آزمایشگاه پایگاه داده
با
SQL Server 2012

برای رشته های مهندسی کامپیوتر
IT و علوم کامپیوتر

مولفین:
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 رویه‌های ذخیره شده و توابع
بازیابی اطلاعات از کاتالوگ
پشتیبان‌گیری 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 عباس‌نژادورزی
مهندس فاطمه عبدی سقاواز
مهندس بهارک شاکری اسکی

فن‌آوری نوین

سرشناسه	:	عباس‌نژاد ورزی، رمضان، ۱۳۴۸
عنوان و نام پدیدآور	:	آزمایشگاه پایگاه داده با SQL Server ۲۰۱۲ / تالیف رمضان عباس‌نژادورزی، فاطمه عبدی سقاواز، بهارک شاکری اسکی
مشخصات نشر	:	مازندران: فن‌آوری نوین، ۱۳۹۲.
مشخصات ظاهری	:	۱۷۶. ص: مصور،
شابک	:	۹۷۸-۶۰۰-۹۲۲۵۴-۶-۰
وضعیت فهرست‌نویسی	:	فیپا
موضوع	:	سرور اس. کیو. ال
موضوع		پایگاه‌های اطلاعاتی -- طراحی
شماره افزوده		عبدی‌سقاواز، فاطمه، ۱۳۵۶ -
شماره افزوده		شاکری اسکی، بهارک، ۱۳۶۲-
رده بندی کنگره	:	QA۷۶/۹/س۴ع۲۱۳۹۲
رده بندی دیویی	:	۰۰۵/۷۵۸۵
شماره کتابشناسی ملی	:	۳۰۶۷۸۳۶

فن‌آوری نوین
www.fanavarienovin.net
بابل، صندوق پستی ۷۳٤٤۸-٤۷۱٦۷ تلفن: ۲۲۵٦٦۸۷-۰۱۱۱

آزمایشگاه پایگاه داده با SQL Server ۲۰۱۲

تالیف: مهندس رمضان عباس‌نژادورزی- مهندس فاطمه عبدی سقاواز- بهارک شاکری اسکی

ناشر: فن‌آوری نوین

چاپ اول: بهار ۱۳۹۲

جلد: ۱۰۰۰

شابک: ۹۷۸ - ۶۰۰ - ۹۲۲۵۴ - ۶ - ۰

حروفچینی و صفحه‌آرایی: فن‌آوری نوین

قیمت: ۸۸۰۰ تومان

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۱۴۴-۶۶۴۰۰۲۲۰

فهرست مطالب

فصل اول: مراحل طراحی بانک‌اطلاعات ۷

۱- ۱. مراحل طراحی بانک اطلاعاتی ۷

۱ – ۱- ۱. تعیین کاربرد اصلی بانک اطلاعاتی ۷

۲ – ۱ – ۱. تعیین جداول مورد نیاز بانک اطلاعاتی ۸

۳-۱-۱. تعیین فیلدهای مورد نیاز بانک اطلاعاتی ۱۰

۴-۱-۱. تعریف رابطه‌های بین جداول ۱۱

۵ – ۱ – ۱. بهینه سازی طراحی ۱۲

۲- ۱. بانک اطلاعاتی SQL Server ۱۳

۱-۲ - ۱. ورود به بانک اطلاعاتی SQL Server ۱۳

۲ - ۲-۱. تایپ و اجرای دستورات SQL .. ۱۴

۳-۱. ایجاد بانک اطلاعاتی ۱۴

۱ – ۳ -۱. گروه‌های فایل ۱۷

۲-۳-۱. تغییر خواص بانک اطلاعاتی موجود ۱۷

۳-۳-۱. حذف بانک اطلاعاتی موجود ۱۸

۴-۱. ایجاد و تغییر ساختار جدول ۲۱

۱ – ۴ – ۱. ایجاد جدول ۲۴

۲-۴-۱. تغییر ساختار جدول با دستور SQL ۲۷

۳-۴-۱. حذف جدول با دستور SQL ۲۸

۵ – ۱. مسائل حل شده ۲۸

۶ – ۱. دستور کار آزمایشگاه ۳۸

۷ – ۱. جواب دستور کار آزمایشگاه ۴۰

۸ – ۱. تمرین‌ها ۴۳

فصل دوم: ورود، ویرایش، حذف و بازیابی اطلاعات ۴۵

۱-۲. دستور INSERT ۴۶

۲-۲. ویرایش رکوردهای جدول ۴۸

۳-۲. حذف رکوردهای جدول ۴۹

۴-۲. پرس‌وجوی بازیابی اطلاعات ۵۰

۱-۴-۲. عملگرها در SQL ۵۰

۲-۴ – ۲. بازیابی اطلاعات از جدول با دستور SELECT ۵۴

۵ – ۲. مرتب سازی رکوردها ۵۷

۶– ۲. پرس‌وجوهای مرکب ۵۸

۱ – ۶ – ۲. عملگر UNION ۵۸

۲ – ۶ – ۲. عملگر UNION ALL ۵۸

۳ – ۶- ۲. عملگر INTERSECT ۵۹

۴ – ۶ – ۲. عملگر EXCEPT ۵۹

۷ – ۲. توابع تجمعی ۶۰

۸ – ۲. گروه‌بندی اطلاعات ۶۰

۹- ۲. تست مقدار تهی فیلد ۶۲

۱۰ – ۲. پرس‌وجوی فرعی ۶۲

۱۱ – ۲. پیوند جداول (رابطه) ۶۵

۱ – ۱۱ – ۲. پیوند ضربدری ۶۵

۲ – ۱۱ – ۲. پیوند متعادل ۶۶

۳ – ۱۱ – ۲. پیوند نامتعادل ۶۷

۴ – ۱۱ – ۲. پیوند درونی ۶۷

۵ – ۱۱ – ۲. پیوند بیرونی ۶۸

۱۲ – ۲. افزودن مقادیر از یک جدول به جدول دیگر ۷۰

۱۳ – ۲. حذف رکوردهای یک جدول از طریق جداول دیگر ۷۱

۱۴ – ۲. مسائل حل شده ۷۲

۱۵-۲. دستور کار آزمایشگاه ۸۳

۱۶ – ۲. پاسخ دستور کار آزمایشگاه..... ۸۴
۱۷ – ۲. تمرین‌ها..... ۸۶

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

۱-۳. دید..... ۹۰
۱-۱-۳. ایجاد دید..... ۹۰
۲-۱-۳. تغییر دید..... ۹۲
۳-۱-۳. حذف دید..... ۹۳
۲ – ۳. برنامه‌نویسی در SQL..... ۹۳
۱ -۲ – ۳. متغیرها..... ۹۳
۲ – ۲-۳. توضیحات..... ۹۴
۳ – ۲ – ۳. دستور RETURN..... ۹۴
۴ – ۲ – ۳. دستور PRINT..... ۹۴
۵ – ۲ – ۳. دستور RAISERROR..... ۹۴
۶ – ۲ -۳. دستور WAITFOR..... ۹۵
۷-۲-۳. ساختارهای تصمیم..... ۹۶
۳-۳. رویه‌های ذخیره شده..... ۹۸
۱-۳-۳. ایجاد رویه‌های ذخیره شده..... ۹۹
۲ – ۳ – ۳. انواع رویه‌های ذخیره شده..... ۱۰۰
۳-۳-۳. اجرای رویه ذخیره شده..... ۱۰۶
۴-۳-۳. تغییر رویه ذخیره شده..... ۱۱۰
۴ - ۳. توابع تعریف شده توسط کاربر..... ۱۱۱
۱ – ۴ – ۳. انواع توابع تعریف شده توسط کاربر ۱۱۲
۲- ۴ -۳. فراخوانی توابعی که یک مقدار اسکالر را برمی‌گردانند..... ۱۱۴
۳ – ۴ – ۳. توابعی که یک جدول را برمی‌گردانند ۱۱۶
۵ – ۴ – ۳. فراخوانی توابع خطی که یک جدول 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

۱-۴. بازیابی اطلاعات از کاتالوگ..... ۱۳۸
۲-۴. پشتیبان‌گیری از اطلاعات ۱۴۱
۱ – ۲ - ۴. پشتیبان گیری با دستور BACKUP 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

منابع:.....۱۷۶

نام کتاب	لینک خرید الکترونیکی
مبانی رایانه و برنامه‌نویسی به زبان C++	http://ktbr.ir/b۳۰۵۸۸
آشنایی با مبانی امنیت شبکه (امنیت اطلاعات)	http://ktbr.ir/b۳۰۳۲۷
اصول طراحی پایگاه داده‌ها	http://ktbr.ir/b۲۹۹٤۳
آموزش گام‌به‌گام برنامه‌نویسی پایتون	http://ktbr.ir/b۲۹۹۸٤
حل مسائل C++ (آزمایشگاه کامپیوتر مرجع کامل)	http://ktbr.ir/b۲۹۹۸۲
آموزش گام‌به‌گام LINQ با C#	http://ktbr.ir/b۲۸٤٥۱
ساختمان داده‌ها با C++	http://ktbr.ir/b۲۹٦۷٦
طراحی سیستم‌های شی‌گرا با زبان C #	http://ktbr.ir/b۲۹٦۲۱
مدیریت استراتژیک فناوری اطلاعات	http://ktbr.ir/b۲۹۷۷۹
گرافیک رایانه‌ای با زبان برنامه‌نویسی C#	http://ktbr.ir/b۲۹٦۷٤
درس و کنکور پایگاه داده پیشرفته	http://ktbr.ir/b۲۹٦٤٤
فیزیک الکتریسیته	http://ktbr.ir/b۲۹٦۸۰
تجارت الکترونیکی	http://ktbr.ir/b۲۹٦۱۹
راهنمای کاربردی کاربری OPNET برای شبکه‌های شبیه‌سازی کامپیوتر	http://ktbr.ir/b۲۸٥۰٤
درس و کنکور سیستم عامل پیشرفته	http://ktbr.ir/b۲۸٥۰٥
شبکه‌های کامپیوتری با رویکرد کاربردی، آزمایشگاه شبیه‌سازی شبکه	http://ktbr.ir/b۲۸٥۲۸
آزمایشگاه پایگاه داده با SQL Server ۲۰۱۲	http://ktbr.ir/b۲۸٥۰۳

کاربرد رایانه در مدیریت و حسابداری	http://ktbr.ir/b۲۸٤٥۰
آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با ویژوال بیسیک‌نت	http://ktbr.ir/b۲۸٤٤۹
آموزش گام‌به‌گام برنامه‌نویسی به زبان C++	http://ktbr.ir/b۲۸٤٥۲
دانلود کتاب آموزش گام‌به‌گام برنامه‌نویسی بانک اطلاعاتی با C#	http://ktbr.ir/b۲۸٤٤۸
حل مسائل پاسکال	http://ktbr.ir/b۲۸۳۹۸
حل مسائل C++	http://ktbr.ir/b۲۸٤۰۱
دانلود کتاب حل مسائل C#	http://ktbr.ir/b۲۸۳۹۹
دانلود کتاب حل مسائل C	http://ktbr.ir/b۲۸۳۹۷

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 ابنک اطلاعاتی یک انتشارات بیان گردید).

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

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

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

از جمله ویژگی بسیار مهم کتاب حاضر، پیوستگی مطالب، مثال‌ها و اجرا مسائل آن برروی 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 کرسرها و چگونگی پیاده‌سازی آن‌ها در SQL Server بیان گردیده است. کتابی که در پیش رو دارید، با بهرگیری 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 پست‌الکترونیک **www.fanavarienovin@Gmail.com** ارسال نمایند.

فصل ۱ مراحل طراحی بانک اطلاعاتی

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 اطلاعاتی‌تان چه هستند) و داده‌هایی که در جداول قرار می‌گیرند، چه نوع‌اند. فیلدهای تشکیل دهنده جداول‌تان 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؟

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

۵. آیا فروش به صورت نسیه 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 جداولی 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[1]. 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 مو لف، نام مو لف، نام خانوادگی، سن و غیره را نگهداری می‌کند.
- **جدول AutBook**، اطلاعات شابک کتاب، مو لف و مبلغ حق تا لیف را نگهداری می‌کند.
- **جدول PubBook**، اطلاعات شابک، کد ناشر، تاریخ نشر و حق نشر را نگهداری می‌کند.

