتر MEDIANAME تعیین گردد، چک می‌کند آیا نام رسانه دریافتی با نام رسانه در سرآیند برابر است یا خیر.
SKIP	تاریخ انقضای و تست اولیه مجموعه پشتیبان را غیر فعال می‌کند.
NOUNLOAD	پس از پشتیبان‌گیری، نوار را از نوار خوان بیرون نمی‌آورد.
UNLOAD	پس از پشتیبان‌گیری، نوار را از نوار خوان بیرون می‌آورد.
NOREWIND	پس از پشتیبان‌گیری، نوار را به عقب برنمی‌گرداند.
REWIND	پس از پشتیبان‌گیری، نوار را به عقب برمی‌گرداند.
RESTART	اگر پشتیبان‌گیری قطع شده باشد، پشتیبان‌گیری را از مکانی که قطع شده است، ادامه می‌دهد.
STATS	فاصله زمانی را تعیین می‌کند که باید درصد پیشرفت کار نمایش داده شود.

۱-۲-۴. پشتیبان گیری با دستور BACKUP DATABASE

روش های متعددی برای پشتیبان گیری از بانک اطلاعات SQL Server وجود دارد. کامل ترین روش پشتیبان گیری، استفاده از دستور BACKUP DATABASE است. زیرا، این دستور، از بانک اطلاعات، فایل های کارنامه^{۱۳} و گروه فایل ها (Filegroups) پشتیبان می گیرد. این دستور، به صورت زیر به کار می رود:

```
BACKUP DATABASE { database_name | @database_name_var }
TO < backup_device > [ ,...n ]
[ [MIRROR TO < backup_device >[ ,...n ] [ ...next-mirror ] ]
[ WITH
[ BLOCKSIZE = { blocksize | @blocksize_variable } ]
[ [ , ] { CHECKSUM | NO_CHECKSUM } ]
[ [ , ] { STOP_ON_ERROR | CONTINUE_AFTER_ERROR } ]
[ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] DIFFERENTIAL ]
[ [ , ] EXPIREDATE = { date | @date_var }
| RETAINDAYS = { days | @days_var } ]
[ [ , ] PASSWORD = { password | @password_variable } ]
[ [ , ] { FORMAT | NOFORMAT } ]
[ [ , ] { INIT | NOINIT } ]
[ [ , ] { NOSKIP | SKIP } ]
[ [ , ] MEDIADESCRIPTION = { 'text' | @text_variable } ]
[ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
[ [ , ] MEDIAPASSWORD={mediapassword|@mediapassword_variable} ]
[ [ , ] NAME = { backup_set_name | @backup_set_name_var } ]
[ [ , ] { NOWIND | REWIND } ]
[ [ , ] { NOUNLOAD | UNLOAD } ]
[ [ , ] RESTART ]
[ [ , ] STATS [ = percentage ] ]
[ [ , ] COPY_ONLY
```

پارامترهای این دستور در جدول ۳-۴ آمده است.

 **مثال ۱۳-۴.** پرس و جویی که از بانک اطلاعاتی PublishDB پشتیبان گرفته، در پوشه C:\SQLBACKUP قرار می دهد (پوشه SQLBACKUP در درایو C باید ایجاد شده باشد).

```
BACKUP DATABASE PublishDB TO
DISK='C:\SQLBACKUP\PublishDB.bak'
```

GO

```
Processed 208 pages for database 'PublishDB', file 'publish_data' on file 1.
Processed 4 pages for database 'PublishDB', file 'publish_log' on file 1.
BACKUP DATABASE successfully processed 212 pages in 0.551 seconds (3.004 MB/sec).
```

۳-۴. تریگرها

تریگرها^{۱۴}، نوعی از رویه های ذخیره شده هستند که کاربردهای از قبیل حافظت از داده ها در برابر تغییرات، پنهان سازی قوانین کاری، ایجاد تغییرات آبخاری نظیر تغییرات به روز رسانی و حذف آبخاری، حفظ تمامیت داده ای و ارجاعی و جلوگیری از ایجاد ناسازگاری در داده را دارند. نمونه های از این کاربردها با مثال در زیر آمده اند:

^{۱۳} - Log
^{۱۴} .Triggers

را در نظر بگیرید. تریگرهای می توان برای جداول این بانک PublishDB به عنوان مثال، بانک اطلاعات اطلاعاتی تعریف کرد تا اعمال زیر را انجام دهد:

۱. در هنگام افزودن کتابی به جدول Books، اگر کد گروه کتاب در جدول GroupBook وجود ندارد، آن کتاب را به جدول Books اضافه نکند.

۲. در هنگام حذف رکوردی از جدول GroupBook، اگر کد گروه کتاب در جدول Books موجود باشد، از حذف رکورد در جدول GroupBook جلوگیری خواهد کرد یا ابتدا رکوردهایی از جدول Books با کد گروه مورد نظر را حذف خواهد کرد. سپس، این رکورد را از جدول GroupBook حذف می نماید. این نوع حذف از جدول را حذف آبشاری می نامند.

۳. در هنگام اضافه کردن کتاب به جدول Books، اگر قیمت کتاب، نوبت چاپ یا نوبت ویرایش منفی وارد شود، از اضافه شدن کتاب به جدول Books جلوگیری خواهد کرد.

برخی از ویژگی های تریگرها در زیر آمده اند:

۱. تریگرها فاقد پارامتر هستند، اما رویه های ذخیره شده می توانند پارامتر داشته باشند.

