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 ۴۷۱۶۷-۷۳۴۴۸ تلفن: ۲۲۵۶۶۸۷-۰۱۱۱

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

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@yahoo.com** ارسال نمایند.

بابل، بهار ۹۲

مولفین

www.fanavarienovin.net

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؟

٦. 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[۱]. نرمال سازی موجب کاهش افزونگی داده، بی‌نظمی و کاهش داده‌های Null می‌شود.

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

- **جدول Books**، اطلاعات کتاب‌ها از قبیل شابک کتاب، عنوان ، تعداد صفحات و غیره را ذخیره می‌کند.
- **جدول Publishers**، کد ناشر، نام ناشر، شماره تلفن و غیره را برای ناشران نگهداری می‌کند.
- **جدول Authors**، اطلاعات مؤلفین از قبیل کد 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.
- **جدول GroupBook**، اطلاعات کد گروه کتاب و نام گروه کتاب را نگهداری می‌کند.

---

[۱] . 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 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[۱] (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.

---

[۱]. Properties (Attributes)

۶. فیلدهای کلید اولیه[۱] 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.

---

۱. 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.

۵ – ۱ – ۱. بهینه سازی طراحی

هنگامی که جداول، فیلدها و رابطه‌های بین جداول را تعریف کردید، می‌توانید طرح خود را مطالعه و بازنگری کنید تا هر نوع مشکل یا خطا در طراحی ساختار بانک اطلاعاتی را برطرف و تصحیح نمایید. پس از این که جداول را ایجاد کرده‌اید و رابطه‌های بین آن‌ها را تعریف نموده‌اید ، چند رکورد آزمایشی در جداول وارد کنید تا ببینید آیا بانک اطلاعاتی طراحی شده نیازها و خواسته‌های مورد نظرتان را پاسخ می‌دهد یا خیر. طرح‌های روشن و واضح از فرم‌ها و گزارش‌های مورد نظرتان رسم کرده، ببینید آیا نتایجی که مد نظرتان است، ارائه می‌دهد یا خیر؟ داده‌های تکراری (افزونگی داده) که در اثر طراحی ناکارآمد ایجاد شده‌اند، را پیدا کرده و حذف کنید. در این مرحله موارد زیر را نیز کنترل نمایید:

- آیا هیچ فیلدی را فراموش نکرده‌اید؟
- آیا اطلاعاتی وجود دارند که به آن‌ها نیاز داشته باشید؟ ولی در بانک اطلاعاتی جایی برای آن‌ها در نظر گرفته نباشید. اگر چنین است، آیا این اطلاعات به یکی از جداول موجود تعلق دارد؟ وگرنه، شاید لازم باشد، جدول جدیدی ایجاد نمایید.
- آیا برای هر جدول یک کلید اولیه مناسب انتخاب کرده‌اید؟ اگر کلید اولیه برای هر جدول انتخاب نموده‌اید، بررسی کنید که هیچ گاه نیاز به ورود مقدار تکراری یا NULL برای کلید اولیه نباشد.

چک کنید، آیا پس از ورود چند رکورد تستی در جدول ممکن است حالتی پیش آید که در آن بعضی از مقادیر درون فیلدهای یک جدول به صورت تکراری در رکوردهای مختلف ظاهر شوند؟ اگر چنین وضعیتی ایجاد شد، جدول مذکور را به دو جدول تقسیم کرده، بین آن‌ها یک رابطه یک به چند ایجاد کنید.

آیا جداولی در بانک اطلاعاتی طراحی شده وجود دارند که فیلدهای زیاد و تعداد محدودی رکورد داشته باشند؟ آیا فیلدهای خالی زیادی در این رکوردها وجود دارند؟ اگر چنین وضعیتی وجود دارد، باید ساختار جدول را طوری تغییر دهید که فیلدهای کمتر و رکوردهای بیشتری داشته باشد.

۲– ۱. بانک اطلاعاتی SQL Server

در دهه ۱۹۷۰، نسخه اصلی زبان پرس‌وجوی ساخت یافته (SQL)[۱] اولین بار توسط شرکت IBM تحت نام Seguel ارائه گردید. در سال ۱۹۸۶، موسسه استانداردسازی ملی آمریکا (ANSI) و سازمان بین‌المللی استانداردسازی (ISO)، استاندارد SQL تحت نام SQL–۸۶ را معرفی نمودند. در سال ۱۹۸۹ موسسه استانداردسازی ملی آمریکا نسخه SQL–۸۹ را ارائه نمود. سپس، نسخه‌های SQL–۹۲، SQL–۹۹ و SQL ۲۰۰۸ معرفی گردیدند. اکنون، جدیدترین نسخه SQL، SQL–۲۰۱۲ می‌باشد. زبان SQL از چندین بخش تشکیل شده است. در این فصل بخش‌های مختلف آن را می‌آموزیم.