---

[1] . نرمال سازی جدول، یعنی شکستن یک جدول به چند جدول دیگر. مباحث مربوط به نرمال‌سازی را می‌توانید در کتاب اصول طراحی پایگاه داده از انتشارات فن‌آوری 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(٦)	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 (مو لفان)	atID	کد مو لف	Varchar(۱۰)	بله	نه	نه
	aFname	نام مو لف	Varchar(۲۰)	نه	نه	نه
	aLname	نام خانوادگی مو لف	Varchar(۲۰)	نه	نه	نه
	Age	سن مو لف	int	نه	نه	نه
	Ranking	مدرک مو لف	Varchar(۳۰)	نه	بله	نه
	Email	پست الکترونیک مو لف	Varchar(۵۰)	نه	بله	نه
	Mobile	موبایل مو لف	Varchar(۱۵)	نه	بله	نه
	SumPayment	مجموع حق تا لیف	Decimal(۱۸, ۰)	نه	بله	نه
AutBook (کتاب‌های تالیف شده)	ISBN	کد کتاب	Varchar(۱۰)	بله	نه	بله
	atID	کد مو لف	Varchar(۱۰)	بله	نه	بله
	Payment	حق تا لیف	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(٦)	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 داده-هایی 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.

٤. **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)

٦. فیلدهای کلید اولیه[۳] 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[۲]) مشخص می‌کند. 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 نمی‌توانید 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 آمده‌اند:

- **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 رابطه‌های بین جداول

همان‌طور که بیان گردید، یک جدول را به چند جدول کوچک‌تر می‌شکنیم تا افزونگی، بی‌نظمی و داده‌های NULL کاهش یابد. به عنوان مثال، اطلاعات مشتریان و فروش‌های انجام شده در دو جدول مجزا و مستقل ذخیره می‌شوند. اکنون، اگر پرس‌وجویی 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 می‌توان بین جداول تعریف کرد:

◈ رابطه یک به چند ◈ رابطه چند به چند ◈ رابطه یک به یک

◈ **رابطه یک به چند**، متداول‌ترین نوع رابطه در یک بانک اطلاعاتی رابطه‌ای است. در یک رابطه یک به چند، یک رکورد (سطر) در یک جدول (نظیر A) می‌تواند 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) می‌تواند بیش از یک رکورد متناظر در جدول دیگر (نظیر B) باشد و بالعکس.

---

۳. Primary Key ۲. Unique ۳. Foreign Key

◈ **رابطه یک به یک**، هر رکورد جدول A نمی‌تواند بیش از یک رکورد متناظر در جدول B داشته باشد و نیز هر رکورد جدول B نمی‌تواند 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 می‌کند. 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 می‌تواند چند مؤلف داشته باشد و چند کتاب می‌تواند توسط یک مؤلف تألیف شود.
- بین جداول Books و GroupBook، فیلد groupID ارتباط را برقرار می‌کند. در این ارتباط، فیلد groupID در جدول GroupBook کلید اولیه است، ولی این فیلد در جدول Books کلید خارجی می‌باشد. ارتباط بین این دو جدول یک به چند است. زیرا، چند کتاب می‌توانند به یک گروه اختصاص یابند.
- فیلد pubID برای برقرار ارتباط بین جداول Publishers و PubBook به کار می‌رود. 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 می‌تواند چند کتاب را انتشار دهد و چند ناشر نیز می‌توانند یک کتاب را با همکاری هم چاپ کنند.

۱ – ۳ –۱. گروه‌های فایل

در بانک اطلاعاتی می‌توان گروه‌هایی برای فایل‌ها تعریف نمود تا با قرار دادن فایل‌های داده و ایندکس‌ها روی دیسک‌های مختلف کارائی سیستم را افزایش داد. دو نوع گروه فایل در SQL Server وجود دارد که عبارت‌اند از:

۱. **گروه فایل اصلی**[۴]، حاوی فایل‌های داده اصلی[۲] و فایل‌هایی است که گروه فایل آن‌ها مشخص نگردید. همه صفحات[۳] اختصاص داده شده به جداول سیستم در این گروه قرار می‌گیرند. هر بانک اطلاعاتی حداقل یک گروه فایل اصلی دارد.

۲. **گروه فایل‌های تعریف شده توسط کاربر**[۴]، با پارامتر FILEGROUP از طریق دستورهای CREATE DATABASE و ALTER DATABASE تعریف می‌شوند. عملکرد این دستورات را در ادامه می‌بینید.

در هنگام ایجاد گروه‌های فایل به نکات زیر توجه کنید:

۱. هر بانک اطلاعاتی که ایجاد می‌کنید، یک گروه فایل پیش‌فرض برای آن ایجاد می‌گردد (گروه فایل پیش‌فرض، همان گروه فایل اصلی است).

---

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

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.
٤. 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).
٦. 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 ntext، text و 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

همان طور که دیدید، فایل های 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 [١,...,n];
```

database_name[١,...,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_data١,
  FILENAME = 'c:\classDatabase\my_data١.mdf',
  SIZE = ٥MB, MAXSIZE = ١٠٠MB, FILEGROWTH = ٥MB
)
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_data٢,FILENAME='c:\classDatabase
\my_data٢.mdf',SIZE=٥MB,MAXSIZE=١٠٠MB,FILEGROWTH = ٥MB
)
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 ٤MB است که می‌تواند تا ۵۰MB با گام ۵MB رشد کند.