۲. تریگرها به طور صریح فراخوانی نمی شوند. یعنی، تریگرها در هنگام دست کاری داده ها جدول فعال می گردند (به طور ضمنی فراخوانی می گردند)، ولی رویه های ذخیره شده برای اجرا باید به طور صریح فراخوانی شوند.

۳. یک تریگر، فقط بر روی یک جدول یا دید ایجاد می گردد. اما، رویه های ذخیره شده می توانند بر روی چند جدول کار کنند.

۴. اگر جدولی را حذف کنید، تریگرهای آن به طور خودکار حذف خواهند شد. ولی، با حذف جدول رویه های ذخیره شده ای که از آن جدول استفاده می کنند، حذف نمی شوند.

۵. تریگرها انواع مختلف دارند که فقط تریگرهای Instead of را می توانید بر روی دیدها ایجاد کنید.

۶. تریگرها می توانند به صورت تودرتو فراخوانی شوند. یعنی، یک تریگر می تواند تریگر دیگری را فراخوانی کرده و باعث فعال شدن آن شود.

۷. تریگرها نمی توانند به صورت بازگشتی (چه مستقیم و چه غیر مستقیم) خودشان را فراخوانی کنند.

۸. تریگرها به دستورات دست کاری کننده حساس هستند (یعنی به INSERT، UPDATE و DELETE حساس هستند)، نه به رکوردهای که تغییر می یابند. به عبارت دیگر، اگر تریگری حساس به دستور UPDATE باشد و یک دستور UPDATE هزار رکورد جدول را تغییر دهد، تریگر یک بار فعال می شود نه هزاربار (یعنی، به ازای تغییر هر رکورد فعال نمی شود، بلکه به ازای هر بار اجرای دستور UPDATE فعال خواهد شد).

۹. اگر هر يك از محدودیت‌های (قيود) NULL، PRIMARY KEY، CHECK CONSTRAINTS و غیره نقض شوند، هیچ تریگری بر روی آن جدول اجرا نمی‌گردد. چون، تریگرها پس از انجام تغییرات بر روی داده، فعال می‌شوند. اما، محدودیت‌ها قبل از انجام تغییرات داده بررسی می‌گردند.

۱۰. تریگرها را نمی‌توان برای جداول موقت ایجاد نمود، ولی آن‌ها می‌توانند به دیدها یا جداول موقت، رجوع داشته باشند.

۱۱. چون دستورات تریگرها نمی‌توانند مجموعه‌ای از رکوردها را برگردانند، بنابراین نمی‌توان از عملگر EXISTS در دستور SELECT یا بخشی از IF در تریگر استفاده نمود.

۱۲. اگر در تریگری از دستور ROLLBACK استفاده شود، کل تراکنش لغو خواهد شد.

۱-۳-۴. انواع تریگرها

تریگرها به دو دسته تقسیم‌بندی می‌شوند که عبارت‌اند از:

🌈 **زمان فعال شدن تریگرها**، تریگرها می‌توانند از لحاظ زمان فعال شدن به دو نوع دیگر تقسیم شوند که عبارت‌اند از:

۱. **تریگرهای نوع AFTER**، پس از اجرای دستورات دست‌کاری کننده فعال می‌شوند. این تریگرها را نمی‌توان روی دیدها ایجاد نمود.
۲. **تریگرهای نوع INSTEAD OF**، به جای دستورات دست‌کاری کننده جداول و دیدها فعال می‌شوند.

🌈 **نوع دستور فعال کننده تریگر**، این تریگرها می‌توانند به سه دسته تقسیم شوند که عبارت‌اند از:

۱. **حساس به دستور INSERT**، در هنگام اجرای دستور INSERT به جدول فعال می‌شوند.
۲. **تریگر حساس به UPDATE**، در هنگام اجرای دستور UPDATE فعال می‌گردند.
۳. **تریگرهای حساس به DELETE**، در زمانی که دستور DELETE اجرا می‌شود، فعال خواهند شد.

۲-۳-۴. ایجاد تریگر

برای ایجاد تریگر از دستور CREATE TRIGGER به صورت زیر استفاده می‌شود:

```
CREATE TRIGGER [ schema_name . ]trigger_name
{ ON { table | view
[ [ WITH <dml_trigger_option> [ ,...n ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ,...n ] | EXTERNAL NAME
{ < [ ; ] <method specifier
```

پارامترهای این ساختار عبارت‌اند از:

۱. **schema_name**، نام مالک جدول را تعیین می‌نماید.
۲. **trigger_name**، نام تریگر است.
۳. **table**، نام جدلی است که تریگر بر روی آن ایجاد می‌شود.
۴. **view**، نام دیدی است که تریگری بر روی آن ایجاد می‌گردد.
۵. **AFTER**، نوع تریگر را تعیین می‌کند.

۶. **INSTEAD OF**، نوع تریگر را **INSREAD OF** تعیین می نماید. ۷. **UPDATE**، **INSERT** و **DELETE**، زمان فعال شدن تریگر را تعیین می کنند. ۸. **WITH APPEND**، با اضافه شدن را تعیین می کند. ۹. **NOT FOR REPLICATION**، تعیین می کند که تریگر در زمان تکثیر فعال نشود. ۱۰. **SQL_statement**، دستور **SQL** مربوط به تریگر را تعیین می کند. ۱۱. **EXTERNAL NAME**، نام بیرونی تریگر را تعیین می کند.