۱–۲ – ۱. ورود به بانک اطلاعاتی SQL Server

یکی از بهترین روش‌های آموزش SQL، انجام دستورات آن در یک سیستم مدیریت بانک اطلاعات (DBMS) نمونه است. یکی از پرکاربردترین سیستم‌های بانک اطلاعاتی جهان SQL Server است. بنابراین، SQL Server را بر روی کامپیوترتان نصب کنید. اکنون برای استفاده از بانک اطلاعاتی باید وارد بانک اطلاعاتی SQL Server شوید. برای انجام این کار مراحل زیر را انجام دهید:

۱. گزینه Start / All Programs / Microsoft SQL Server ۲۰۱۲ / SQL Server Management Studio را اجرا کنید تا پنجره Connect to Server ظاهر گردد (شکل ۱– ۱).

شکل ۱– ۱ پنجره Connect to Server.

[۱]. Structured Query Language

۲. در بخش Server type، گزینه Database Engine، در بخش Server name، نام سرویس‌دهنده بانک اطلاعاتی و در بخش Authentication، گزینه Windows Authentication را انتخاب کنید و دکمه Connect را کلیک نمایید تا اولین صفحه SQL Server ظاهر گردد (شکل۲- ۱).

۲ – ۲–۱. تایپ و اجرای دستورات SQL

به دو روش می‌توان با بانک اطلاعاتی SQL Server کار کرد که عبارت‌اند از:

۱. روش Enterprise: با این روش می‌توان کارهای ایستا را بر روی بانک اطلاعاتی اجرا کرد.

۲. روش دستورات: با این روش می‌توان دستورات پویا را بر روی بانک اطلاعاتی اجرا نمود.

پس باید محیطی برای اجرای دستورات SQL داشته باشیم. برای تایپ و اجرای دستورات SQL مراحل زیر را انجام دهید:

۱. در صفحه اول SQL Server، دکمه New Query را کلیک کنید تا صفحه خالی برای تایپ دستورات SQL ظاهر شود (شکل۳- ۱).

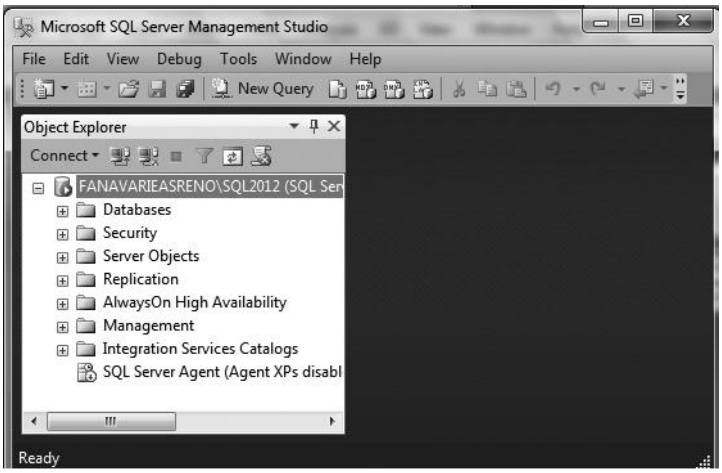
۲. در این محیط دستورات SQL را تایپ کنید.

۳. کلید F۵ را فشار دهید (گزینه Execute را کلیک کنید) تا دستور مورد نظر کامپایل گردد. چنانچه این دستور خطایی نداشته باشد، اجرا می‌شود.

۳–۱. ایجاد بانک اطلاعاتی

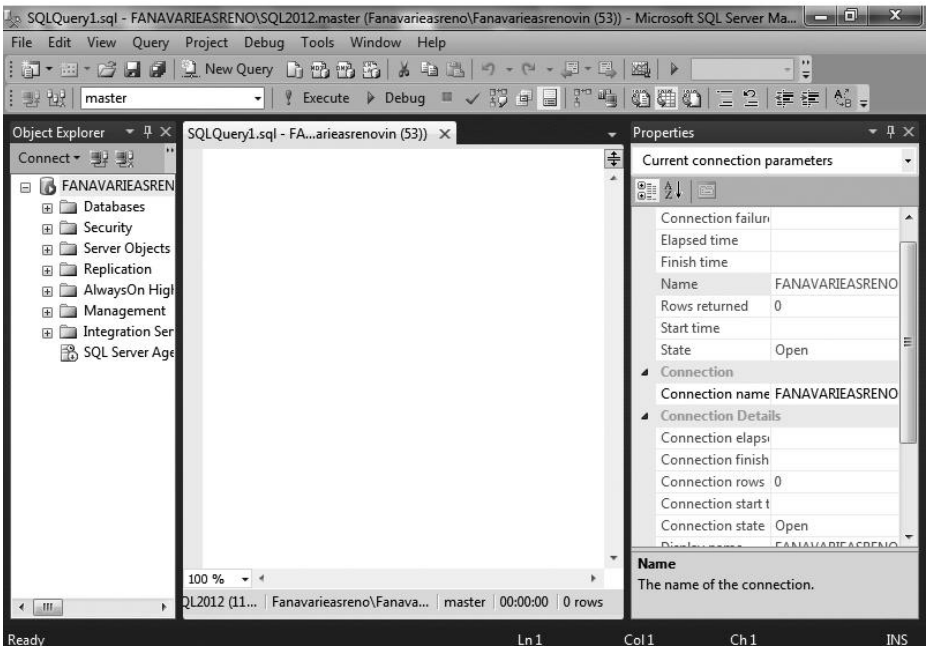
قبل از این که به ایجاد بانک اطلاعاتی بپردازیم، ابتدا فایل‌های تشکیل دهنده هر بانک اطلاعاتی را بیان خواهیم کرد. هر بانک اطلاعاتی می‌تواند از دو فایل تشکیل شود که عبارت‌اند از:

۱. فایل داده (Data File)، اطلاعات داده‌ای از قبیل اطلاعات جداول بانک اطلاعات را نگهداری می‌کند. پسوند این فایل معمولاً **mdf.**[۱] است.