```
ALTER DATABASE myDatabase
ADD FILE(NAME=my_data٣,FILENAME=N'c:\classdatabase

\my_data٣.mdf',SIZE=٤MB,MAXSIZE=٥٠MB,FILEGROWTH=٥MB)
 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,
    .
    .
    .
  Column$_n$ type$_n$ Constraint$_n$)
```

در این ساختار table_name، نام جدولی است که می‌خواهید ایجاد کنید. Column$_۱$ تا Column$_n$، نام فیلدهای (ستون‌های) جدول را تعیین می‌کنند. type$_۱$ تا type$_n$، نوع ستون‌های Column$_۱$ تا Column$_n$ را مشخص می‌کنند و Constraint$_۱$ تا Constraint$_n$ به ترتیب محدودیت‌هایی را تعیین می‌کنند که باید بر روی ستون‌های Column$_۱$ تا Column$_n$ 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(٦) 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(١٠) ,
 atID     varchar(١٠) ,
 Payment decimal(١٨, ٠),
 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 واقعی‌تر بیان گردیده، تمام مسائل حل شده در کل کتاب از این بانک اطلاعاتی استفاده می‌کند. این بانک اطلاعاتی برای خرید، فروش و پخش محصولات مختلف به کار می‌رود. برخی از مراحل بیان شده طراحی پایگاه داده در درس در زیر آمده است:

۱. تعیین جداول مورد نیاز، این بانک اطلاعاتی اطلاعات مشتریان، حساب‌ها و غیره را نگهداری می‌کند. جداول در نظر گرفته شده برای این بانک در زیر آمده‌اند:

- **جدول Customer**، اطلاعات مشتریان از قبیل کد مشتری، نام مشتری، نام خانوادگی مشتری، کد ملی و غیره را نگهداری می‌کند (جدول ۳ – ۱).
- **جدول City**، اطلاعات شهرها از قبیل کد شهر، نام شهر و کد کاربر را نگهداری می‌نماید (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(٦)	بله	نه	نه
	firstName	نام مشتری	Varchar(٣٠)	نه	نه	نه
	lastName	نام خانوادگی مشتری	Varchar(٣٠)	نه	نه	نه
	nationalCode	کد ملی	Varchar(١٠)	نه	نه	نه
	Tel	شماره تلفن	Varchar(١٥)	نه	بله	بله
	Mobile	شماره موبایل	Varchar(١٥)	نه	بله	نه
	Address	آدرس	Varchar(٥٠)	نه	بله	نه
	acivityType	نوع فعالیت	Varchar(٥٠)	نه	بله	نه
	cityID	کد شهر	Varchar(٦)	نه	بله	بله

	stateID	کد استان	Varchar(٦)	نه	بله	نه
	Email	ایمیل	Varchar(٥٠)	نه	بله	نه
	goodAccount	خوش حسابی	Bit	نه	بله	نه
	Credit	میزان اعتبار	Decimal(١٨،٠)	نه	بله	نه
	userCode	کد کاربر	Varchar(٦)	نه	بله	بله
Login١ (کاربران)	userCode	کد کاربر	Varchar(٦)	بله	نه	نه
	usernName	نام کاربر	Varchar(٥٠)	نه	نه	نه
	password	کلمه عبور	Varchar(٥٠)	نه	بله	نه
City (شهرها)	cityID	کد شهر	Varchar(٦)	بله	نه	نه
	cityName	نام شهر	Varchar(٥٠)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله

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

نام جدول	نام فیلد	هدف	نوع	کلید اولیه	تهی	کلید خارجی
State (استان‌ها)	stateID	کد استان	Varchar(٦)	بله	نه	نه
	stateName	نام استان	Varchar(٥٠)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
Store (انبار)	storeID	کد انبار	Varchar(٦)	بله	نه	نه
	storeName	نام انبار	Varchar(٥٠)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	نه
Merchandise (کالا)	merchandiseID	کد کالا	Varchar(٦)	بله	نه	نه
	merchandiseGroupID	کد گروه کالا	Varchar(٦)	نه	نه	بله
	merchandiseName	نام کالا	Varchar(٦)	نه	نه	نه
	consumerPrice	قیمت مصرف کننده	Numeric(١٨, ٠)	نه	بله	نه
	buyPrice	قیمت خرید	Numeric(١٨, ٠)	نه	بله	نه
	poketSellingPrice	قیمت فروش بسته‌ای	Numeric(١٨, ٠)	نه	بله	نه
	changueSellingPrice	قیمت تعویض	Numeric(١٨, ٠)	نه	بله	نه
	hav	موجودی	Numeric(١٨, ٠)	نه	بله	نه
	peritionAmount	حد سفارش	Numeric(١٨, ٠)	نه	بله	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
merchandiseGroup (گروه کالا)	merchandiseGroupID	کد گروه کالا	Varchar(٦)	بله	نه	نه
	merchandiseGroupName	نام گروه کالا	Varchar(٥٠)	نه	نه	نه
	unit	واحد	Varchar(٥٠)	نه	بله	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
BankSave (بانک)	fishNo	شماره فیش	Varchar(١٥)	بله	نه	نه
	hesabNo	شماره حساب	Varchar(١٥)	نه	نه	نه
	Date	تاریخ	Varchar(٨)	نه	نه	نه
	mablagTake	مبلغ برداشتی	Decimal(١٨, ٠)	نه	نه	نه
	mablagSave	مبلغ واریزی	Decimal(١٨, ٠)	نه	نه	نه
	sellerID	کد فروشنده	Varchar(٦)	نه	نه	بله
	comment	توضیحات	Varchar(٦)	نه	بله	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
BuyInvoice (فاکتور خرید)	invoiceNumeral	شماره فاکتور	Decimal(١٨, ٠)	بله	نه	نه
	Date	تاریخ فاکتور	Varchar(٨)	نه	نه	نه
	ID	شماره مشتری	Varchar(٦)	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(٦)	نه	نه	بله
buyInvoice۱ (ریز فاکتور خرید)	invoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(٦)	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(٦)	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(٦)	نه	نه	بله
	sellType	نوع فروش	Varchar(۵۰)	نه	بله	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
sellInvoice۱ (ریز فاکتور فروش)	invoiceNumeral	شماره فاکتور	Decimal(۱۸, ۰)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(٦)	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(٦)	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(٦)	نه	نه	بله
Seller (فروشندگان)	sellerID	شماره فروشنده	Varchar(٦)	بله	نه	نه
	sellerName	نام فروشنده	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
Recipt (رسید)	recepitID	شماره رسید	Varchar(٦)	بله	نه	نه
	storeID	کد انبار	Varchar(٦)	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	تحویل گیرنده	Varchar(۵۰)	نه	نه	نه
	Teller	تحویل دهنده	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
Receipt۲ (ریز کالا رسید شده)	receiptID	شماره رسید	Varchar(٦)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(٦)	بله	نه	بله
	Amount	تعداد کالا	Numeric(۱۸،۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
Order (حواله)	receiptID	شماره حواله	Varchar(٦)	بله	نه	نه
	storeID	کد انبار	Varchar(٦)	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	تحویل گیرنده	Varchar(۵۰)	نه	نه	نه
	Teller	تحویل دهنده	Varchar(۵۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله
Order۲ (ریز کالای حواله شده)	receiptID	شماره حواله	Varchar(٦)	بله	نه	بله
	merchandiseID	کد کالا	Varchar(٦)	بله	نه	بله
	Amount	تعداد کالا	Numeric(۱۸،۰)	نه	نه	نه
	userCode	کد کاربر	Varchar(٦)	نه	نه	بله

- **جدول Login**، اطلاعات کاربران از قبیل کد کاربر، نام کاربر و کلمه عبور را نگهداری می‌کند (جدول ۳-۱).
- **جدول State**، اطلاعات استان‌ها از قبیل کد استان، نام استان و کد کاربر را نگهداری می‌کند (جدول ۳ - ۱).
- **جدول Store**، اطلاعات انبار از قبیل کد انبار، نام انبار و کد کاربر را نگهداری می‌کند (جدول۳ - ۱).
- **جدول Merchandise**، اطلاعات کالا از قبیل کدکالا، نام کالا، کد گروه کالا، قیمت مصرف‌کننده، قیمت خرید و غیره را نگهداری می‌کند (جدول ۳ - ۱).
- **جدول MerchandiseGroup**، اطلاعات گروه کالا از قبیل کد گروه، نام گروه، واحد و کد کاربر را نگهداری می‌نماید (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 می‌نماید (جدول ۳ – ۱).
- **جدول buyInvoice**، اطلاعات فاکتور از قبیل شماره فاکتور، تاریخ فاکتور، شماره مشتری، نوع خرید و کد کاربر را ذخیره می‌کند (جدول۳ – ۱).
- **جدول buyInvoice۱**، اطلاعات ریز فاکتورها از قبیل شماره فاکتور، کد کالا، قیمت، تخفیف، تعداد و کد کاربر را ذخیره می‌کند (جدول ۳ – ۱).
- **جدول SellInvoice**، اطلاعات فاکتور فروش از قبیل شماره فاکتور، تاریخ، کد مشتری و غیره را نگهداری می‌کند (جدول ۳ – ۱).
- **جدول SellInvoice۱**، اطلاعات ریز فاکتور فروش از قبیل شماره فاکتور، کد کالا، قیمت، تخفیف، تعداد و کد کاربر را نگهداری می‌نماید (جدول ۳ – ۱).
- **جدول Hessab**، اطلاعات حساب‌ها از قبیل شماره حساب، نام بانک، نام حساب و کد کاربر را ذخیره می‌نماید (جدول ۳ – ۱).
- **جدول Seller**، اطلاعات فروشندگان از قبیل شماره فروشنده، نام فروشنده و کد کاربر را ذخیره می‌نماید (جدول ۳ – ۱).
- **جدول Receipt**، اطلاعات رسیدها از قبیل شماره سند، کد انبار، تاریخ، شماره فاکتور و غیره را ذخیره می‌نماید (جدول ۳ – ۱).
- **جدول Receipt۲**، اطلاعات ریز رسیدها از قبیل شماره رسید، کد کالا، تعداد و کد کاربر را ذخیره می‌کند (جدول ۳ – ۱).
- **جدول Order**، اطلاعات حواله‌ها از قبیل شماره حواله، کد انبار، تاریخ و غیره را نگهداری می‌کند (جدول ۳ – ۱).
- **جدول Order۲**، اطلاعات ریز حواله‌ها از قبیل شماره حواله، کد کالا، تعداد و کد کاربر را نگهداری می‌کند (جدول ۳ – ۱).

۲. تعیین فیلدهای کلید اولیه

همان‌طور 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 نمی‌توانند 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 نمی‌تواند 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 نمی‌توانند کد یکسان داشته باشند.
- **در جدول MerchandiseGroup**، فیلد merchandiseGroupID کلید اولیه است. زیرا، کد گروه کالا نمی‌تواند تکراری باشد.
- **در جدول BankSave**، فیلد fishNo کلید اولیه است. زیرا، هیچ دو فیشی نمی‌توانند 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 buyInvoice۱ و SellInvoice۱**، ترکیب فیلدهای invoiceNumeral، merchandiseID کلید اولیه است.
- **در جدول Hesab**، فیلد hesabNo کلید اولیه است، چون هیچ دو حسابی نمی‌توانند شماره تکراری داشته باشند.
- **در جدول Seller**، فیلد sellerID کلید اولیه است. چون کد هیچ دو فروشنده نمی‌تواند یکی باشد.
- **در جدول Receipt و Order**، به ترتیب receiptID و orderID کلید اولیه هستند.
- **در جدول Receipt۲**، ترکیب فیلدهای receiptID و merchandiseID کلید اولیه است.
- **در جدول Order۲**، ترکیب فیلدهای orderID و merchandiseID کلید اولیه است.

۳. تعیین ارتباط بین جداول

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

- بین جداول Customer و Login۱، فیلد ارتباط userCode است. این فیلد در جدول Customer کلید خارجی، ولی در جدول Login کلید اولیه است.
- بین جدول Customer و City، فیلد ارتباط cityID است. این فیلد در جدول Customer کلید خارجی است.
- بین جدول Customer و State، فیلد ارتباط stateID است که در جدول Customer کلید خارجی است. چون در جدول State کلید اولیه می‌باشد.
- بین جدول Login۱ و تمام جدول، userCode ارتباط را برقرار می‌کند که در جدول Login۱، فیلد userCode کلید اولیه است، اما، در بقیه جداول کلید خارجی می‌باشد.
- بین جداول Store، جداول Receipt و Order فیلد ارتباط storeID است. این فیلد در جدول Store کلید اولیه است. اما در جدول Receipt و Order کلید خارجی می‌باشد.

- بین جدول Merchandise و جداول Receipt۱، Order۱، buyInvoice۱ و SellInvoice۱ 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؟

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

جدول ٤ – 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
	capicity	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

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

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.

نکته:	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 (۲۰)varchar) 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 می‌کند، برای ذخیره اطلاعات فیلدهای زیر از چه انواعی استفاده می‌شوند:

الف: شماره دانشجویی ب: نمره ج: مدرک استاد د: کد ملی

پ: عکس دانشجو ت: اطلاعات XML خ: تاریخ فارغ‌التحصیلی چ: قیمت کالا

۲۰. جدولی به نام Test با فیلدهای x۱ و x۲ از نوع int به بانک TransportationAgency اضافه کنید.

۲۱. فیلد x۲ را در جدول Test کلید اولیه تعیین کنید.

۲۲. جدول Test را حذف کنید.

۸ – ۱. تمرین‌ها

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.

٤. ارتباط بین جداول را تعیین کرده و نوع ارتباط را مشخص نمایید.

۵. جداول را به همراه ارتباط بین آن‌ها در بانک اطلاعاتی Accounting ایجاد کنید.

جدول ۵ – ٤ توزیع فیلدهای جداول سیستم حسابداری.

نام جدول	نام فیلد	نوع فیلد	اندازه	د 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

سربرگ 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 تفضیلی 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

فصل ۲ 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 دست‌کاری 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	۰۱۱۱۲۲۵۶۶۸۷	www.fanavarienovin.net	بابل	رمضان	عباس‌نژاد	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	عباس‌نژاد	۴۲	فوق لیسانس	fanavarienovin@yahoo.com	۰۹۱۱۱۱۱۶۲۱۲	۲۸۰۰۰۰۰۰
۰۲	یوسف	عباس‌نژاد	۲۴	لیسانس	yousef_dssht@yahoo.com	۰۹۱۱۲۱۸۵۴۲۶	
۰۳	باقر	رحیم‌پور	۳۲	فوق لیسانس			۱۰۰۰۰۰۰۰
۰۴	مرتضی	پویان	۲۷	فوق لیسانس			
۰۵	سیدحجت	جلیلی	۳۵	لیسانس			۶۰۰۰۰۰۰

						اله	

اطلاعات کتاب‌ها (Books).

ISBN	Title	page	Price	editNO	printNO	groupID
۶۰۰۹۱۴۱۳۹۵	اصول طراحی پایگاه داده	۲۵۶	۷۳۰۰۰	۱	۱	۰۲
۶۰۰۹۲۲۵۴۲۲	طراحی سیستم‌های شی گرا	۴۳۲	۱۳۰۰۰۰	۱	۱	۰۱
۶۰۰۹۱۴۱۳۰۲	حل مسائل C++	۲۱۶	۵۷۰۰	۱	۲	۰۱
۶۰۰۹۱۴۱۳۴۰	حل مسائل C#	۲۰۰	۶۲۰۰۰	۱	۲	
۶۰۰۹۱۴۱۳۸۸	امنیت شبکه	۱۹۲	۶۵۰۰۰	۱	۱	۰۳
۶۰۰۹۲۲۵۴۳۹	مدیریت استراتژیک	۲۷۲	۷۵۰۰۰	۱	۱	۰۴
۷۰۰۹۲۲۵۴۳۱	رهیافت C++	۳۰۴	۴۰۰۰۰	۱	۱	۰۱

اطلاعات جدول PubBook.

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

اطلاعات جدول GroupBook.

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

اطلاعات جدول AutBook.

ISBN	atID	Payment
۶۰۰۹۱۴۱۳۹۵	۰۱	۵۰۰۰۰۰۰
۶۰۰۹۱۴۱۳۹۵	۰۳	۵۰۰۰۰۰۰
۶۰۰۹۲۲۵۴۲۲	۰۱	۵۰۰۰۰۰۰
۶۰۰۹۲۲۵۴۲۲	۰۳	۵۰۰۰۰۰۰
۶۰۰۹۱۴۱۳۰۲	۰۱	۶۰۰۰۰۰۰
۶۰۰۹۱۴۱۳۴۰	۰۱	۶۰۰۰۰۰۰
۶۰۰۹۱۴۱۳۸۸	۰۱	۶۰۰۰۰۰۰
۶۰۰۹۲۲۵۴۳۹	۰۵	۶۰۰۰۰۰۰
۷۰۰۹۲۲۵۴۳۱	۰۱	۶۰۰۰۰۰۰

شکل ۱ – ۲ **اطلاعات جدول بانک اطلاعاتی PublishDB.**

۱–۲. 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 «وَ» بیتی، «یا» بیتی و «یا انحصاری» بیتی را انجام می‌دهند (جدول۲- ۲).

عملگرهای رابطه‌ای

عملگرهایی هستند که دو عبارت (عملوند) را با هم مقایسه کرده، نتیجه مقایسه را به صورت منطقی برمی‌گردانند (جدول۳- ۲).

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 ۴- ۲ آمده‌اند.

---

۵. True ۶. False ۷. Null

جدول ۳ - ۲ عملگرهای رابطه‌ای در SQL.

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

جدول ٤ - 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).

٦. 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 بازیابی می‌کنیم. سپس، رواسای انتشارات را بازیابی 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	عباس نژاد
4	سید حجت اله	جلیلی
5	محمد رضا	حاتمی
6	مرتضی	یوبان
7	یوسف	عباس 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	عباس نژاد
2	یوسف	عباس نژاد
3	باقر	رحیم پور
4	مرتضی	یوبان
5	سید حجت اله	جلیلی
6	رمضان	عباس نژاد
7	محمد 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 پرس‌وجو اشتراک می‌گیرد (مانند عملگر ∩ جبر رابطه‌ای) و نتیجه اشتراک را برمی‌گرداند.

مثال ۲۵ – ۲. پرس‌وجویی که کد مؤلفینی که کتاب تألیف کرده‌اند را نمایش می‌دهد.

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

	atID
1	01
2	03
3	05

مثال ۲۶ – ۲. پرس‌وجویی که نام و نام خانوادگی افرادی که هم مؤلف و هم مدیر انتشارات هستند را نمایش می‌دهد.

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

	aFname	aLname
1	رمضان	عباس نژاد

۴ – ۶ – ۲. عملگر EXCEPT

این عملگر تفاضل بین نتیجه پرس‌وجوی اول و دوم را برمی‌گرداند (مانند عملگر – در جبر رابطه‌ای). یعنی، رکوردهایی را برمی‌گرداند که در پرس‌وجوی اول باشد، ولی در نتیجه پرس‌وجوی دوم نباشد.

مثال ۲۷ – ۲. پرس‌وجویی که کد مؤلفینی که کتاب چاپ شده در این بانک ندارند، را نمایش می‌دهد.

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

	atID
1	02
2	04

مثال ۲۸ – ۲. پرس‌وجویی که نام و نام خانوادگی رواسای انتشارات را نمایش می‌دهد که مؤلف نیستند.

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

	bFname	bLname
1	رضا	امینی
2	محمدرضا	حاتمی

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

نکته: همان‌طور که بیان گردید، عبارت ORDER BY برای مرتب‌سازی رکوردها به کار می‌رود. در هنگام به کارگیری ORDER BY در پرس‌وجوهای مرکب باید دقت کنید که این عبارت را فقط در یک پرس‌وجو تایپ کنید.

مثال ۲۹ – ۲. پرس‌وجویی که شماره مؤلفینی را نمایش می‌دهد که در جدول AutBook حق تألیف نگرفته‌اند (اطلاعات را بر اساس شماره مؤلف به صورت نزولی مرتب کرده، نمایش می‌دهد).

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

	atID
1	04
2	02

جدول ۷ – ۲ توابع تجمعی (خلاصه‌سازی).

تابع	ورودی	هدف

AVG	عددی	میانگین مقادیر را برمی‌گرداند.
SUM	عددی	مجموع مقادیر را برمی‌گرداند.
MAX	عددی یا غیر عددی	بزرگ‌ترین مقدار را برمی‌گرداند.
MIN	عددی یا غیر عددی	کوچک‌ترین مقدار را برمی‌گرداند.
COUNT	عددی یا غیر عددی	تعداد رکوردها یا مقادیر غیر تهی را برمی‌گرداند.
STDEV	عددی	انحراف معیار یک فیلد یا چند مقدار را برمی‌گرداند.
STDEVP	عددی	انحراف معیار توزیعی یک فیلد یا چند مقدار را برمی‌گرداند
VAR	عددی	واریانس مقادیر یک یا چند فیلد را برمی‌گرداند.
VARP	عددی	واریانس توزیعی مقادیر یک فیلد یا چند مقدار را برمی‌گرداند

۷ – ۲. توابع تجمعی

توابع تجمعی برای خلاصه‌سازی اطلاعات به کار می‌روند (یعنی، مجموعه‌ای (کلکسیونی) از مقادیر را به عنوان ورودی دریافت کرده، یک مقدار را برمی‌گرداند). با توابع تجمعی می‌توانید مجموع مقادیر، بزرگ‌ترین مقدار، کوچک‌ترین مقدار، میانگین مقادیر یک یا چند فیلد را برگردانید. نمونه‌ای از این توابع در جدول ۷ – ۲ آمده‌اند.

۱۰ – ۲. پرس‌وجوی فرعی

پرس‌وجویی که نتیجه آن توسط پرس‌وجوی دیگر مورد استفاده قرار می‌گیرد، **پرس‌وجوی فرعی**[۲] **(متداخل**[۳]**)** نام دارد. در پرس‌وجوهای فرعی، ممکن است چندین پرس‌وجو به صورت پشت سرهم قرار گیرند. در این صورت، نتیجه پرس‌وجوی سطح بعدی در سطح قبلی استفاده می‌شود و این روند تا اولین سطح پرس‌وجو ادامه می‌یابد. پرس‌وجوهای فرعی را می‌توان در عبارت WHERE و HAVING مربوط به پرس‌وجوی اصلی استفاده کرد. در هنگام استفاده از پرس‌وجوهای فرعی باید نکات زیر را رعایت کنید:

- از عملگر BETWEEN نمی‌توان در پرس‌وجوی فرعی استفاده کرد. ولی، در پرس‌وجوی اصلی می‌توان از این عملگر استفاده نمود.
- پرس‌وجوهای فرعی باید در داخل پرانتز قرار گیرند.
- پرس‌وجوهای فرعی که بیش از یک رکورد (تاپل) را برمی‌گردانند، می‌توانند با عملگرهایی مانند IN مورد استفاده قرار گیرند.
- در یک پرس‌وجوی فرعی نمی‌توان از گزینه ORDER BY استفاده کرد. برای این منظور، می‌توان از گزینه GROUP BY استفاده نمود تا عمل ORDER BY نیز انجام شود.

پرس‌وجوی فرعی به صورت زیر به کار می‌رود:

دستور پرس‌وجوی اصلی

...

WHERE [**لیست فیلدها**] **عملگر** (**دستور پرس‌وجوی فرعی**)

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	فناوري نوين	011122566687	fanavarienovin.net	بابل	رمضان	عباس نژاد	15
2	02	دانش نگار	02166400220		تهران	محمد رضا	حاتمي	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	اميني	10

مثال ۳۸ – ۲. پرس‌وجویی که لیست گروه کتاب‌های را بازیابی 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)
```