دستورات زیر را نمی توانید در بدنه تریگر استفاده کنید:

- | | |
|--------------------------|----------------------------|
| ۱. ALTER DATABASE | ۲. CREATE DATABASE |
| ۳. DROP DATABASE | ۴. RESTORE DATABASE |
| ۵. LOAD DATABASE | ۶. LOAD LOG |
| ۷. RESTORE LOG | ۸. DISK INIT |
| ۹. DISK RESIZE | ۱۰. RECONFIGURE |

همان طور که بیان گردید، تریگرها دارای نوع **AFTER** و **INSTEAD OF** هستند. این تریگرها ویژگی های زیر را دارند:

۱. تریگرها اطلاعات خودشان را در دو جدول **Deleted** و **Inserted** درج می کنند. این دو جدول در حافظه اصلی و به صورت موقت ایجاد می گردند که فقط در بدنه تریگر به آن ها می توان مراجعه نمود. یعنی، خارج از تریگر نمی توان به این جداول دسترسی داشت تا اطلاعات آن ها را بازیابی کرد.
۲. در تریگرهای نوع **AFTER**، وقتی دستور فعال کننده تریگر، رکورد یا رکوردهای را به جدول پایه اضافه می نماید، آن رکوردها از به جدول **Inserted** نیز اضافه خواهند شد.
۳. در تریگرهای نوع **AFTER**، وقتی فعال کننده، تریگر رکورد یا رکوردهای را از جدول پایه حذف می نماید، این رکوردها از جدول **Deleted** حذف خواهند کرد.
۴. در تریگرهای نوع **AFTER**، وقتی فعال کننده تریگر رکورد یا رکوردهایی را در جدول پایه تغییر می دهد، رکوردهای قبل از تغییر را در جدول **Deleted** و رکوردهای بعد از تغییر را در جدول **Inserted** اضافه خواهد کرد.
۵. در تریگرهای نوع **INSTEAD OF**، دستور فعال کننده تریگر، رکوردها را فقط به جدول **Deleted** و **Inserted** اضافه می کند.
۶. اگر نوع تریگری در هنگام ایجاد ذکر نشود، **AFTER** در نظر گرفته می شود (البته ذکر کلمه **FOR** الزامی است).
۷. یک جدول حداکثر می تواند یک تریگر از نوع **INSTEAD OF** حساس به هر یک از اعمال تغییر دهنده از قبیل **DELETE**، **UPDATE** و **INSERT** داشته باشد. پس هر جدول می تواند حداکثر سه تریگر **INSTEAD OF** داشته باشد.

۸. تعداد تریگرهای نوع AFTER برای هر جدول محدود نمی‌باشد. پس می‌توان چند تریگر برای عمل حذف بر روی یک جدول داشت. بنابراین باید بتوان اولویت اجرای تریگرها را تعیین کرد. برای این منظور، می‌توان اولین و آخرین تریگری که باید اجرا شوند را تعیین نمود، ولی بقیه تریگرها به صورت تصادفی اجرا می‌شوند. در ادامه چگونگی انجام این عمل را می‌بینید.

۹. اگر برای جدولی هم تریگر INSTEAD OF و هم تریگر AFTER حساس به یک تغییر دهنده داشته باشید، تریگر INSTEAD OF اجرا خواهد شد (تریگر AFTER اجرا نمی‌شود).

مثال ۲۳-۴. پرس‌وجویی که تریگری به نام trAddGroupBook بر روی جدول GroupBook ایجاد می‌کند تا با هر بار اجرای دستور INSERT تعداد رکوردهای اضافه شده را نمایش دهد.

```
CREATE TRIGGER trAddGroupBook
ON GroupBook FOR INSERT
AS raiserror('%d rows have been Inserted', ۱۱, ۱, @@rowcount)
RETURN
```

در این دستور @@rowcount، متغیری است که تعداد سطری را برمی‌گرداند که تحت تأثیر آخرین دستور قرار گرفته‌اند.

مثال ۲۴-۴. پرس‌وجویی که یک رکورد به جدول GroupBook اضافه کرده و باعث فعال شدن تریگر trAddGroupBook می‌شود.

```
INSERT INTO GroupBook VALUES ('۰۷', 'علوم کامپیوتر')
```

با اجرای این دستور خروجی زیر ظاهر می‌شود:

```
Msg ۵۰۰۰۰, Level ۱۱, State ۱, Procedure trAddGroupBook, Line ۴
  ۱ rows have been Inserted
(۱ row(s) affected)
```

این خروجی نشان می‌دهد که یک رکورد به جدول GroupBook اضافه شده است (دو سطر اول خروجی توسط تریگر نمایش داده شده است).