شکل۲- ۱ صفحه اول بانک اطلاعاتی SQL Server.

[۱] .Meta Data File



شکل۳- ۱ محیط تایپ دستورات SQL.

۲. **فایل کارنامه (Log File)**، اطلاعاتی از قبیل چه کسی، چه تراکنشی را در چه زمانی انجام داده است، نگهداری می‌کند. پسوند این فایل معمولاً **ldf.**[۱] است. این فایل دو کاربرد مهم دارد که عبارت‌اند از:

۱. تا مین امنیت

۲. ترمیم (**recovery**) بانک اطلاعات در صورت خرابی آن.

برای ایجاد بانک اطلاعاتی باید این فایل‌ها را ایجاد کرده، ویژگی‌های آن‌ها را تعیین کنید. این فایل‌ها ویژگی‌های زیادی دارند، در این‌جا فقط ویژگی‌های مهم آن‌ها را می‌آموزیم. برخی از مهم‌ترین ویژگی‌های این فایل‌ها عبارت‌اند از:

- **NAME**، نام منطقی فایل را تعیین می‌کند که بانک اطلاعاتی فایل را با آن می‌شناسد.
- **FILENAME**، نام فایل در سطح سیستم عامل می‌باشد که سیستم عامل فایل را به این نام می‌شناسد.
- **SIZE**، اندازه فایل را تعیین می‌کند. بعد از اندازه فایل می‌تواند KB (کیلوبایت)، MB (مگابایت)، GB (گیگابایت) و TB (ترابایت) آورده شود. اگر ذکر نشود، MB در نظر گرفته می‌شود.
- **MAXSIZE**، حداکثر اندازه فایل را تعیین می‌کند. اگر مقدار این ویژگی UNLIMITED تعیین شود، اندازه فایل آن قدر اضافه می‌شود تا دیسک پر شود.
- **FILEGROWTH**، هنگامی که فایل پر شود (به اندازه ویژگی SIZE در فایل اطلاعات قرار گرفت)، حداکثر رشد (افزایش اندازه فایل) را تعیین می‌کند. اگر ذکر نشود، فایل آن‌قدر رشد خواهد کرد تا دیسک

---

[۱].Log Data File

پر گردد. مقدار رشد می‌تواند به درصد بیان شود. در این صورت میزان رشد فایل بر حسب اندازه فایل محاسبه می‌گردد. مقدار رشد فایل نمی‌تواند کمتر از ۶۴ کیلوبایت باشد.

FOR ATTACH، اگر بانک اطلاعاتی را با دستور Deattach جدا کرده باشید، این پارامتر برای اتصال مجدد فایل به کار می‌رود.

پارامتر COLLATE، برای تعیین زبان ذخیره سازی داده از طریق Collation_name به کار می‌رود.

پارامتر FILEGROUP، برای اضافه کردن گروه فایل به کار می‌رود.

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

```
CREATE DATABASE database_name
ON PRIMARY ( ویژگی‌های فایل داده)
LOG ON     (ویژگی‌های فایل کارنامه)
```

database_name، نام بانک اطلاعاتی است که می‌خواهید ایجاد کنید.

مثال ۱ – ۱. پرس‌وجویی که بانک اطلاعاتی به نام PublishDB را ایجاد می‌کند که نام فایل داده آن publish_data.mdf است. این فایل در پوشه ClassDatabase درایو C قرار می‌گیرد. اندازه این فایل ۱۰ مگابایت و حداکثر اندازه آن ۱۵۰ مگابایت می‌باشد. نام فایل کارنامه publish_log.ldf می‌باشد که در پوشه ClassDatabase درایو C به اندازه ۱۲ و حداکثر اندازه ۱۲۰ مگابایت ایجاد می‌شود.

```
CREATE DATABASE PublishDB
ON
(NAME=publish_data,FILENAME='C:\classDatabase\publish_data.mdf',
  SIZE = ۱۰, MAXSIZE = ۱۰۰MB
 )
LOG ON
(NAME=publish_log, FILENAME ='C:\classDatabase\publish_log.ldf',
  SIZE = ۱۲MB, MAXSIZE = ۱۲۰
 )
 COLLATE Arabic_CS_AS_KS_WS
 GO
```

همان‌طورکه می‌بینید برای COLLATE مقدار Arabic_CS_AS_KS_WS انتخاب گردید تا بتوان اطلاعات فارسی را در جدول بانک اطلاعات ذخیره نمود.