	groupID	groupName
1	01	برنامه نويسي
2	02	پايگاه داده
3	03	شبكه رايانه اي
4	04	فناوري اطلاعات

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	يوسف	عباس نژاد	26	ليسانس	yousef_dssht@yahoo.com	NULL	NULL
2	04	مرتضي	پويان	27	NULL	NULL	NULL	NULL

همان‌طور که در این پرس‌وجو می‌بینید، عملگر NOT EXISTS مولفینی که هیچ حق تالیفی نگرفته‌اند را تعیین می‌کند. به جای این دستور می‌توانید از عملگر NOT IN به صورت زیر استفاده کنید:

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

مثال ۴۰ – ۲. پرس‌وجویی که گروه‌هایی که اصلاً کتاب ندارند را نمایش می‌دهد (برای حل این پرس‌وجو از عملگر EXCEPT استفاده کنید).

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

	groupID	groupName
1	05	گرافیک رایانه ای

پرس‌وجوی فرعی کد گروه‌هایی را برمی‌گرداند که کتاب ندارند (با عملگر EXCEPT) و پرس‌وجوی اصلی اطلاعات این گروه‌ها را نمایش می‌دهد.

مثال ۴۱ – ۲. پرس‌وجویی که کتاب‌های را نمایش می‌دهد که حق تألیف برای آن‌ها دریافت گردید (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 طراحي پايگاه داده	256	73000	1	1	02
5	6009225422	اصول طراحي سيستم هاي شي گرا	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 ro

پرس‌وجوی فرعی، کد کتاب‌هایی را برمی‌گرداند که در جدول AutBook هستند. سپس، پرس‌وجوی اصلی، اطلاعات کتاب‌ها را نمایش می‌دهد.

مثال ۴۲ – ۲. پرس‌وجویی که کتاب‌های با کد گروه کتاب '۰۲' را نمایش می‌دهد که تاکنون برای آن‌ها حق تألیف دریافت نگردید (با عملگر EXCEPT).

```
SELECT * FROM Books
WHERE ISBN IN
(SELECT ISBN  FROM Books
WHERE groupID ='۰۱'
 EXCEPT
 SELECT ISBN  FROM AutBook)
```

	ISBN	Title	Page	Price	editNo	printNo	groupID

پرس‌وجوی فرعی شابک کتاب‌های با کد گروه '۰۱' که هیچ حق تألیفی برای آن‌ها دریافت نگردید را برمی‌گرداند و پرس‌وجوی اصلی، کلیه اطلاعات این کتاب‌ها را نمایش می‌دهد.

۲ – ۱۱ – ۲. پیوند متعادل

پیوند متعادل، رایج‌ترین پیوند است که با استفاده از فیلد مشترک بین دو جدول انجام می‌شود. این پیوند همان ضرب دکارتی بین دو جدول است، به طوری که شرط برابری فیلد مشترک آن‌ها باید در ضرب دکارتی ذکر گردد. الحاق بین دو جدول به صورت زیر انجام می‌شود:

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

مثال ۴۶ – ۲. پرس‌وجویی که اطلاعات نام کتاب و نام گروه را نمایش می‌دهد.

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

	Title	groupName
1	حل مسائل C++	برنامه نويسي
2	حل مسائل C#	برنامه نويسي
3	امنيت شبكه	شبكه رايانه اي
4	اصول طراحي پايگاه داده	پايگاه داده
5	اصول طراحي سيستم هاي شي گرا	برنامه نويسي
6	مديريت استراتژيک	فناوري اطلاعات
7	رهيافت C++	برنامه نويسي

همان‌طور که در این پرس‌وجو می‌بینید، شرط WHERE، عبارت b.groupID = g.groupID است. (فیلد groupID، فیلد مشترک بین جداول Books و GroupBook می‌باشد).

مثال ۴۷ – ۲. پرس‌وجویی که نام ناشر، نام خانوادگی مدیر انتشارات ، نام کتاب و میزان حق نشر دریافت شده برای هر کتاب را نمایش می‌دهد.

```
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	فناوري نوين	رمضان	عباس نژاد	حل مسائل C++	6000000
2	فناوري نوين	رمضان	عباس نژاد	حل مسائل C#	7000000
3	فناوري نوين	رمضان	عباس نژاد	امنيت شبكه	6000000
4	فناوري نوين	رمضان	عباس نژاد	اصول طراحي پايگاه داده	6000000
5	فناوري نوين	رمضان	عباس نژاد	اصول طراحي سيستم هاي شي گرا	8000000
6	فناوري نوين	رمضان	عباس نژاد	مديريت استراتژيک	5000000
7	فناوري نوين	رمضان	عباس نژاد	رهيافت C++	3500000
8	دانش نگار	محمد رضا	هاتمي	رهيافت C++	3500000

همان‌طور که در این پرس‌وجو می‌بینید، برای برقراری ارتباط بین سه جدول، ارتباط بین جداول 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.

۳ – ۱۱ – ۲. پیوند نامتعادل

در پیوند متعادل 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. پرس‌وجویی که دو جدول Books و GroupBook را از طریق عملگر < پیوند می‌زند.

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

	ISBN	Title	Page	Price	editNo	printNo	groupID	groupID	groupName
1	6009141388	امنيت شبكه	192	65000	1	1	03	01	برنامه نويسي
2	6009141388	امنيت شبكه	192	65000	1	1	03	02	پايگاه داده
3	6009141395	اصول طراحي پايگاه داده	256	73000	1	1	02	01	برنامه نويسي
4	6009225439	مديريت استراتژيک	272	75000	1	1	04	01	برنامه نويسي
5	6009225439	مديريت استراتژيک	272	75000	1	1	04	02	پايگاه داده
6	6009225439	مديريت استراتژيک	272	75000	1	1	04	03	شبکه رايانه اي

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
INNER   JOIN  t۲ ON ۱ شرط پیوند
INNER   JOIN  t۳ ON  ۲ شرط پیوند
...
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	امنيت شبكه	6000000
4	رمضان	عباس نژاد	اصول طراحي پايگاه داده	5000000
5	باقر	رحيم پور	اصول طراحي پايگاه داده	5000000
6	رمضان	عباس نژاد	اصول طراحي سيستم هاي شي گرا	5000000
7	باقر	رحيم پور	اصول طراحي سيستم هاي شي گرا	5000000
8	سيد حجت اله	جليلي	مديريت استراتژيک	6000000
9	رمضان	عباس نژاد	رهيافت C++	6000000

مثال ۵۰ – ۲. پرس‌وجویی که نام ناشر، نام کتاب و مبلغ حق نشر کتابی که حق نشر آن زیر ٤٠٠٠٠٠٠ 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 < ٤٠٠٠٠٠٠
```

	pName	Title	Payment
1	فناوري نوين	رهيافت C++	3500000
2	دانش نگار	رهيافت C++	3500000

مثال ۵۱ – ۲. پرس‌وجویی که لیست ناشرانی را بازیابی می‌کند که برای هیچ کتابی حق نشر بیش از ٤٠٠٠٠٠٠ 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 > ٤٠٠٠٠٠٠)
```

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

مثال ۵۲ – ۲. پرس‌وجویی که لیست مؤلفانی که برای هیچ کتابی حق تألیف بیش از ۵۰۰۰۰۰۰ ریال دریافت نکردند، را نمایش می‌دهد.

```
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 > ٥٠٠٠٠٠٠ )
```

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

١٤ – ٢. مسائل حل شده

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

```
/****** اضافه کردن کاربر ******/
INSERT INTO Login١  VALUES('٠١', 'Admin', '١٣٥٧٩٢٤٦٨')
INSERT INTO Login١  VALUES('٠٢', 'test', '٨٧٦٥٤٣٢١٠٩')
/******اضافه کردن شهر ******/
INSERT INTO City  VALUES('٠٢١', '٠١' ,'تهران')
INSERT INTO City  VALUES('٠١١١', '٠٢' ,'بابل')
INSERT INTO City  VALUES('٠١٢١', '٠٢' ,'آمل')
INSERT INTO City  VALUES('٠١٥١', '٠٢' ,'ساری')
INSERT INTO City  VALUES('٠٣١١', '٠١' ,'اصفهان')
INSERT INTO City  VALUES('٠٥١١', '٠١' ,'مشهد')
/******اضافه کردن استان ******/
INSERT INTO State  VALUES('٠١', '٠١' ,'تهران')
INSERT INTO State VALUES('٠٢', '٠١' ,'اصفهان')
INSERT INTO State  VALUES('٠٣', '٠١' ,'مشهد')
/******اضافه کردن فروشنده ******/
INSERT INTO Seller   VALUES('٠١','٠١' ,'احمدی')
INSERT INTO Seller VALUES('٠٢', '٠١' ,'رضائی')
INSERT INTO Seller  VALUES('٠٣', '٠٢' ,'محمدی')
/******اضافه کردن  انبار ******/
INSERT INTO Store VALUES('٠١','٠١', 'انبار مرکزی')
INSERT INTO Store VALUES('٠٢', '٠١', 'انبار شماره ١')
/****** اضافه کردن  گروه کالا ******/
INSERT INTO merchandiseGroup VALUES('٠١', '٠٢', 'کیلو',
       'حبوبات')
INSERT INTO merchandiseGroup VALUES('٠٢','٠١', 'نوشیدنی',
       'باطری')
INSERT INTO merchandiseGroup VALUES('٠٣','٠٢' ,'متر' ,'کابل')
/******اضافه کردن مشتری *****/
INSERT INTO Customer VALUES  ('٠١', 'رضا', 'احمدی',
'٢١٥٢٠٦٤٤٤٣', '٠٢١٦٦٤٠٠٢٢٠', '٠٩١٢٢٢٢٠٩٩٩', 'تهران خیابان
ولیعصر', NULL, '٠٢١', '٠١', NULL, ١, ١٠٠٠٠٠٠٠, '٠١')
INSERT INTO Customer VALUES  ('٠٢','علی', 'رضوی',
'٠٥١١٦٦٤٠٢٢٠٠', '٠٩١٥٢٢٢٠٩٩٩', 'مشهد خیابان امام رضا', NULL,
'٠٥١١', '٠٣', NULL, ١, ١٠٠٠٠٠٠٠, '٠١')
/****** اضافه کردن کالا ******/
INSERT INTO Merchandise  VALUES ('٠١','٠٢', 'نوشابه بزرگ',
٦٠٠٠, ٤٠٠٠, ٥٠٠٠, ٥٥٠٠, ١٠٠٠, '٠١')
```