مثال ۲۵-۴. پرس‌وجویی که تریگری به نام trDelGroupBook ایجاد می‌کند تا تعداد رکوردهای حذف شده جدول GroupBook را بازبایی کند.

```
CREATE TRIGGER trDelGroupBook
ON GroupBook FOR DELETE
AS raiserror ('%d rows have been Deleted', ۱۱, ۱, @@rowcount)
RETURN
```

مثال ۲۶-۴. پرس‌وجویی که تریگر trDelGroupBook را آزمایش می‌کند.

```
DELETE FROM GroupBook WHERE groupName = 'علوم کامپیوتر'
```

با اجرای این دستور خروجی زیر ظاهر می‌شود:

```
Msg ۵۰۰۰۰, Level ۱۱, State ۱, Procedure trDelGroupBook, Line ۴
  ۱ rows have been Deleted
(۱ row(s) affected)
```

مثال ۲۷-۴. پرس و جویی که رکوردی با کد گروه کتاب '۰۱' را می‌خواهد حذف کند، چون این کد گروه در جدول Books وجود دارد (groupID کلید خارجی جدول Books است)، آن رکورد را حذف نمی‌کند.

```
DELETE FROM GroupBook WHERE groupID = '۰۱'
```

با اجرای این دستور خروجی زیر ظاهر می‌شود:

```
Msg ۵۴۷, Level ۱۱, State ۰, Line ۱
The DELETE statement conflicted with the REFERENCE constraint
"FK_Books_groupID_۰۵۱۹C۱AF". The conflict occurred in
database "PublishDB", table "dbo.Books", column 'groupID'.
The statement has been terminated.
```

همان‌طور که در این خروجی می‌بینید، تریگر trDelGroupBook هیچ خروجی نشان نداده است. چون تریگر بعد از اعمال تغییرات فعال خواهد شد. از آنجایی که محدودیت کلید خارجی نقض شده است، پس تریگر اجرا نمی‌شود.

۴-۴. کرسرها

همان‌طور که بیان گردید، دستور SELECT برای بازیابی مجموعه‌ای از رکوردها به کار می‌رود. یعنی با دستور SELECT نمی‌توانید مکان‌نما را به رکوردهای خاصی از جدول انتقال دهید. به عنوان مثال، با دستور SELECT نمی‌توانید رکوردهای که شماره رکورد آن‌ها فرد یا زوج است را جمع کرده، نمایش دهید. برای این منظور می‌توانید از کرسر^{۱۵} استفاده کنید. با استفاده از کرسر می‌توانید مکان‌نما را به رکورد خاصی منتقل کرده، یا در بین رکوردها به سمت بالا و پایین حرکت کنید. کرسرها انواع مختلف دارند که عبارت‌اند از:

۱. **کرسر ایستا**، یک کپی فوری^۲ از داده‌های مشخص شده در بانک اطلاعاتی tempdb نگهداری می‌کند. در این نوع کرسرها فقط امکان تغییر کپی فوری وجود دارد. پس، این نوع کرسرها فقط خواندنی هستند. یعنی، فقط امکان پیمایش رکوردها را دارند.

۲. **کرسر پویا**، امکان تغییر بر روی جداول پایه را دارند. دستور SELECT را می‌توانید در این نوع کرسرها استفاده کنید. اما هنگامی بخواهید از گزینه ORDER BY در دستور SELECT استفاده کنید، باید براساس فیلد یا فیلدهای که جلوی ORDER BY قرار می‌گیرند، ایندکس ایجاد کنید.

۳. **کرسر مجموعه کلید**^۳، گاهی اوقات اتفاق می‌افتد ترکیب چند فیلد، منحصر به فرد (یکتا بودن) رکوردها را در جدول تضمین می‌کنند. کرسرهای مجموعه کلید، فیلدهایی را در بانک اطلاعاتی tempdb ذخیره خواهند کرد که یکتایی رکورد در جدول را تضمین می‌کنند. این کرسرها می‌توانند ویژگی‌های از قبیل فقط خواندنی (مانند کرسرهای ایستا)، دست کاری (مانند کرسرهای پویا) و پیمایش رکوردها را داشته باشند.

مراحل استفاده از کرسر

برای استفاده از کرسر مراحل زیر را انجام دهید:

^{۱۵}. Cursor.

^۲. Snapshot.

^۳. Keyset Cursor.

۱. معرفی کرسر، با دستور DECLARE ...CURSOR می‌توانید کرسر را معرفی کنید. در ادامه چگونگی معرفی کرسر را می‌آموزید.

۲. مقداردهی به کرسر، از چند مرحله دیگر تشکیل می‌شود که عبارت‌اند از: ۱. تعریف متغیرهایی از نوع کرسر ۲. مقدار دهی به متغیر کرسر ۳. باز نمودن کرسر ۴. استفاده از کرسر برای مدیریت برر کوردها ۵. بستن کرسر ۶. حذف کرسر

جدول ۵ - ۴ پارامترهای تعریف کرسر.	
نام پارامتر	هدف
cursor_name	نام کرسر را تعیین می‌کند.
LOCAL	برای ایجاد کرسرهای محلی به کار می‌رود.
GLOBAL	برای ایجاد کرسرهای سراسری به کار می‌رود.
FORWARD_ONLY	کرسرهایی را ایجاد می‌کند که فقط در آن‌ها امکان پیمایش رکوردهای بعدی وجود دارد.
SCROLL	کرسرهای ایجاد می‌کند که امکان پیمایش رکوردها در آن‌ها وجود دارد. در ادامه با پیمایش رکوردها آشنا خواهید شد.
FOR UPDATE OF	فیلدهایی را برای به روز رسانی تعیین می‌کند. در حالت عادی کلیه فیلدها را می‌توان ویرایش کرد، مگر این که کرسر READ ONLY باشد.
STATIC	کرسرهای نوع ایستا ایجاد می‌کند.
KEYSET	کرسرهای نوع مجموعه کلید ایجاد می‌نماید.
DYNAMIC	کرسرهای نوع پویا ایجاد می‌کند.
FAST_FORWARD	کرسرهایی ایجاد می‌کند که فقط خواندنی هستند و امکان پیمایش رکوردها را ندارد.
READ_ONLY	کرسری فقط خواندنی ایجاد می‌کند که نمی‌توان در آن دستورات دست کاری داده از قبیل INSERT، DELETE و UPDATE را استفاده نمود.
SCROLL_LOCK	در هنگام خواندن داده‌ها رکوردها را قفل می‌کند.
OPTIMISTIC	تضمین می‌کند به روز رسانی یا حذفی که از طریق کرسر انجام شد موفق نباشد.
TYPE WARNING	اگر کرسری از نوعی به نوع دیگر تبدیل شود، پیغام اخطار سرویس گیرنده ارسال می‌کند.
select_Statement	دستور ایجاد کرسر را تعیین می‌کند.