مثال ۲ – ۱. پرس‌وجویی که بانک اطلاعاتی Library با ویژگی‌های زیر ایجاد می‌کند:

دارای یک فایل داده به نام lib_data است. اندازه این فایل ۳۰۷۴KB می‌باشد که می‌تواند تا پر شدن دیسک رشد یابد و رشد فایل در هر مرحله ۱۰۲۴KB است. این فایل در درایو C پوشه ClassDatabase با نام lib_data.mdf قرار می‌گیرد.

یک فایل کارنامه به نام lib_log دارد که در درایو C پوشه ClassDatabase با نام lib_log.ldf ذخیره می‌شود. اندازه این فایل نیز ۳۰۷۲KB است. اما حداکثر اندازه آن ۲۰۴۸GB می‌باشد. درصد رشد این فایل ۱۰ درصد است.

```
CREATE DATABASE Library
 ON  PRIMARY
 (NAME=N'lib_data',FILENAME=N'c:\classdatabase\lib_data.mdf' ,
```

```
   SIZE = ۳۰۷۲KB , MAXSIZE = UNLIMITED, FILEGROWTH = ۱۰۲٤KB )
 LOG ON
  (NAME=N'lib_log',FILENAME=N'c:\classdatabase\lib_data.ldf' ,
    SIZE = ۳۰۷۲KB , MAXSIZE = ۲۰٤۸GB , FILEGROWTH = ۱۰٪)
GO
```

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**[۱]، 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**[٤]، 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).

۲. می‌توان به جای پشتیبان‌گیری[۵] و ترمیم[٦] 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)
```

---

۱. Primary File Group ۲. Primary Data File ۳. Pages ٤. User Defined File Group
۵. Backup ٦. Recovery

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 می کنند. بنابراین، اگر بانک اطلاعاتی را نیاز نداشته باشید باید آن را حذف کنید تا فضای اشغال شده توسط آن به سیستم عامل برگردد. برای حذف بانک اطلاعات دستور DROP DATABASE به صورت زیر استفاده می‌شود:

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

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

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

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

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

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

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

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

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

مثال ۷ – ۱. پرس‌وجویی که فایل my_data۳ در پوشه classDatabase درایو C با نام my_data۳.mdf را به گروه myFileGroup بانک اطلاعاتی myDatabase اضافه می‌کند. این فایل ٤MB است که می‌تواند تا ۵۰MB با گام ۵MB رشد کند.

```
ALTER DATABASE myDatabase
ADD FILE(NAME=my_data۳,FILENAME=N'c:\classdatabase

\my_data۳.mdf',SIZE=٤MB,MAXSIZE=۵۰MB,FILEGROWTH=۵MB)
 TO FILEGROUP myFileGroup
 GO
```

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

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

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

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

مثال ۱۰ – ۱. پرس‌وجویی که فایل my_data٤ در پوشه classDatabase درایو C با نام my_data٤.mdf را به گروه myFileGroup بانک اطلاعاتی myDatabase اضافه می‌نماید.

```
ALTER DATABASE myDatabase
ADD FILE(NAME=my_data٤,FILENAME=N'c:\classdatabase\
    my_data٤. mdf',SIZE=٤MB,MAXSIZE=۵۰MB,FILEGROWTH=۵MB)
 TO FILEGROUP myFileGroup
 GO
```

مثال ۱۱ – ۱. پرس‌وجویی که گروه فایل myFileGroup۱ را به myDatabase اضافه می‌کند.

```
ALTER DATABASE myDatabase
 ADD FILEGROUP myFileGroup۱
GO
```

مثال ۱۲ – ۱. پرس‌وجویی که گروه فایل myFileGroup را از بانک اطلاعات myDatabase حذف می‌کند.

```
ALTER DATABASE myDatabase
REMOVE FILE my_data۳
GO
ALTER DATABASE myDatabase
REMOVE FILE my_data٤
GO
ALTER DATABASE myDatabase
REMOVE FILEGROUP myFileGroup
GO
```

همان گونه که مشاهده می‌شود، برای حذف گروه فایل ابتدا باید فایل‌های موجود در آن گروه فایل را حذف نمود، سپس گروه فایل را حذف کرد.

مثال ۱۳ – ۱. پرس‌وجویی که فایل my_data۳ را به گروه فایل myFileGroup۱ از بانک اطلاعات myDatabase اضافه می‌کند.

```
ALTER DATABASE myDatabase
ADD  FILE(NAME=my_data۳,FILENAME=N'c:\classdatabase\
   my_data۳ .mdf',SIZE=٤MB,MAXSIZE=٥٠MB,FILEGROWTH=٥MB)
 TO FILEGROUP myFileGroup۱
 GO
```

مثال ۱۴ – ۱. پرس‌وجویی که گروه فایل myFileGroup۱ از بانک اطلاعاتی myDatabase را به گروه فایل پیش‌فرض تغییر می‌دهد.