```
INSERT INTO Merchandise  VALUES ('۰۲','۰۲', 'دوغ بزرگ', ٦۰۰۰,
٤۰۰۰, ٥۰۰۰, ٥۷۰۰, ٥۰۰, '۰۱')
INSERT INTO Merchandise VALUES('۰۳','۰۲', 'برنج', ۳۲۰۰۰,
۲٦۰۰۰, ۲٥۰۰۰, ۲٦۰۰۰, ۱٥۰۰۰, '۰۲')
/****** اضافه کردن فاکتور خرید ******/
INSERT INTO BuyInvoice VALUES(۹۷٥۰۰,'۹۱/۰۳/۳۱','۰۱','۰۲' ,'نقد')
INSERT INTO BuyInvoice VALUES(۹۷٥۰۱,'۹۱/۰٤/۰۱','۰۲','۰۲' ,'چک')
INSERT INTO BuyInvoice VALUES(۹۷٥۰۲,'۹۱/۰٤/۱٥','۰۱','۰۲' ,'نقدی')
/****** اضافه کردن ریز فاکتور خرید ******/
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۰,'۰۱',٤۰۰۰،۰،٥۰۰,'۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۰,'۰۲',٥۰۰۰،۰،۳۰۰,'۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۰,'۰۳',۱٥۰۰۰،۰،۳۰۰, '۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۱,'۰۱',٤۰۰۰،۰،۲۰۰,'۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۱,'۰۲',٥۰۰۰,۰,۱۰۰,'۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۱,'۰۳',۱٥۰۰۰،۰،۳۰۰,'۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۲,'۰۱',٤۰۰۰،۰,۲۰۰,'۰۲')
INSERT INTO BuyInvoice۱ VALUES (۹۷٥۰۲,'۰۳',۱٥۰۰۰, ۰،۱۰۰,'۰۲')
```

مثال ۲. پرس‌وجویی که استان 'مازندران' با کد استان '۰٤' را کاربری با کد '۰۲' به جدول State اضافه می‌کند.

```
INSERT INTO State VALUES('۰٤', '۰۲' ,'مازندران')
```

مثال ۳. پرس‌وجویی که اطلاعات جدول State را به جدول State۱ اضافه می‌کند.

```
SELECT * INTO State۱ FROM State
```

مثال ۴. پرس‌وجویی که اطلاعات جدول State۱ را نمایش می‌دهد.

```
SELECT * FROM State۱
```

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

مثال ۵. پرس‌وجویی که مقدار فیلدهای کد استان (stateID) و نام استان (stateName) جدول State را به جدول State۱ اضافه می‌کند.

```
INSERT INTO State۱( stateID, stateName)
  SELECT stateID, stateName FROM State
```

مثال ۶. پرس‌وجویی که اطلاعات جدول State۱ را نمایش می‌دهد.

```
SELECT *  FROM State۱
```

	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

همان‌طور که در این خروجی می‌بینید، برخی از رکوردها مقدار فیلد userCode آن‌ها NULL است. چون، فقط مقدار فیلدهایی کد استان و نام استان از جدول State به جدول State۱ اضافه شدند.

مثال ۷. پرس‌وجویی که رکوردهایی از جدول State۱ را حذف می‌کند که مقدار فیلد userCode آن‌ها NULL نیست. سپس رکوردهای باقی مانده را نمایش می‌دهد.

```
DELETE FROM State۱ WHERE userCode IS NOT NULL
SELECT * FROM State۱
```

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

همان‌طور که در خروجی مشاهده می‌گردد، رکوردهایی که مقدار فیلد userCode آن‌ها NULL نبود، حذف شدند.

مثال ۸. پرس‌وجویی که رکوردهایی از جدول State۱ را حذف می‌کند، که مقدار فیلد userCode آن‌ها NULL باشد و سپس اطلاعات جدول State۱ را نمایش می‌دهد.

```
DELETE FROM State۱ WHERE userCode IS NULL
SELECT * FROM State۱
```

مثال ۳۵. پرس‌وجویی که کاربرانی که هم اطلاعات استان‌ها و هم اطلاعات مشتریان را وارد کرده‌اند یا تغییر داده‌اند، نمایش می‌دهد.

```
SELECT userName FROM Login۱
 WHERE userCode IN( SELECT userCode FROM Customer
 INTERSECT
 SELECT userCode FROM City )
```

	userName
1	Admin

مثال ۳۶. پرس‌وجویی که نام و نام خانوادگی مشتریانی که از آن‌ها خرید نداشته‌ایم را بازیابی می‌کند.

```
SELECT firstName, lastName FROM Customer
 WHERE ID NOT IN(SELECT ID FROM BuyInvoice۱)
```

	firstName	lastName

مثال ۳۷. پرس‌وجویی که گروه‌های کالایی را بازیابی می‌کند 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
       BuyInvoice۱ ) )
```

این پرس‌وجوی از سه پرس‌وجوی تودرتو تشکیل شده است. داخلی‌ترین پرس‌وجو، کالاهایی که تاکنون از آن‌ها خرید انجام شده را بازیابی می‌کند:

```
( SELECT merchandiseID FROM BuyInvoice۱ )
```

پرس‌وجو داخلی دوم، کد گروه کالاهایی را بازیابی می‌کند که از آن‌ها خرید انجام شده است:

```
(SELECT merchandiseGroupID FROM Merchandise
WHERE merchandiseID IN ( SELECT merchandiseID FROM
BuyInvoice۱ ) )
```

و در نهایت پرس‌وجو کلی، گروه‌های کالا را برمی‌گرداند که از آن گروه کالا خرید انجام شده است.

مثال ۳۸. پرس‌وجویی 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
BuyInvoice۱ ) )
```

	merchandiseGroupID	merchandiseGroupName	unit	userCode
1	03	کابل	متر	02

این پرس‌وجو مانند پرس‌وجوی قبلی است. با این تفاوت که در بخش WHERE پرس‌وجوی خارجی به جای IN، عملگر NOT IN آمده است.

مثال ۳۹. پرس‌وجویی 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,BuyInvoice۱ 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	احمدي	نوشابه بزرگ
2	97500	رضا	احمدي	دوغ بزرگ
3	97500	رضا	احمدي	برنج
4	97502	رضا	احمدي	نوشابه بزرگ
5	97502	رضا	احمدي	برنج
6	97501	علي	رضوي	نوشابه بزرگ
7	97501	علي	رضوي	دوغ بزرگ
8	97501	علي	رضوي	برنج

۱۵-۲. دستور کار آزمایشگاه

۱. یکسری اطلاعات به جدول رانندگان (Drivers)، ماشین (Cars)، مشتریان (Customers) و سفارشات (Orders) اضافه کرده، سپس به پرس‌وجوهای زیر پاسخ دهید:

۲. پرس‌وجویی بنویسید که تمام ماشین‌هایی را بازیابی کند که واحد آن‌ها تن است.

۳. پرس‌وجوی بنویسید تا نام و نام خانوادگی رانندگانی را بازیابی کند که در گواهی‌نامه آن‌ها کلمه 'پایه۱' آمده است.

٤. پرس‌وجویی بنویسید تا اطلاعات رانندگانی را بازیابی کند که سفارشات شهر شیراز یا تهران را انجام داده‌اند.

۵. پرس‌وجویی بنویسید تا اطلاعات ماشین‌هایی را بازیابی کند که سفارشات شهر آمل یا بابل را انجام داده‌اند (مقصد مشتریان آن‌ها آمل یا بابل بوده است).

٦. پرس‌وجویی بنویسید تا اطلاعات ماشین‌هایی را بازیابی کند که سفارشات شهر آمل یا بابل را انجام نداده‌اند (مقصد مشتریان آن‌ها آمل یا بابل نبود).

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

۸. پرس‌وجویی بنویسید تا اطلاعات رانندگانی که فقط یک سفارش را انجام داده‌اند، بازیابی کند.

۹. پرس‌وجویی بنویسید تا اطلاعات رانندگانی که حداقل دو سفارش را انجام داده‌اند، بازیابی کند.

۱۰. پرس‌وجویی بنویسید تا ضرب دکارتی جداول ماشین‌ها (Cars) و رانندگان (Drivers) را انجام داده، بازیابی کند.

۱۱. پرس‌وجویی بنویسید تا نام، نام خانوادگی رانندگان و نام ماشین‌های مربوط به آن‌ها را بازیابی کند.

۱۲. پرس‌وجویی بنویسید تا نام، نام خانوادگی رانندگان و نام ماشین‌های آن‌ها 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. پرس‌وجویی بنویسید تا نام، نام خانوادگی مشتریان و شماره سفارش‌های 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.

۱٤. پرس‌وجویی بنویسید تا نام و نام خانوادگی رانندگان و مجموع کرایه‌هایی که دریافت کرده‌اند را بازیابی کند.

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

۱٦. پرس‌وجویی بنویسید تا اطلاعات رانندگانی را بازیابی کند که واحد ماشین آن‌ها "تن" است.

۱۱۷ – ۲. تمرین‌ها

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

۲. پرس جویی بنویسید تا نام کل، نام معین و نام تفضیلی را نمایش دهد.

۳. پرس‌وجویی بنویسید تا نام گروه حساب و نام کل حساب‌ها را نمایش دهد.

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

۵. پرس‌وجویی بنویسید تا اسنادی را نمایش دهد که در آن اسناد یکی از اقلام بدهکار یا بستانکار بالای ۲۰۰۰۰۰۰۰۰ ریال باشد.

٦. پرس‌وجویی بنویسید تا اسنادی را بازیابی کند که در نام تفضیلی آن‌ها، علی‌الحساب وجود داشته باشد.

۷. پرس‌وجویی بنویسید تا کلیه حساب‌های کل، معین، تفضیلی را بازیابی کند.

۸. پرس‌وجویی بنویسید تا که حساب‌های معین گروه حساب دارایی‌ها را بازیابی کند.

۹. پرس‌وجویی بنویسید تا سند خاصی را بازیابی کند.

۱۰. پرس‌وجویی 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
علی‌الحساب یا پرداختی	۱۰۰۰۱	۱۰۰۰۰	۱۵۰
پیش پرداخت مالیات	۱۰۰۰۱	۱۰۰۰۰	۱۵۰
...	...	...	...
علی‌الحساب خرید زمین	۲۲۲۰۰	۲۲۰۰۰	۱۵۰
علی‌الحساب 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

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

جدول ۱۱ - ۲ اطلاعات جدول کل.

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

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 بازیابی کند.

۱۲. پرس‌وجویی بنویسید تا سندی را بازیابی 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 بازیابی نماید.

۱٤. پرس‌وجویی بنویسید تا کد معین‌هایی را بازیابی کند که هنوز در ردیف‌های اسناد استفاده نشده‌اند (با عملگر NOT IN).

۱۵. پرس‌وجویی بنویسید تا نام معین و تفضیلی‌هایی که در ردیف اسناد استفاده شده‌اند را بازیابی کند.

۱۶. پرس‌وجوی بنویسید تا نام کل‌هایی که در ردیف اسناد استفاده نشده‌اند را بازیابی کند (با عملگر NOT EXISTS).

۱۷. پرس‌وجویی بنویسید تا نام معین‌هایی که هنوز در ردیف‌های اسناد استفاده نشده‌اند، را بازیابی کند (با عملگر EXCEPT).

۱۸. پرس‌وجویی 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 می‌کند. اطلاعاتی که در دفتر معین ظاهر می‌شوند، عبارت‌اند از: ردیف، شماره سند، تاریخ، کد کل، نام کل، کد معین، نام معین، مبلغ بدهکاری و مبلغ بستانکار).

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

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

۲۱. پرس‌وجویی بنویسید تا شماره و نام کل‌هایی را بازیابی کند که مجموع مبلغ بدهکار یا بستانکار آن‌ها در جدول ریز اسناد دقیقاً برابر ۱۰۰۰۰۰۰۰۰۰ ریال باشد.

۲۲. پرس‌وجویی بنویسید تا پیوند بیرونی چپ بین جدول گروه حساب و جدول کل را با عملگر > نمایش دهد.

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

۲۴. پرس‌وجویی بنویسید تا پیوند بیرونی کامل بین جداول گروه حساب و کل را با عملگر > نمایش دهد.

۲۵. پرس‌وجویی بنویسید تا اطلاعات تفضیلی‌هایی را بازیابی کند که نام تفضیلی آن‌ها با عبارت "مشترکین" شروع شده باشد.

۲۶. پرس‌وجویی بنویسید که اطلاعات تفضیلی‌هایی را بازیابی کند که نام تفضیلی آن‌ها با عبارت "ثابت" خاتمه یابد.

۲۷. پرس‌وجویی بنویسید تا شماره اسنادی را بازیابی کند که بالانس هستند (مجموع مبلغ بدهکار و بستانکار آن‌ها یکی است).

۲۸. پرس‌وجویی بنویسید تا شماره کل‌هایی را بازیابی کند که مجموع بدهکار آن‌ها بیشتر از مجموع بستانکار است (اختلاف بین بدهکار و بستانکار را نیز نمایش دهد).

۲۹. پرس‌وجویی بنویسید تا تمام تفضیلی که از یک شماره سند خاص تا شماره سند دیگر گردش دارند را بازیابی کند.

فصل ۳ 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[۸] 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 نماید. 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 جداول.

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 ثبت‌شده کاربران مختلف نگهداری می‌شود. فرض کنید کاربر خاصی بخواهد فاکتورهای ثبت‌شده 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

---

۸. View ۲. Stored Procedures ۳. Functions ۴. Vitrtual table

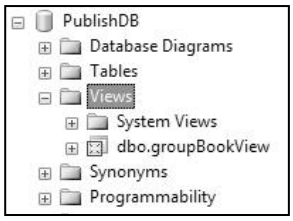
رکوردهای فاکتورهای خودش را بازیابی نماید. اگر تعداد دفعات اجرای این دستور زیاد باشد (به دفعات زیاد بخواهد این کار را انجام دهد)، تایپ دستور نه تنها کسل کننده است، بلکه زمان‌بر نیز خواهد بود. برای رفع این مشکل می‌توان برای هر یک از کاربران نام‌برده، دیدی ایجاد کرد تا کاربران فقط بتوانند اطلاعات خودشان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