کرسرها مانند متغیرهای معمولی هستند. همان طوری که در ۵ مرحله استفاده از کرسرها دیدید، پس از این که کارتان با کرسر پایان یافت، باید آن را حذف کنید. اما، متغیرهای معمولی با خروج از حوزه خودشان به طور خودکار حذف می‌شوند.

در ادامه مراحل مقداردهی به کرسر را می‌بینید.

۱- ۴- ۴. معرفی کرسر

همان طور که بیان گردید، برای استفاده از کرسر ابتدا باید آن را معرفی نموده، سپس به آن مقدار تخصیص داد (مانند متغیرهای معمولی). دستور DECLARE CURSOR، برای معرفی کرسر به کار می‌رود. این دستور به صورت زیر استفاده می‌شود:

```
DECLARE cursor_name CURSOR [ LOCAL | GLOBAL ]
    [ FORWARD_ONLY | SCROLL ] [ STATIC | KEYSET |
    DYNAMIC | FAST_FORWARD ] [ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
    [ TYPE_WARNING ] FOR select_statement
    [ FOR UPDATE [ OF column_name [ , ... n ] ] ] [;]
```

شرح پارامترهای این دستور در جدول ۵-۴ آمده است.

مثال ۴۲-۴. پرس‌وجویی که کرسری به نام crsGroupBook ایجاد می‌کند که برای بازیابی اطلاعات جدول GroupBook استفاده می‌شود.

```
DECLARE crsGroupBook CURSOR
FOR SELECT * FROM GroupBook
```



۴-۴. مسائل حل شده

مثال ۱. پرس‌وجویی که لیست دیدهای بانک اطلاعاتی Buy_Sell را بازیابی می‌کند.

```
USE Buy_Sell
SELECT * FROM INFORMATION_SCHEMA.VIEWS
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION	CI
1	Buy_Sell	dbo	CityCustomerView	CREATE VIEW CityCustomerView AS SELECT * FROM C...	N
2	Buy_Sell	dbo	MerchandiseConsumerPrice	CREATE VIEW MerchandiseConsumerPrice AS SELECT ...	N
3	Buy_Sell	dbo	MerchandiseGHan	CREATE VIEW MerchandiseGHan AS SELECT * FRO...	N
4	Buy_Sell	dbo	CustomerInfoView	CREATE VIEW CustomerInfoView AS SELECT firstName...	N
5	Buy_Sell	dbo	BuyInCountView	CREATE VIEW BuyInCountView AS SELECT invoiceNu...	N
6	Buy_Sell	dbo	CustomerNullView	CREATE VIEW CustomerNullView AS SELECT firstName...	N

مثال ۲. پرس‌وجویی که لیست توابع بانک اطلاعاتی Buy_Sell را بازیابی می‌کند.

```
USE Buy_Sell
SELECT * FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE = 'FUNCTION'
```

	SPECIFIC_CATALOG	SPECIFIC_SCHEMA	SPECIFIC_NAME	ROUTINE_CATALOG	ROUTINE_SCHEMA	ROUTINE_NAME
1	Buy_Sell	dbo	CountMerchandise	Buy_Sell	dbo	CountMerchandise
2	Buy_Sell	dbo	GetPrice	Buy_Sell	dbo	GetPrice
3	Buy_Sell	dbo	fn_diagramobjects	Buy_Sell	dbo	fn_diagramobjects

مثال ۳. پرس‌وجویی که لیست ستون‌های جدول City بانک اطلاعاتی Buy_Sell را بازیابی می‌کند.

```
USE Buy_Sell
SELECT * FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'City'
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE
1	Buy_Sell	dbo	City	cityID	1	NULL	NO
2	Buy_Sell	dbo	City	cityName	2	NULL	NO
3	Buy_Sell	dbo	City	userCode	3	NULL	YES

مثال ۴. پرس‌وجویی که لیست جداول بانک اطلاعاتی Buy_Sell را بازیابی می‌کند.

```
USE Buy_Sell
SELECT * FROM sys.tables
```

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	
1	City	21575115	NULL	1	0	U	USER_TABLE	2012-06-28 18:04:14.003	2
2	State	101575400	NULL	1	0	U	USER_TABLE	2012-06-28 18:04:31.273	2
3	Customer	181575685	NULL	1	0	U	USER_TABLE	2012-06-28 18:04:52.527	2
4	Store	293576084	NULL	1	0	U	USER_TABLE	2012-06-28 18:05:53.013	2
5	merchandiseGroup	373576369	NULL	1	0	U	USER_TABLE	2012-06-28 18:06:29.843	2
6	Merchandise	453576654	NULL	1	0	U	USER_TABLE	2012-06-28 18:07:00.090	2
7	City	515756000	NULL	1	0	U	USER_TABLE	2012-06-28 18:07:24.283	2