```
ALTER DATABASE myDatabase
  MODIFY FILEGROUP myFileGroup۱ DEFAULT
GO
```

مثال ۱۵ – ۱. پرس‌وجویی که گروه فایل PRIMARY بانک اطلاعاتی myDatabase را به گروه فایل پیش‌فرض تغییر می‌دهد.

```
ALTER DATABASE MyDatabase
  MODIFY FILEGROUP [PRIMARY] DEFAULT
GO
```

دقت کنید که PRIMARY باید بین [ ] یا " " قرار گیرد.

مثال ۱۶ – ۱. پرس‌وجویی که فایل my_log۱ با نام my_log۱.ldf درایو C پوشه classDatabase را به بانک اطلاعاتی myDatabase اضافه می‌کند (یک فایل کارنامه به بانک اطلاعاتی اضافه می‌نماید).

```
ALTER DATABASE myDatabase
ADD LOG FILE(NAME=my_log۱,FILENAME='c:\classDatabase
    \my_log۱.ldf',SIZE=٤MB,MAXSIZE=۲۰MB,FILEGROWTH=٤MB)
GO
```

مثال ۱۷ – ۱. پرس‌وجویی که بانک اطلاعاتی myDatabase را به نام myDatabase۱ تغییر نام می‌دهد.

```
ALTER DATABASE myDatabase
```

```
MODIFY NAME = myDatabase۱
GO
```

مثال ۱۸ – ۱. پرس‌وجویی که بانک اطلاعاتی myDatabase۱ را حذف خواهد کرد. یعنی، تمام فایل‌های داده و کارنامه مربوط به این بانک موجود در پوشه ClassDatabase درایو C را حذف خواهد کرد.

```
DROP DATABASE myDatabase۱
```

نکته: در هنگام اجرای دستورات ALTER DATABASE و DROP DATABASE، اگر بانک اطلاعاتی که می‌خواهید خواص آن را تغییر دهید یا آن را حذف نمایید، موجود نباشد، بانک اطلاعاتی SQL Server پیغام خطا صادر می‌کند. ولی، اگر در هنگام اجرای دستور CREATE DATADBASE، چنانچه بانک اطلاعاتی که می‌خواهید ایجاد کنید از قبل موجود باشد، پیغام خطا ظاهر می‌گردد.

۴–۱. ایجاد و تغییر ساختار جدول

همان‌طور که می‌دانید، جدول از مجموعه‌ای از رکوردها تشکیل می‌شود و رکورد نیز از مجموعه‌ای از فیلدها تشکیل می‌گردد و فیلدها نیز حاوی تعدادی خواص هستند. بنابراین، قبل از این که به ایجاد جدول بپردازیم، باید خواص فیلدها را بررسی کنیم. این خواص در زیر آمده‌اند:

- **نام فیلد:** هر فیلد در جدول یک نام **یکتا**[۱] دارد که از قانون نام‌گذاری شناسه‌ها در بانک اطلاعات پیروی می‌کند.
- **نوع فیلد:** برای هر فیلد باید تعیین شود، اولاً چه نوع داده‌ای را ذخیره می‌کند، ثانیاً تعداد بایت‌هایی که ذخیره می‌نماید، چقدر است. انواع داده‌ها و مقدار فضایی که هر یک از انواع در بانک اطلاعات نیاز دارند، در جدول ۲– ۱ آمده است.
- **محدودیت‌های فیلد:** این ویژگی تعیین می‌کند هر فیلد چه محدودیت‌هایی (قیدهای) دارد. برخی از این محدودیت‌ها در زیر آمده‌اند:

۱. محدودیت کلید اولیه[۲]: فیلدی کلید اولیه است که دارای شرایط زیر باشد (مانند شماره کارمندی، شماره دانشجویی، شماره وام، شماره مشتری، شماره درس، کد گروه و کد شهر):

- مقدار این فیلد تکراری نباشد.
- اطلاعات رکوردها بر اساس این فیلد مرتب باشند.
- مقدار این فیلد نمی‌تواند **تهی**[۳] باشد.

مثال ۱۹– ۱. دستوری که فیلد کد شهر را به عنوان کلید اولیه جدول City تعریف می‌کند.

```
PRIMARY KEY (CityID)
```

مثال ۲۰ – ۱. دستوری که فیلد کد مؤلف را کلید اولیه معرفی می‌کند.

---

۱. Unique ۲. Primary Key ۳. Null

```
PRIMARY KEY (atID)
```

مثال ۲۱ - ۱. دستوری که فیلدهای شابک کتاب و کد مؤلف را به عنوان کلید اولیه تعریف می‌کند.