```
CREATE VIEW view_name(fields_list) AS دستور SQL
```

در این ساختار، view_name، نام دیدی است که می‌خواهید ایجاد کنید، fields_list، لیست فیلدهای دید را تعیین می‌کند.

مثال ۱ - ۳. پرس‌وجویی که دیدی به نام groupBookView ایجاد می‌کند. این دید نام و نام گروه کتاب را بازیابی می‌کند.

```
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 می‌شوند. 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. پرس‌وجویی که دید groupBookView را اجرا می‌کند.

```
SELECT * FROM groupBookView
```

	Title	groupName
1	حل مسائل C++	برنامه نويسي
2	حل مسائل C#	برنامه نويسي
3	امنيت شبكه	شبكه رايانه اي
4	اصول طراحي پايگاه داده	پايگاه داده
5	اصول طراحي سيستم هاي شي گرا	برنامه نويسي
6	مديريت استراتژيک	فناوري اطلاعات
7	رهيافت C++	برنامه نويسي

در هنگام ایجاد دید باید به نکات زیر دقت کنید:

۱. 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:

```
DECLARE @نام متغیر نوع متغیر [= پیشفرض مقدار]
```

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[۱۰] 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 ذخیره‌شده 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.

٤. 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 ٤. 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 می‌کنند 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 سرویس‌گیرنده 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 وردی/ 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 اشیای جدیدی را ایجاد کرد، از آن‌ها استفاده نمود. اشیایی که رویه ذخیره شده می‌تواند ایجاد کند، دو نوع‌اند:

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

۲. **اشیا موقت دائمی**، اشیایی 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 پیش‌فرض صفر برگشت داده می‌شود که بیان می‌کند رویه ذخیره شده بدون خطا اجرا شد.
- در رویه ذخیره شده نمی‌توانید از دستورات CREATE RULE، CREATE VIEW، CREATE PROCEDURE، CREATE DEFAULT و CREATE TRIGGER 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 _sp 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
٤. رویه‌های ذخیره شده بازیابی 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 (SELECT نام فیلد کلید FROM نام جدول)
BEGIN
        SET @پارامتر خروجی = 'مقدار خطا'
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 (SELECT فیلد کلید اولیه متناظر کلید خارجی
          FROM جدول ارتباط)
BEGIN
        SET @پارامتر خروجی = 'مقدار خطا'
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(٤۰),
 @Page int, @Price decimal(۱٥،۰),
 @editNo int, @printNo int, @groupID varchar(٦),
 @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 ='٤'
    RETURN ٤
 END
ELSE IF @printNo < ۰
 BEGIN
    SET @errorCode ='٥'
    RETURN ٥
 END
ELSE IF @editNo  < ۰
 BEGIN
    SET @errorCode ='٦'
    RETURN ٦
 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 می‌کنند. در این رویه ذخیره شده اعمال زیر انجام می‌شود:

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 شده‌اند).

۳. در سطر ۵، پارامتر errorCode@ از نوع ورودی / خروجی (output) تعریف شده است که کد خطای رخ داده شده را نگهداری می‌کند. اگر خطای رخ ندهد، این پارامتر مقدار '۰' را در خودش نگهداری می‌کند.

٤. شرط IF @ISBN چک می‌کند که اگر شابک 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 می‌کند که اگر کد گروه کتاب در جدول GroupBook وجود ندارد، مقدار پارامتر errorCode@ برابر '۲' شود و رویه ذخیره شده ۲ را برگرداند.

٦. شرط ELSE IF @Page < ۰ بررسی می‌کند 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 می‌کند 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 می‌کند 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 می‌کند 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 شرط‌های بیان شده برقرار نباشد، رکورد با دستور INSERT به جدول Books اضافه می‌شود و مقدار پارامتر errorCode@ برابر '۰' خواهد شد. در پایان، رویه ذخیره شده ۰ را برمی‌گرداند.

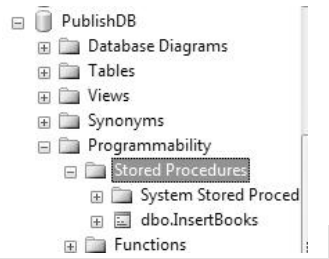
اگر این دستورات را اجرا کنید، رویه ذخیره شده InsertBooks در پوشه Stored ایجاد خواهد شد قرار خواهد گرفت. برای PublishDB بانک اطلاعاتی Programmability موجود در پوشه Procedures دیدن این رویه ذخیره شده، مراحل زیر را انجام دهید:

۱. پوشه Databases را کلیک مضاعف کنید تا بانک‌های اطلاعاتی موجود در سرویس‌دهنده سیستم‌تان 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 برنامه‌ای بانک اطلاعاتی PublishDB را مشاهده نمایید.

٤. پوشه Stored Procedures را کلیک مضاعف کرده تا پوشه رویه‌های 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 می‌توانند برگردانند، به دو دسته زیر تقسیم می‌شوند:

۱. توابعی که یک مقدار برمی‌گردانند، این توابع فقط یک مقدار را برمی‌گردانند. اطلاعاتی که این توابع برمی‌گردانند، **اسکالر (غیر جدولی)** نام دارند. این توابع نمی‌توانند داده‌های نوع timestamp، انواع داده‌های غیر اسکالر مانند Table، image، ntext، Text، 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 ایـــن ســـاختار، schema_name، نـــام شِـــما و function_name، نـــام تـــابع را تعیـــین مـــی‌کنـــد، parameter_name ، نام پارامتر، type_schema_name، نوع شِما و parameter_data_type، نوع پـارامتر را مشخص می‌کند. پارامترهای default و READONLY به ترتیـب مقـدار پـیش‌فـرض و فقـط خوانـدنی بـودن پارامتر را تعیین می‌کنند. return_data_type، نوع داده‌ای که تـابع برمـی‌گردانـد، مـی‌باشـد. function_body بدنه (دستورات) تابع را تعیین می‌کند و scalar_expression، عبارتی است که نتیجـه آن یـک مقـدار اسـکالر می‌باشد و دستور RETURN آن را برمی‌گرداند.

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

۱. این توابع فقط یک مقدار را برگشت می‌دهند. پس باید حـداقل یـک دسـتور RETURN در بدنـه آن‌هـا 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 نـام خـانوادگی مـدیر انتشارات را به عنوان پارامتر دریافت کرده، نام انتشارات را برگشت دهد.

```
CREATE FUNCTION getPubName(@bFname varchar(۱۵), @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 تـا نـام کتـاب را بـه عنـوان 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 برمی‌گردانند

از آنجایی که این توابع یک جدول را برمی‌گردانند، از طریق دستور SELECT می‌توان این توابع را فراخوانی کرد. نمونه‌های از فراخوانی این توابع را در پرس‌وجوهای بعدی خواهید دید.

مثال ۴۳- ۳. پرس‌وجویی که با فراخوانی تابع ()getBookAut، نام و قیمت کتاب‌های تألیف شده توسط رمضان‌عباس‌نژاد را نمایش می‌دهد.

```
SELECT * FROM getBookAut('رمضان' ,'عباسنژاد')
```

	Title	Price
1	حل مسائل C++	57000
2	حل مسائل C#	62000
3	امنیت شبکه	65000
4	اصول طراحي پايگاه داده	73000
5	اصول طراحي سيستم هاي شي گرا	130000
6	زهیافت C++	43000

همان‌طور که در این فراخوانی می‌بینید، در توابع خطی) الزامی dboکه جدولی را برمی‌گردانند، ذکر نام مالک (به جای نام جدول ()getBookAutنیست (در این جا نام تابع قرار گرفته است).SELECTیا دید در دستور

نکته: البته در هنگام فراخوانی تابع خطی که جدولی را می‌توان فیلدهای خاصی را بازیابی کرد (پرس‌وجویی بعدی را ببینید).*برمی‌گرداند، به جای علامت

مثال ۲۷. پرس‌وجویی که تابع ()MerchandiseGroup را فراخوانی می‌کند.

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

با اجرای این دستور خطای زیر ظاهر می‌شود:

```
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 پیش‌فرض باشد، برای فراخوانی تابع اگر نخواهید مقداری برای پارامتر پیش‌فرض ارسال کنید، حتماً باید کلمه DEFAULT را ذکر نمایید. پرس‌وجویی بعدی این موضوع را بیان می‌کند.

مثال ۲۸. پرس‌وجویی که تابع ()MerchandiseGroup را فراخوانی می‌کند تا اطلاعات کالاهای نوشیدنی را بازیابی کند.

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

همان‌طور که در فراخوانی این تابع می‌بینید، کلمه DEFAULT به جای نام پارامتر ذکر شده است. با اجرای این دستور خروجی زیر ظاهر می‌شود:

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

همان‌طور که در این خروجی می‌بینید، فقط کالاهای نوشیدنی نمایش داده شده‌اند. چون، مقدار پیش‌فرض ، نوشیدنی بود.groupName@پارامتر

مثال ۲۹. پرس‌وجویی که با فراخوانی تابع ()MerchandiseGroup لیست کالاهایی که نام گروه آن‌ها حبوبات باشد را برمی‌گرداند.

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

با اجرای این دستور خروجی زیر ظاهر می‌گردد:

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

همان‌طور که در این خروجی مشاهده می‌گردد، فقط کالاهایی که در گروه حبوبات قرار دارند، نمایش داده شده‌اند.

مثال ۳۰. پرس‌وجویی که تابعی به نام ()GetPrice ایجاد می‌کند 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. پرس‌وجویی بنویسید که دیدی به نام DriversView ایجاد کند تا اطلاعات رانندگانی را بازیابی نماید که برای شهرهای آمل یا بابل سفارش حمل کردند.

۲. پرس‌وجویی بنویسید که دید DriversView را اجرا کند.

۳. پرس‌وجویی بنویسید که با استفاده از دید DriversView، نام و نام خانوادگی رانندگانی که سفارشی برای شهرهای بابل یا آمل حمل نموده‌اند را بازیابی کند.

٤. پرس‌وجویی بنویسید که دید DriversView را طوری تغییر دهد تا اطلاعات رانندگانی که برای شهر بابل و شیراز سفارشی حمل کردند، بازیابی کند.

۵. پرس‌وجویی بنویسید که دیدی به نام CusOrderView ایجاد کند تا اطلاعات مشتریانی که حداقل یک سفارش داشته‌اند را بازیابی کند.

٦. پرس‌وجویی بنویسید که دید CusOrderView را طوری تغییر دهد تا اطلاعات مشتریانی که اصلاً سفارشی نداشته‌اند، بازیابی کند.