مثال ۵. پرس وجویی که لیست ستونهای جدول Sellers در بانک اطلاعاتی Buy_Sell را بازیابی می کند.

```
USE Buy_Sell
SELECT * FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'Seller'
```

	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE
1	Buy_Sell	dbo	Seller	sellerID	1	NULL	NO
2	Buy_Sell	dbo	Seller	sellerName	2	NULL	NO
3	Buy_Sell	dbo	Seller	userCode	3	NULL	YES

۴-۵. دستور کار آزمایشگاه

۱. پرس وجویی بنویسید که لیست فایل های بانک اطلاعاتی TransportationAgency را بازیابی کند.
۲. پرس وجویی بنویسید که لیست جداول بانک اطلاعاتی TransportationAgency را بازیابی کند.
۳. پرس وجویی بنویسید که لیست دیدهای بانک اطلاعاتی TransportationAgency را بازیابی کند.
۴. پرس وجویی بنویسید که لیست تمام پارامترهای بانک اطلاعاتی TransportationAgency را بازیابی کند.
۵. پرس وجویی بنویسید که لیست پارامترهای ورودی بانک اطلاعاتی TransportationAgency را بازیابی کند.
۶. پرس وجویی بنویسید که لیست توابع بانک اطلاعاتی TransportationAgency را بازیابی کند.
۷. پرس وجویی بنویسید که لیست رویه های ذخیره شده بانک اطلاعاتی TransportationAgency را بازیابی کند.
۸. پرس وجویی بنویسید که لیست بانک اطلاعاتی موجود در سرویس دهنده فعلی را بازیابی کند.
۹. پرس وجویی بنویسید که لیست ستونهای جدول Cars مربوط به بانک اطلاعاتی TransportationAgency را بازیابی کند.
۱۰. پرس وجویی بنویسید که لیست محدودیت های بانک اطلاعاتی TransportationAgency را بازیابی کند.
۱۱. پرس وجویی بنویسید که لیست گروه فایل های بانک اطلاعاتی TransportationAgency را بازیابی کند.

۱۲. پرس وجویی بنویسید که لیست کلیدهای خارجی بانک اطلاعاتی TransportationAgency را بازیابی کند.

۱۳. پرس وجویی بنویسید که لیست کلیدهای اولیه بانک اطلاعاتی TransportationAgency را بازیابی کند.

۱۴. پرس وجویی بنویسید که لیست کاربران بانک اطلاعاتی TransportationAgency را بازیابی کند.

۱۵. پرس وجویی بنویسید که از بانک اطلاعاتی TransportationAgency پشتیبان تهیه کرده، برروی پشتیبان کلمه عبور mail_fan را قرار دهد.

۱۶. پرس وجویی بنویسید که پشتیبان تهیه شده بانک اطلاعاتی TransportationAgency که برروی آن کلمه عبور mail_fan قرار دارد را بازیابی کند (البته فایل های داده و کارنامه آن را به مکان جدید منتقل نماید).

۱۷. پرس وجویی بنویسید که پشتیبان تهیه شده بانک اطلاعاتی TransportationAgency که برروی آن کلمه عبور mail_fan قرار دارد را با کلمه عبور mail_fan می خواهد بازیابی کند، دستور را نوشته، پیغام خطا را مشاهده کنید.

۱۸. پرس وجویی بنویسید که بانک اطلاعاتی بازیابی شده TransportationAgency را حذف کند.

۱۹. پرس وجویی بنویسید که از بانک اطلاعاتی TransportationAgency پشتیبان کامل تهیه کند.

۲۰. پرس وجویی بنویسید که پشتیبان TransportationAgency را به نام T۱ با فایل های داده t۱_data و کارنامه t۱_log بازیابی کند.

۲۱. پرس وجویی بنویسید تا نام تمام بانک های اطلاعاتی موجود برروی رایانه تان را بازیابی کند.

۲۲. پرس وجویی بنویسید تا تمام جدول بانک اطلاعاتی TransportationAgency را بازیابی کند.

۲۳. پرس وجویی بنویسید تا نام تمام دیدهای بانک اطلاعاتی TransportationAgency را بازیابی کند.

۲۴. پرس وجویی بنویسید که تریگری به نام trInsertOrders ایجاد کند تا تعداد رکوردهای اضافه شده به جدول Orders را با هر بار اجرای دستور INSERT نمایش دهد.

۲۵. پرس وجویی بنویسید تا تریگر trInsertOrders را آزمایش کند.

۲۶. پرس وجویی بنویسید تا تریگر trInsertOrders را طوری تغییر دهد که از ورود عدد منفی برای فیلد Freight جلوگیری کند.

۲۷. پرس وجویی بنویسید تا تریگری به نام trDeleteDrivers را ایجاد کند. این تریگر اگر رکوردی از جدول رانندگان (Drivers) می خواهد حذف شود که کد راننده در جدول Orders وجود دارد، از حذف آن راننده جلوگیری می کند.

۲۸. پرس وجویی بنویسید تا تریگر trDeleteDrivers را آزمایش کند.

۲۹. پرس وجویی بنویسید تا تریگری به نام trModifyDrivers را ایجاد کند. این تریگر در هنگام اجرای دستور INSERT و UPDATE چنانچه کد راننده در جدول Drivers موجود باشد، جلوی تغییر و اضافه شدن رکورد را بگیرد.