```
PRIMARY KEY (ISBN, atID)
```

جدول ۲- ۱ انواع داده در SQL.

نام	تعداد بایت	هدف
int	۴	اعداد صحیح از ۲,۱۴۷,۴۸۳,۶۴۸- تا ۲,۱۴۷,۴۸۳,۶۴۷ است.
intBig	۸	اعداد صحیح از $-۲^{۶۳}$ تا $۲^{۶۳}-۱$ است.
binary(n)	n	داده‌های دودویی حداکثر ۲۵۵ بایت را اشغال می‌کنند.
bit	۱	مقادیر ۰ یا ۱ را می‌پذیرد.
char(n)	n	حداکثر تا ۸۰۰۰ کاراکتر را می‌پذیرد و به ازای هر کاراکتر یک بایت را اشغال می‌کند.
datetime	۸	تاریخ و زمان را نگهداری می‌کند.
decimal(p, s)	حداکثر۱۷	اعداد اعشاری یا صحیح $۱+۱۰^{۳۸}$- تا $۱-۱۰^{۳۸}$ را نگهداری می‌کند.
float(n)	۸	از مقدار $۱,۷۹E^{۳۸}$- تا $۱,۷۹E^{۳۸}$ را نگهداری می‌کند و دقت آن تا ۱۵ رقم است.
real(n)	۴	تقریباً از مقدار $۳.۴۰E^{۳۸}$- تا $۳.۴۰E^{۳۸}$ را نگهداری می‌کند (با تقریب ۷ رقم).
image		داده‌های تصویری تا ۲GB را در صفحات ۸ کیلوبایتی ذخیره می‌کند.
money	۸	داده‌های ارزی از ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷.۵۸۰۸- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷.۵۸۰۷ است.
varchar(n)	n*۲	داده‌های یونیکد از ۱ تا ۸۰۰۰ کاراکتر را نگهداری می‌کند.
varchar(max)	n*۲	برای نگهداری کاراکترهای غیر یونیکد تا حداکثر ۲ گیگابایت منهای یک حرف استفاده می‌شود.
ntext	n*۲	داده‌های یونیکد با طول متغیر که طول بیشتر از ۸۰۰۰ کاراکتر تا ۲ گیگابایت است.
numeric	حداکثر۱۷	مترادف Decimal است.
real	۴	مترادف float است.
smalldatetime	۴	ترکیب تاریخ و زمان با طول کوتاه را نگهداری می‌کند.
smallint	۲	اعداد صحیح از ۳۲,۷۶۸- تا ۳۲,۷۶۷ را نگهداری می‌کند.
currency	۸	مقادیر پولی با حداکثر ۴ رقم اعشار را نگهداری می‌کند.
sysname(n)	n*۲	برای ذخیره کردن اسامی اشیای سیستم به کار می‌رود و به ازای هر کاراکتر دو بایت را اشغال می‌کند.
timestamp		برای نگهداری مهر زمانی در بانک اطلاعاتی به کار می‌رود.
tinyint(n)	n	اعداد صحیح ۰ تا ۲۵۵ بایت که n تعداد بایت‌ها را مشخص می‌کند.
varbinary(n)	n	الگوی بیتی تا ۲۵۵ بایت را نگهداری می‌کند.
uniqueidentifier	۱۶	عدد ۱۶ بایتی هگزا دسیمال که شناسه یکتای عمومی را نشان می‌دهد.

XML	حداکثر ۲GB	برای نگهداری اسناد XML به کار می‌رود.
varbinary(max)		اطلاعات بایتی را به صورتی در نظر می‌گیرد که می‌توان تصاویر را در آن ذخیره کرد.

۲. ایجاد محدودیت کلید خارجی[۱]: کلید خارجی برای ایجاد ارتباط بین دو جدول به کار می‌رود. یعنی، با استفاده از کلید اولیه یا کلید فرعی یک جدول و کلید خارجی جدول دیگر می‌توان ارتباط بین این دو جدول را برقرار کرد. کلید خارجی در واقع کلید اولیه یا کلید فرعی جدول دیگر در همان بانک است. برای تعریف کلید خارجی به صورت زیر عمل می‌شود:

FOREIGN KEY (فیلد۱, ... , فیلدn) **REFERENCES** نام جدول ۲

مثال ۲۲-۱. دستوری که فیلد pubID را بین جدول Publishers و PubBook به عنوان کلید خارجی تعریف می‌کند.

FOREIGN KEY (pubID) **REFERENCES** Publishers

وقتی که محدودیت جامعیت ارجاع نقض گردد، معمولاً عملی که موجب نقض گردیده است، رد می‌شود. یعنی، اگر در جدول GroupBook، بخواهید کد نوع کتاب ۰۱ را حذف کنید، در صورتی که این کد در جدول Books استفاده شده باشد، دستور حذف از جدول GroupBook لغو خواهد شد. اما، بخش FOREIGN KEY را می‌توان طوری تعریف کرد که تراکنش‌های حذف و به هنگام رسانی لغو نگردند. برای این منظور می‌توانید از گزینه‌های زیر استفاده کنید:

- **گزینه ON DELETE CASCADE**، موجب می‌شود تا عمل حذف لغو نگردد. ابتدا، تمام جداولی که با این جدول از طریق این فیلد ارتباط دارند، مقداری که در حال حذف است به صورت آبشاری حذف خواهد کرد. سپس، در جدول اصلی مقدار مورد نظر حذف می‌گردد. یعنی، اگر بخواهید در جدول GroupBook، کد نوع گروه کتاب ۰۱ را حذف کنید، ابتدا در جدول Books، رکوردهایی با کد نوع گروه کتاب ۰۱ حذف خواهند شد. سپس، در جدول GroupBook، رکوردی با کد نوع گروه کتاب ۰۱ حذف می‌گردد.
- **گزینه ON UPDATE CASCADE**، اگر در هنگام به روز رسانی در فیلدی که محدودیت جامعیت در آن تعریف شده باشد، این محدودیت نقض شود، به هنگام روز رسانی رد نمی‌گردد. به عنوان مثال، اگر بخواهید در جدول GroupBook، کد نوع گروه کتاب ۰۱ را به ۰۳ تغییر دهید. ابتدا، در جدول Books، کد نوع‌های گروه کتاب ۰۱ را به ۰۳ تغییر می‌دهد. سپس، در جدول GroupBook این تغییر را اعمال می‌کند.

استفاده از گزینه‌های ON DELETE CASCADE و ON UPDATE CASCADE به صورت زیر است:

---

۱. Foreign Key