۷. پرس‌وجویی بنویسید که دیدی به نام DrvCountOrderView ایجاد کند تا اطلاعات رانندگانی که بیش از ۲ سفارش انجام داده‌اند، بازیابی کند.

۸. پرس‌وجویی بنویسید که دیدی به نام sumFreightCusView ایجاد کند تا نام، نام خانوادگی و مجموع کرایه پرداخت شده توسط مشتریان را بازیابی کند.

۹. پرس‌وجویی بنویسید که دیدهای DrvCountOrderView و CusOrderView را حذف کند.

۱۰. پرس‌وجویی بنویسید که رویه ذخیره شده‌ای 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. پرس‌وجویی بنویسید که رویه ذخیره شده InsertCars را اجرا کرده و پیغام "شماره ماشین تکراری است" را نمایش دهد.

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

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

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

۱۵. پرس‌وجویی 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.

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

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

۲۱. پرس‌وجویی بنویسید که با دستور EXEC تابع ()CountDrvOrder را فراخوانی کند تا تعداد سفارش-های انجام شده توسط راننده رضا رضازاده را نمایش دهد.

۲۲. پرس‌وجویی 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. پرس‌وجویی بنویسید که تابع ()drvName را فراخوانی کند. به خروجی دقت کنید، رکوردهای تکراری می‌بینید.

۲۴. پرس‌وجویی بنویسید که تابع drvName() را طوری تغییر دهد تا نام و نام خانوادگی‌های تکراری را فقط یک بار برگرداند.

۲۵. پرس‌وجویی بنویسید که تابع drvName() را فراخوانی کند. به خروجی دقت نمایید، رکوردهای تکراری فقط یک بار ظاهر گردیدند.

۲۶. پرس‌وجویی بنویسید که تابعی به نام CusName() ایجاد کند تا نام و نام خانوادگی مشتریانی را بازیابی کند که سفارش داشته‌اند.

۲۷. پرس‌وجویی بنویسید که با استفاده از فراخوانی تابع CusName() تعداد مشتریانی که سفارش داشته‌اند را نمایش دهد.

۲۸. پرس‌وجویی بنویسید که با استفاده از فراخوانی تابع CusName()، مشتریانی که نام آن‌ها رضا است و سفارش داشته‌اند را بازیابی کند.

۲۹. پرس‌وجویی بنویسید که تابعی به نام keyReturns() ایجاد کند تا نام جدول را به عنوان پارامتر دریافت کرده، محدودیت‌هایی تعریف شده در آن جدول را برگرداند.

۳۰. پرس‌وجویی بنویسید تا تابع keyReturns() را برای جدول Orders فراخوانی کند.

۳۱. پرس‌وجویی بنویسید تا تابع keyReturns() را طوری فراخوانی کند تا کلیدهای خارجی جدول Orders را نمایش دهد.

۳۲. پرس‌وجویی بنویسید تا تابع keyReturns() را طوری فراخوانی کند تا کلید اولیه جدول Customers را برگرداند.

۳۳. پرس‌وجویی 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.

۱٤. 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).

۱٦. 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 تفضلی 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. پرس‌وجویی بنویسید که رویه ذخیره شده‌ای ایجاد کند تا جدول پروژه را حذف کند.

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

۲۴. پرس‌وجویی 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. پرس‌وجویی 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. پرس‌وجویی 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 پیش‌فرض صفر در نظر گرفته شود).

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

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

۳۰. پرس‌وجویی 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 پیش‌فرض این پارامتر 'اسناد موقت' است).

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

فصل ۴ مباحث پیشرفته در SQL Server

تاکنون با مراحل ایجاد بانک اطلاعاتی، ایجاد و دست‌کاری 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
٤. 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 اشیا، حساب کاربران، حقوق دست‌یابی و ایمنی کاربران و اطلاعات دیگر را نگهداری می‌کند. سیستم کاتالوگ توسط سیستم مدیریت بانک اطلاعاتی[۲] ایجاد و دست‌کاری می‌شود.

جدول ۱ - ٤ 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 ۲. 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 اشیا در جدول۱ – 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 (شِما 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 (جداول تعریف شده توسط کاربر)

۲. مقدار V (شیء از نوع دید است)

۳. مقدار P (شیء از نوع رویه ذخیره شده است)

٤. 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	RO
1	PublishDB	dbo	InsertBooks	PublishDB	dbo	InsertBooks	PU
2	PublishDB	dbo	GroupBookP	PublishDB	dbo	GroupBookP	PU
3	PublishDB	dbo	DeleteBooks	PublishDB	dbo	DeleteBooks	PU
4	PublishDB	dbo	UpdateBooks	PublishDB	dbo	UpdateBooks	PU

۲–۴. 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 (اشتباهاً).

٤. ویروس‌های رایانه‌ای، اطلاعات را تخریب کنند.

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

جدول ۳ - ٤ 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.
RETAINDAYS	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 پشتیبان گیری، نوار را از نوار خوان بیرون نمی‌آورد.
UNLOAD	پس از پشتیبان گیری، نوار را از نوار خوان بیرون می‌آورد.
NOREWIND	پس از پشتیبان گیری، نوار را به عقب برنمی‌گرداند.
REWIND	پس از پشتیبان گیری، نوار را به عقب برمی‌گرداند.
RESTART	اگر پشتیبان گیری قطع شده باشد، پشتیبان گیری را از مکانی که قطع شده است، ادامه می‌دهد.
STATS	فاصله زمانی را تعیین می‌کند که باید درصد پیشرفت کار نمایش داده شود.

[۱۲]- Header

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

روش‌های متعددی برای پشتیبان گیری از بانک اطلاعات SQL Serverوجود دارد. کامل‌ترین روش پشتیبان گیری، استفاده از دستور BACKUP DATABASE است. زیرا، این دستور، از بانک اطلاعات، فایل‌های کارنامه[۱۳] و گروه فایل‌ها (Filegroups) پشتیبان می‌گیرد. این دستور، به صورت زیر به کار می‌رود:

```
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 } ]
  [ [ , ] { NOREWIND | REWIND } ]
  [ [ , ] { NOUNLOAD | UNLOAD } ]
  [ [ , ] RESTART ]
  [ [ , ] STATS [ = percentage ] ]
  [ [ , ] COPY_ONLY
```

پارامترهای این دستور در جدول ۳ – ۴ آمده است.

مثال ۱۳– ۴. پرس‌وجویی که از بانک اطلاعاتی PublishDB پشتیبان گرفته، در پوشه C:\SQLBACKUP قرار می‌دهد (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 رویه‌های ذخیره شده هستند که کاربردهای از قبیل حافظت از داده‌ها در برابر تغییرات، پنهان‌سازی قوانین کاری، ایجاد تغییرات آبشاری نظیر تغییرات به روز رسانی و حذف آبشاری، حفظ تمامیت داده‌ای و ارجاعی و جلوگیری از ایجاد ناسازگاری در داده را دارند. نمونه‌های از این کاربردها با مثال در زیر آمده‌اند:

---

[۱۳]- Log

[۱۴].Triggers

را در نظر بگیرید. تریگرهای می‌توان برای جداول این بانک PublishDBبه 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 می‌نماید. این نوع حذف از جدول را **حذف آبشاری** می‌نامند.

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 ویژگی‌های تریگرها در زیر آمده‌اند:

۱. تریگرها فاقد پارامتر هستند، اما رویه‌های ذخیره شده می‌توانند پارامتر داشته باشند.

۲. تریگرها به طوری صریح فراخوانی نمی‌شوند. یعنی، تریگرها در هنگام دست‌کاری داده‌ها جدول فعال می‌گردند(به طور ضمنی فراخوانی می‌گردند)، ولی رویه‌های ذخیره شده برای اجرا باید به طور صریح فراخوانی شوند.

۳. یک تریگر، فقط بر روی یک جدول یا دید ایجاد می‌گردد. اما، رویه‌های ذخیره شده می‌توانند بر روی چند جدول کار کنند.

٤. اگر جدولی را حذف کنید، تریگرهای آن به طور خودکار حذف خواهند شد. ولی، با حذف جدول رویه‌های ذخیره شده‌ای که از آن جدول استفاده می‌کنند، حذف نمی‌شوند.

٥. تریگرها انواع مختلف دارند که فقط تریگرهای Instead of را می‌توانید بر روی دیدها ایجاد کنید.

٦. تریگرها می‌توانند به صورت تودرتو فراخوانی شوند. یعنی، یک تریگر می‌تواند تریگر دیگری را فراخوانی کرده و باعث فعال شدن آن شود.

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

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

۹. اگر هر یک از محدودیت‌های (قیـود) NULL، PRIMARY KEY، CHECK CONSTRAINTS و غیره نقض شوند، هیچ تریگری بر روی آن جدول اجرا نمی‌گردد. چون، تریگرها پـس از انجـام تغییـرات بـر 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 بـه دیـدها یـا جـداول موقـت، رجوع داشته باشند.

۱۱. چون دستورات تریگرها نمی‌توانند مجموعه‌ای از رکوردها را برگرداند، بنـابراین نمـی‌تـوان از عملگـر 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 دیگـر تقسـیم شـوند کـه عبارت‌اند از : ۱. **تریگرهای نوع AFTER**، پس از اجرای دستورات دست‌کاری کننده فعال می‌شوند. این تریگرها را نمی‌توان روی دیدها ایجاد نمـود. ۲. **تریگرهـای نـوع INSTEADOF**، بـه جـای دسـتورات دست‌کاری کننده جداول و دیدها فعال می‌شوند.

نوع دستور فعال‌کننده تریگـر، ایـن تریگرهـا مـی‌تواننـد بـه سـه دسـته تقسـیم شـوند کـه عبـارت‌انـد 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 مـی‌نمایـد. ۲. **trigger_name**، نـام تریگـر اسـت.

۳. **table**، نام جدلی است که تریگر بر روی آن ایجاد می‌شود. ۴. **view**، نام دیدی است که تریگری بـر روی آن ایجـاد مـی‌گـردد. ۵. **AFTER**، نـوع تریگـر را AFTER تعیـین مـی‌کنـد.

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

دستورات زیر را نمی‌توانید در بدنه تریگر استفاده کنید:

۱. ALTER DATABASE	۲. CREATE DATABASE
۳. DROP DATABASE	٤. RESTORE DATABASE
٥. LOAD DATABASE	٦. 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 ویژگی‌های زیر را دارند:

۱. تریگرها اطلاعات خودشان را در دو جدول Deleted و Inserted درج می‌کنند. این دو جدول در حافظه اصلی و به صورت موقت ایجاد می‌گردند که فقط در بدنه تریگر به آن‌ها می‌توان مراجعه نمود. یعنی، خارج از تریگر نمی‌توان به این جداول دسترسی داشت تا اطلاعات آن‌ها را بازیابی کرد.

۲. در تریگرهای نوع AFTER، وقتی دستور فعال کننده تریگر، رکورد یا رکوردهای را به جدول پایه اضافه می‌نماید، 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 می‌نماید، این رکوردها از جدول Deleted حذف خواهند کرد.

٤. در تریگرهای نوع AFTER، وقتی فعال کننده تریگر رکورد یا رکوردهایی را در جدول پایه تغییر می‌دهد، رکوردهای قبل از تغییر را در جدول Deleted و رکوردهای بعد از تغییر را در جدول Inserted اضافه خواهد کرد.

٥. در تریگرهای نوع INSTEAD OF، دستور فعال کننده تریگر، رکوردها را فقط به جدول Deleted و Inserted اضافه می‌کند.

٦. اگر نوع تریگری در هنگام ایجاد ذکر نشود، AFTER در نظر گرفته می‌شود (البته ذکر کلمه FOR الزامی است).

۷. یک جدول حداکثر می‌تواند 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 می‌تواند 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 سطرهایی 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 ٤
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 بازیابی 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 ٤
۱ rows have been Deleted
(۱ row(s) affected)
```

مثال ۲۷-۴. پرس‌وجویی 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 ٥٤٧, Level ١٦, State ٠, Line ١
The DELETE statement conflicted with the REFERENCE constraint
    "FK__Books__groupID__٠٥١٩C٦AF". 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 تغییرکپی فوری وجود دارد. پس، این نوع کرسرها فقط خواندنی هستند. یعنی، فقط امکان پیمایش رکوردها را دارند.

۲. کرسر پویا، امکان تغییر برروی 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 CURSOR… DECLARE 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 بررکوردها ۵. بستن کرسر ۶. حذف کرسر

جدول ۵ - ٤ 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

همان‌طور 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

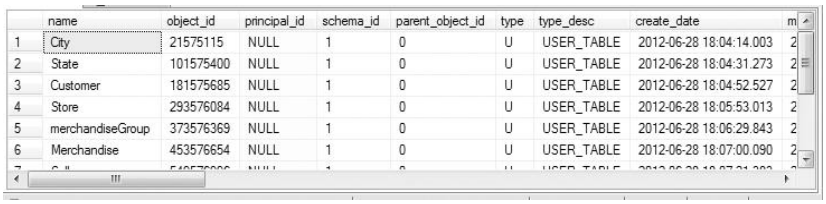
مثال ۳. پرس‌وجویی که لیست ستون‌های جدول City بانک اطلاعاتی Buy_Sell را بازیابی می‌کند.