۳۰. پرس وجویی بنویسید تا تریگر trModifyDrivers را آزمایش کند.

۳۱. پرس وجویی بنویسید تا تریگرهای trDeleteDrivers، trModifyDrivers و trInsertOrders را حذف کند.

۳۲. پرس وجویی بنویسید تا کرسری ایجاد کند که نام و نام خانوادگی جدول رانندگان (Derivers) را بازیابی کند.

۳۳. پرس وجویی بنویسید تا با استفاده از کرسری نام و نام خانوادگی اولین و آخرین رکورد جدول رانندگان (Derivers) را بازیابی کند.

۳۴. پرس وجویی بنویسید تا کرسری ایجاد کند که شماره رکورد و نام ماشین ها را نمایش دهد.

۳۵. پرس وجویی بنویسید که کرسری ایجاد کند تا آن کرسر نام و نام خانوادگی مشتریان بابلی را نمایش دهد.

۳۶. پرس وجویی بنویسید که کرسری ایجاد کند تا این کرسر مجموع کرایه های راننده ای با نام خانوادگی امیدی را بازیابی کند.

۶-۴. پاسخ دستور کار آزمایشگاه

۷-۴. تمرین ها

۱. پرس وجویی بنویسید تا لیست کاربران بانک اطلاعاتی حسابداری را بازیابی کند.
۲. پرس وجویی بنویسید تا لیست جداول بانک اطلاعاتی حسابداری را بازیابی کند.
۳. پرس وجویی بنویسید تا لیست دیدهای بانک اطلاعاتی حسابداری را بازیابی کند.
۴. پرس وجویی بنویسید تا لیست ستون های جدول کل از سیستم حسابداری را بازیابی کند.
۵. پرس وجویی بنویسید تا لیست کلیدهای خارجی جدول تفضیلی از سیستم حسابداری را بازیابی کند.
۶. پرس وجویی بنویسید تا لیست کلیدهای اولیه جدول تفضیلی از سیستم حسابداری را بازیابی کند.
۷. پرس وجویی بنویسید تا لیست گروه فایل های بانک اطلاعاتی حسابداری را بازیابی کند.
۸. پرس وجویی بنویسید تا لیست بانک های موجود در سرویس دهنده فعلی را بازیابی کند.
۹. پرس وجویی بنویسید تا لیست پارامترهای ورودی سیستم حسابداری را بازیابی کند.
۱۰. پرس وجویی بنویسید تا لیست توابع سیستم حسابداری را بازیابی کند.
۱۱. پرس وجویی بنویسید تا لیست رویه های ذخیره شده سیستم حسابداری را بازیابی کند.
۱۲. پرس وجویی بنویسید تا لیست پارامترهای خروجی بانک اطلاعاتی حسابداری را بازیابی کند.
۱۳. پرس وجویی بنویسید تا لیست فیلدهای جدول تفضیلی را بازیابی کند.

۱۴. پرس وجویی بنویسید تا انواع تعریف شده SQL را بازیابی کند.
۱۵. پرس وجویی بنویسید تا از بانک اطلاعاتی حسابداری پشتیبانی کامل گرفته، این پشتیبان را در مسیر C:\Backup رایانه تان قرار دهد.
۱۶. پرس وجویی بنویسید تا از تغییرات بانک اطلاعاتی حسابداری پشتیبانی گرفته، این پشتیبان را در مسیر C:\Backup با پسوند bck ذخیره کند.
۱۷. پرس وجویی بنویسید تا از بانک اطلاعاتی حسابداری پشتیبان تهیه کند. در هنگام تهیه پشتیبان فایل ها را فشرده نموده و بر روی آن Password قرار دهد.
۱۸. پرس وجویی بنویسید تا پشتیبان تهیه شده در سوال ۱۵ را در بانک اطلاعاتی به نام Account۱ بازیابی کند.
۱۹. پرس وجویی بنویسید تا پشتیبان تهیه شده در سوال ۱۷ را به نام بانک اطلاعاتی Account۲ بازیابی کند.
۲۰. پرس وجویی بنویسید که از بانک اطلاعاتی حسابداری پشتیبان کامل تهیه کند.
۲۱. پرس وجویی بنویسید که از تغییرات بانک اطلاعاتی حسابداری پشتیبان تهیه کند.
۲۲. پرس وجویی بنویسید که پشتیبان تهیه شده از بانک اطلاعاتی حسابداری را بازیابی کند.
۲۳. پرس وجویی بنویسید که از بانک اطلاعاتی حسابداری پشتیبان تهیه کرده، به طوری که بر روی آن کلمه عبور ۱۰۹۸۷۶۵۴۳۲۱ را قرار دهد.
۲۴. پرس وجویی بنویسید که پشتیبان دارای کلمه عبور بانک اطلاعاتی حسابداری را بازیابی کند، به طوری که فایل های داده و کارنامه (log) آن را به پوشه جدیدی انتقال دهد.
۲۵. پرس وجویی بنویسید که تریگری بر روی جدول کل ایجاد کند تا در هنگام حذف رکورد فعال شود و تعداد رکوردهای حذف شده را نمایش دهد.
۲۶. پرس وجویی بنویسید که تریگری برای جدول ریز سند ایجاد کند تا در هنگام افزودن و تغییر رکورد فعال شود و تعداد رکوردهای تاثیر یافته با دستور INSERT و UPDATE را نمایش دهد.
۲۷. پرس وجویی بنویسید تا دو تریگر بیان شده سوالات ۲۵ و ۲۶ را آزمایش کند.
۲۸. پرس وجویی بنویسید که تریگری ایجاد کند تا در هنگام حذف رکوردی از جدول معین، بررسی نماید کد معین در جداول ریز سند و تفصیلی وجود ندارد. چنانچه کد معین در این جداول وجود داشت، از حذف رکورد در جدول معین جلوگیری کرده، پیغام مناسب نمایش دهد.
۲۹. پرس وجویی بنویسید که تریگری ایجاد کند تا در هنگام ورود و ویرایش جدول ریز سند، چنانچه در مبالغ بدهکار و بستانکار منفی وارد گردید، جلوی افزودن و ویرایش رکورد را بگیرد.
۳۰. پرس وجویی بنویسید تا تریگر ایجاد شده سوال ۲۹ را آزمایش کند.
۳۱. پرس وجویی بنویسید که تریگری ایجاد کند تا در هنگام افزودن رکورد در جدول معین، چنانچه کد کل در جدول کل وجود نداشت از افزودن رکورد جلوگیری کند.
۳۲. پرس وجویی بنویسید تا تریگر ایجاد شده سوال ۳۱ را آزمایش کند.