```
CREATE TABLE Books
 (
   …
   FOREIGN KEY (groupID) REFERENCES GroupBook
   ON DELETE CASCADE,
   ON UPDATE CASCADE,
   …
 )
```

۳. محدودیت DEFAULT، این محدودیت مقداری را تعیین می‌کند که اگر کاربر برای فیلد مقدار وارد نکند، آن مقدار در این فیلد قرار می‌گیرد. یعنی، مقدار پیش‌فرض فیلد را تعیین می‌نماید.

٤. محدودیت UNIQUE، تضمین می‌کند صفات (فیلدهای) تعیین شده تشکیل کلید کاندید را دهند. یعنی، رکوردهای (چند تایی‌های) جدول (رابطه) **یکتا** هستند (هیچ دو رکوردی در جدول تکراری نیستند). فیلدهای با محدودیت UNIQUE می‌توانند تهی (NULL) باشند، مگر این که با محدودیت NOT NULL معرفی گردند. این محدودیت به صورت زیر به کار می‌رود:

```
UNIQUE (فیلد ۱, فیلد۲, ..., فیلد n)
```

۵. محدودیت NOT NULL، تضمین می‌کند مقدار فیلد در هیچ رکوردی (چند تایی) تهی نباشد. به عنوان مثال، نام کتاب نمی‌تواند تهی باشد. پس، به صورت زیر تعریف می‌گردد:

```
Title   varchar (۳۰) NOT NULL
```

٦. محدودیت CHECK، یکی از رایج‌ترین محدودیت‌ها CHECK است. این محدودیت تضمین می‌کند مقادیر صفت (فیلد)، شرط تعیین شده را داشته باشد.

مثال ۲۳ – ۱. دستوری که تضمین می‌کند که فیلد مدرک اساتید(Ranking)، فقط یکی از مقادیر "دکتری"، "فوق‌لیسانس" یا "لیسانس" را بپذیرد.

```
CHECK (Ranking IN ('دکتری' , 'فوق‌لیسانس' , 'لیسانس'))
```

مثال ۲۴ – ۱. دستوری که تضمین می‌کند که مقدار نمره دانشجویان بین ۰ تا ۲۰ باشد.

```
CHECK  (Score >= ۰,۰۰  AND  Score <= ۲۰,۰۰)
```

۷. محدودیت IDENTITY، موجب می‌شود تا این فیلد مانند یک فیلد Autonumber در اکسس عمل کند. در جدول‌هایی که فیلد کلید وجود ندارد، معمولاً فیلدی از این نوع انتخاب می‌کنند تا به عنوان فیلد کلید عمل کند. این فیلد با اضافه شدن هر رکورد، به طور خودکار یک واحد به مقدار آن اضافه خواهد شد.

۱ – ۴ – ۱. ایجاد جدول

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

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

در این ساختار table_name، نام جدولی است که می‌خواهید ایجاد کنید. $Column_1$ تا $Column_n$، نام فیلدهای (ستون‌های) جدول را تعیین می‌کنند. $type_1$ تا $type_n$، نوع ستون‌های $Column_1$ تا $Column_n$ را مشخص می‌کنند و $Constraint_1$ تا $Constraint_n$ به ترتیب محدودیت‌هایی را تعیین می‌کنند که باید بر روی ستون‌های $Column_1$ 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(٣٠) 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 ۲۶ – ۱. پرس‌وجویی که جدول Books را ایجاد می‌کند (ساختار این جدول را در جدول ۱ – ۱ ببینید).

```
USE PublishDB
GO
CREATE TABLE Books
(
  ISBN    varchar(١٠) PRIMARY KEY,
  Title   varchar(٤٠) NOT NULL,
  Page int NOT NULL,
  Price decimal(١٥،٠) NOT NULL,
  editNo int,
  printNo int,
  groupID varchar(٦) references GroupBook
)
GO
```

این دستور فیلد ISBN (شابک) را کلید اولیه تعریف می‌کند و فیلد groupID را کلید خارجی تعریف کرده تا بین جداول GroupBook و Books ارتباط برقرار کند.

مثال ۲۷ – ۱. پرس‌وجویی که جدول Publishers که ساختار آن در جدول ۱ – ۱ قرار دارد را ایجاد می‌کند.