```
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_NULLAE
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.

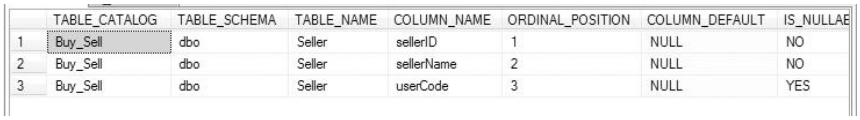
```
USE Buy_Sell
SELECT * FROM sys.tables
```



	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	m
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

مثال ۵. پرس‌وجویی که لیست ستون‌های جدول Sellers در بانک اطلاعاتی Buy_Sell را بازیابی می‌کند.

```
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_NULLAE
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

۵ – ۴. دستور کار آزمایشگاه

۱. پرس‌وجویی بنویسید که لیست فایل‌های بانک اطلاعاتی TransportationAgency را بازیابی کند.

۲. پرس‌وجویی بنویسید که لیست جداول بانک اطلاعاتی TransportationAgency را بازیابی کند.

۳. پرس‌وجویی بنویسید که لیست دیدهای بانک اطلاعاتی TransportationAgency را بازیابی کند.

٤. پرس‌وجویی بنویسید که لیست تمام پارامترهای بانک اطلاعاتی TransportationAgency را بازیابی کند.

۵. پرس‌وجویی بنویسید که لیست پارامترهای ورودی بانک اطلاعاتی TransportationAgency را بازیابی کند.

٦. پرس‌وجویی بنویسید که لیست توابع بانک اطلاعاتی TransportationAgency را بازیابی کند.

۷. پرس‌وجویی بنویسید که لیست رویه‌های ذخیره شده بانک اطلاعاتی TransportationAgency را بازیابی کند.

۸. پرس‌وجویی بنویسید که لیست بانک اطلاعاتی موجود در سرویس‌دهنده فعلی را بازیابی کند.

۹. پرس‌وجویی بنویسید که لیست ستون‌های جدول Cars مربوط به بانک اطلاعاتی TransportationAgency را بازیابی کند.

۱۰. پرس‌وجویی بنویسید که لیست محدودیت‌های بانک اطلاعاتی TransportationAgency را بازیابی کند.

۱۱. پرس‌وجویی بنویسید که لیست گروه فایل‌های بانک اطلاعاتی TransportationAgency را بازیابی کند.

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

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

۱٤. پرس‌وجویی بنویسید که لیست کاربران بانک اطلاعاتی TransportationAgency را بازیابی کند.

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

۱٦. پرس‌وجویی بنویسید که پشتیبان تهیه شده بانک اطلاعاتی TransportationAgency که برروی آن کلمه عبور mail_fan قرار دارد را بازیابی کند (البته فایل‌های داده و کارنامه آن را به مکان جدید منتقل نماید).

۱۷. پرس‌وجویی بنویسید که پشتیبان تهیه شده بانک اطلاعاتی TransportationAgency که برروی آن کلمه عبور mail_fan قرار دارد را با کلمه عبور mail_fan۱ می‌خواهد بازیابی کند، دستور را نوشته، پیغام خطا را مشاهده کنید.

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

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

۲۰. پرس‌وجویی بنویسید که پشتیبان TransportationAgency را به نام T۱ با فایل‌های داده t۱_data و کارنامه t۱_log بازیابی کند.

۲۱. پرس‌وجویی بنویسید تا نام تمام بانک‌های اطلاعاتی موجود برروی رایانه‌تان را بازیابی کند.

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

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

۲٤. پرس‌وجویی 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. پرس‌وجویی بنویسید تا تریگر trInsertOrders را آزمایش کند.

۲٦. پرس‌وجویی بنویسید تا تریگر trInsertOrders را طوری تغییر دهد که از ورود عدد منفی برای فیلد Freight جلوگیری کند.

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

۲۸. پرس‌وجویی بنویسید تا تریگر trDeleteDrivers را آزمایش کند.

۲۹. پرس‌وجویی 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. پرس‌وجویی بنویسید تا تریگر trModifyDrivers را آزمایش کند.

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

۳۲. پرس‌وجویی بنویسید تا کرسری ایجاد کند که نام و نام‌خانوادگی جدول رانندگان (Derivers) را بازیابی کند.

۳۳. پرس‌وجویی بنویسید تا با استفاده از کرسری نام و نام‌خانوادگی اولین وآخرین رکورد جدول رانندگان (Derivers) را بازیابی کند.

۳٤. پرس‌وجویی بنویسید تا کرسری ایجاد کند که شماره رکورد و نام ماشین‌ها را نمایش دهد.

۳۵. پرس‌وجویی بنویسیدکه کرسری ایجاد کند تا آن کرسر نام و نام‌خانوادگی مشتریان بابلی را نمایش دهد.

۳٦. پرس‌وجویی بنویسید که کرسری ایجاد کند تا این کرسر مجموع کرایه‌های راننده‌ای با نام خانوادگی امیدی را بازیابی کند.

۶-۴. پاسخ دستور کار آزمایشگاه

۷ - ۴. تمرین‌ها

۱. پرس‌وجویی بنویسید تا لیست کاربران بانک اطلاعاتی حسابداری را بازیابی کند.

۲. پرس‌وجویی بنویسید تا لیست جداول بانک اطلاعاتی حسابداری را بازیابی کند.

۳. پرس‌وجویی بنویسید تا لیست دیدهای بانک اطلاعاتی حسابداری را بازیابی کند.

٤. پرس‌وجویی بنویسید تا لیست ستون‌های جدول کل از سیستم حسابداری را بازیابی کند.

۵. پرس‌وجویی بنویسید تا لیست کلیدهای خارجی جدول تفضیلی از سیستم حسابداری را بازیابی کند.

٦. پرس‌وجویی بنویسید تا لیست کلیدهای اولیه جدول تفضیلی از سیستم حسابداری را بازیابی کند.

۷. پرس‌وجویی بنویسید تا لیست گروه فایل‌های بانک اطلاعاتی حسابداری را بازیابی کند.

۸. پرس‌وجویی بنویسید تا لیست بانک‌های موجود در سرویس‌دهنده فعلی را بازیابی کند.

۹. پرس‌وجویی بنویسید تا لیست پارامترهای ورودی سیستم حسابداری را بازیابی کند.

۱۰. پرس‌وجویی بنویسید تا لیست توابع سیستم حسابداری را بازیابی کند.

۱۱. پرس‌وجویی بنویسید تا لیست رویه‌های ذخیره شده سیستم حسابداری را بازیابی کند.

۱۲. پرس‌وجویی بنویسید تا لیست پارامترهای خروجی بانک اطلاعاتی حسابداری را بازیابی کند.

۱۳. پرس‌وجویی بنویسید تا لیست فیلدهای جدول تفضیلی را بازیابی کند.

۱٤. پرس‌وجویی بنویسید تا انواع تعریف شده SQL را بازیابی کند.

۱۵. پرس‌وجویی بنویسید تا از بانک اطلاعاتی حسابداری پشتیبانی کامل گرفته، این پشتیبان را در مسیر C:\Backup رایانه‌تان قرار دهد.

۱٦. پرس‌وجویی بنویسید تا از تغییرات بانک اطلاعاتی حسابداری پشتیبانی گرفته، این پشتیبان را در مسیر C:\Backup با پسوند bck ذخیره کند.

۱۷. پرس‌وجویی بنویسید تا از بانک اطلاعاتی حسابداری پشتیبان تهیه کند. در هنگام تهیه پشتیبان فایل‌ها را فشرده نموده و بر روی آن Password قرار دهد.

۱۸. پرس‌وجویی بنویسید تا پشتیبان تهیه شده در سوال ۱۵ را در بانک اطلاعاتی به نام Account۱ بازیابی کند.

۱۹. پرس‌وجویی بنویسید تا پشتیبان تهیه شده در سوال ۱۷ را به نام بانک اطلاعاتی Account۲ بازیابی کند.

۲۰. پرس‌وجویی بنویسید که از بانک اطلاعاتی حسابداری پشتیبان کامل تهیه کند.

۲۱. پرس‌وجویی بنویسید که از تغییرات بانک اطلاعاتی حسابداری پشتیبان تهیه کند.

۲۲. پرس‌وجویی بنویسید که پشتیبان تهیه شده از بانک اطلاعاتی حسابداری را بازیابی کند.

۲۳. پرس‌وجویی بنویسید که از بانک اطلاعاتی حسابداری پشتیبان تهیه کرده، به طوری که برروی آن کلمه عبور ۱۰۹۸۷۶۵۴۳۲۱ را قرار دهد.

۲٤. پرس‌وجویی بنویسید که پشتیبان دارای کلمه عبور بانک اطلاعاتی حسابداری را بازیابی کند، به طوری که فایل‌های داده و کارنامه (log) آن را به پوشه جدیدی انتقال دهد.

۲۵. پرس‌وجویی بنویسید که تریگری برروی جدول کل ایجاد کند تا در هنگام حذف رکورد فعال شود و تعداد رکوردهای حذف شده را نمایش دهد.

۲٦. پرس‌وجویی 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. پرس‌وجویی بنویسید تا دو تریگر بیان شده سوالات ۲۵ و ۲۶ را آزمایش کند.

۲۸. پرس‌وجویی بنویسید که تریگری ایجاد کند تا در هنگام حذف رکوردی از جدول معین، بررسی نماید کد معین در جداول ریز سند و تفضیلی وجود ندارد. چنانچه کد معین در این جداول وجود داشت، از حذف رکورد در جدول معین جلوگیری کرده، پیغام مناسب نمایش دهد.

۲۹. پرس‌وجویی 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. پرس‌وجویی بنویسید تا تریگر ایجاد شده سوال ۲۹ را آزمایش کند.

۳۱. پرس‌وجویی 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. پرس‌وجویی بنویسید تا تریگر ایجاد شده سوال ۳۱ را آزمایش کند.

۳۳. پرس‌وجویی 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.

۳٤. پرس‌وجویی 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. پرس‌وجویی بنویسید که چند تریگر موجود را حذف کند.

۳٦. پرس‌وجویی 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. پرس‌وجویی 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. پرس‌وجویی بنویسید تا تریگر ایجاد شده در سوالات ۳۳ و ۳۶ را حذف کند.

۳۹. پرس‌وجویی بنویسید که تریگری ایجادکرده تا در هنگام درج، ویرایش و حذف رکورد از جدول تفضیلی، تعیین کند کدام یک از اعمال درج، ویرایش و حذف انجام شده است.

٤۰. پرس‌وجویی بنویسید تا با اجرای دستورات INSERT، UPDATE و DELETE بر روری جدول تفضیلی تریگر ایجاد شده سوال ۳۹ را آزمایش نماید.

٤۱. پرس‌وجویی بنویسید که کرسری ایجاد کند تا رکوردهای زوج جدول ریز سند را بازیابی کند.

٤۲. پرس‌وجویی بنویسید که کرسری ایجاد کند تا جمع مبلغ بدهکار و بستانکار جدول ریز سند را بازیابی کند.

٤۳. پرس‌وجویی بنویسید که کرسری ایجاد نماید تا اطلاعات رکورد شماره ۱۰ و ۲۰ جدول کل را بازیابی کند.

٤٤. پرس‌وجویی بنویسید که کرسری ایجاد نماید تا اطلاعات اولین و آخرین رکورد جدول سند را بازیابی کند.

٤۵. پرس‌وجویی بنویسید که کرسری ایجاد نماید تا اطلاعات کد و نام معین جدول معین را رکورد به رکورد بازیابی نماید.

٤٦. پرس‌وجویی بنویسید که کرسری ایجاد کند تا مانده جمع بدهکار و بستانکار سند شماره ۱۵ و مانده کل آن را نمایش دهد.

منابع

۱. عباس‌نژادورزی، رمضان، آموزش گام به گام بانک اطلاعاتی با #C (مرجع کامل)، بابل انتشارات فن-آوری نوین، ۱۳۸۸.

۲. جعفری، محمدرضا، "ایجاد بانک اطلاعاتی در SQL Server ۲۰۰۵" مشهد، واژگان خرد، ۱۳۸۷.

۳. Abraham Silberschatz, Henry Korth, and S. Sudarshan , "Database System Concepts (۶th Edition) ", MC Graw Hill, ۲۰۱۰

٤. 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.

٦. 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.