۳۳. پرس وجویی بنویسید که تریگری ایجاد کند تا در هنگام افزودن رکورد به جدول کل، چنانچه کد کل در جدول وجود داشت، از افزودن رکورد جلوگیری کند.
۳۴. پرس وجویی بنویسید تا تریگر ایجاد شده در سوال ۳۳ را طوری تغییر دهد تا چنانچه نام و کد کل در جدول کل وجود داشت، از درج آن در جدول کل جلوگیری نماید.
۳۵. پرس وجویی بنویسید که چند تریگر موجود را حذف کند.
۳۶. پرس وجویی بنویسید که تریگری ایجاد کند تا در هنگام درج رکورد در جدول ریز سند، رکوردی با مبلغ بدهکار یا بستانکار یک (۱) ریال اضافه نکند.
۳۷. پرس وجویی بنویسید که تریگر سوال ۳۶ را طوری تغییر دهد تا در هنگام درج در جدول ریز سند، رکوردی با مبلغ ۲۰ ریال اضافه نکند.
۳۸. پرس وجویی بنویسید تا تریگر ایجاد شده در سوالات ۳۳ و ۳۶ را حذف کند.
۳۹. پرس وجویی بنویسید که تریگری ایجاد کرده تا در هنگام درج، ویرایش و حذف رکورد از جدول تفضیلی، تعیین کند کدام یک از اعمال درج، ویرایش و حذف انجام شده است.
۴۰. پرس وجویی بنویسید تا با اجرای دستورات INSERT، UPDATE و DELETE بر روری جدول تفضیلی تریگر ایجاد شده سوال ۳۹ را آزمایش نماید.
۴۱. پرس وجویی بنویسید که کرسری ایجاد کند تا رکوردهای زوج جدول ریز سند را بازیابی کند.
۴۲. پرس وجویی بنویسید که کرسری ایجاد کند تا جمع مبلغ بدهکار و بستانکار جدول ریز سند را بازیابی کند.
۴۳. پرس وجویی بنویسید که کرسری ایجاد نماید تا اطلاعات رکورد شماره ۱۰ و ۲۰ جدول کل را بازیابی کند.
۴۴. پرس وجویی بنویسید که کرسری ایجاد نماید تا اطلاعات اولین و آخرین رکورد جدول سند را بازیابی کند.
۴۵. پرس وجویی بنویسید که کرسری ایجاد نماید تا اطلاعات کد و نام معین جدول معین را رکورد به رکورد بازیابی نماید.
۴۶. پرس وجویی بنویسید که کرسری ایجاد کند تا مانده جمع بدهکار و بستانکار سند شماره ۱۵ و مانده کل آن را نمایش دهد.

منابع

۱. عباس نژادورزی، رمضان، آموزش گام به گام بانک اطلاعاتی با C# (مرجع کامل)، بابل انتشارات فن-آوری نوین، ۱۳۸۸.
۲. جعفری، محمدرضا، "ایجاد بانک اطلاعاتی در ۲۰۰۵ SQL Server" مشهد، واژگان خرد، ۱۳۸۷.
۳. Abraham Silberschatz, Henry Korth, and S. Sudarshan, "Database System Concepts (۶th Edition)", MC Graw Hill, ۲۰۱۰.
۴. Ryan Stephens, Ron D. Plew, and Arie D. Jones "Teach Yourself SQL in One Hour a Day (۵th Edition)", Sams, ۲۰۰۹.

۵. Ken Simmons and Syivester Carstarphen, "pro SQL Server ۲۰۱۲ Administration", Apress, ۲۰۱۲.
۶. Itzik Ben-Gen, " Microsoft SQL Server ۲۰۱۲ High-Performancr T-SQL using Window Functions, O Reilly, ۲۰۱۲
۷. Paul Turley, Robert Bruckner, Thiago Silva, ken Withee, Grant Paisley, "Microsoft SQL Server ۲۰۱۲" Reporting Services", John Wiley & Sons, ۲۰۱۲.

این کتاب شامل ۱۷۶ صفحه است که فایل الکترونیکی آن را می‌توانید از سایت کتابراه دانلود نمایید

<http://ktbr.ir/b28503>