```
CREATE TABLE Publishers
(
  pubID    varchar(۱۰) PRIMARY KEY,
  pName  varchar(۵۰) NOT NULL,
  Tel varchar(۱۵),
  URL varchar(۱۰۰),
  cityName varchar(۵۰),
  bFname varchar(۲۰) NOT NULL,
  bLname varchar(۲۰) NOT NULL,
  countBookPrint int,
 )
GO
```

مثال ۲۸ – ۱. پرس‌وجویی که ساختار جدول Authors را ایجاد می‌کند.

```
CREATE TABLE Authors
(
  atID    varchar(۱۰) PRIMARY KEY,
  aFName varchar(۲۰) NOT NULL,
  aLname varchar(۲۰) NOT NULL,
  Age int,
  Ranking varchar(۳۰),
  Email varchar(۵۰),
  Mobile varchar(۱۵),
  sumPayment decimal(۱۸, ۰)
 )
 GO
```

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

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

این دستور، دو محدودیت به نام‌های FK_ISBN و FK_atID ایجاد کرده است. محدودیت FK_ISBN، برای برقراری ارتباط بین جداول Books و AutBook به کار می‌رود که فیلد ارتباط ISBN می‌باشد و محدودیت FK_atID ارتباط بین جداول Authors و AutBook را برقرار می‌کند که فیلد ارتباط atID است.

مثال ۳۰ – ۱. پرس‌وجویی که ساختار جدول PubBook را ایجاد می‌کند.

```
CREATE TABLE PubBook
(
  ISBN    varchar(۱۰) ,
  pubID    varchar(۱۰) ,
  Payment decimal(۱۸, ۰),
  CONSTRAINT PK_ISBN_PUBID PRIMARY KEY(ISBN, pubID),
```

```
  CONSTRAINT FK_ISBN۱ FOREIGN KEY (ISBN) REFERENCES
             Books(ISBN),
  CONSTRAINT FK_pubID FOREIGN KEY (pubID) REFERENCES
                Publishers(pubID)
)
GO
```

این دستور سه محدودیت زیر را ایجاد می‌کند:

- **محدودیت PK_ISBN_PUBID**، فیلدهای ISBN و pubID را کلید اولیه این جدول تعریف می‌کند.
- **محدودیت FK_ISBN۱**، از طریق فیلد atID کلید خارجی را برای ارتباط بین جداول Books و AutBook تعریف می‌کند.
- **محدودیت FK_PubID**، از طریق فیلد pubID کلید خارجی را برای ارتباط بین جداول AutBook و Publishers تعریف می‌کند.

۲-۴-۱. تغییر ساختار جدول با دستور SQL

گاهی نیاز است در جدولی که داده وجود دارد، فیلد جدیدی اضافه نموده، فیلدی را حذف نمایید یا مشخصات فیلد را تغییر دهید. برای این منظور دو روش وجود دارد: ۱. **حذف جدول موجود و ایجاد مجدد همان جدول**. در این روش کلیه اطلاعات موجود در جدول حذف خواهند شد که معقول نمی‌باشد. ۲. **استفاده از دستور ALTER TABLE**، با این دستور می‌توانید ساختار جدول را تغییر دهید، به طوری که اطلاعات قبلی در جدول باقی بماند. دستور ALTER TABLE به صورت زیر به کار می‌رود:

```
ALTER TABLE  table_name
 {
   | { ALTER COLUMN column_name <column_definition>}
   | { ADD <column_definitiON>}
   | { DROP[CONSTRAINT]Constraint_name|COLUMN column_name }
 } [ ; ]
```

پارامترهای این دستور در زیر آمده‌اند:

- **پارامتر table_name**، نام جدولی است که می‌خواهید ساختار آن را تغییر دهید.
- **پارامتر ALTER COLUMN**، خواص فیلدهای موجود را تغییر می‌دهد.
- **پارامتر ADD**، فیلدهای جدیدی به جدول اضافه می‌کند.
- **پارامتر DROP**، فیلدها یا محدودیت‌های موجود را از جدول حذف می‌کند.

مثال ۳۱ - ۱. دستوراتی که فیلد جدیدی به نام newColumn در جدول Authors از بانک اطلاعات فعلی (همان PublishDB) اضافه می‌کنند.

```
ALTER  TABLE Authors
   ADD newColumn varchar(۱۰) NULL
GO
```

مثال ۳۲ - ۱. پرس‌وجویی که فیلدهای newColumn۱ و newColumn۲ را به جدول Authors اضافه می‌کنند.

```
ALTER  TABLE Authors
```

```
  ADD newColumn۱ int , newColumn۲ varchar(۳۰)
GO
```

مثال ۳۳ – ۱. پرس‌وجویی که اندازه فیلد newColumn۲ در جدول Authors را از ۳۰ به ۲۰ کاراکتر تغییر می‌دهد.

```
ALTER  TABLE Authors
  ALTER COLUMN newColumn۲ varchar(۲۰)
GO
```

مثال ۳۴ – ۱. پرس‌وجویی که فیلدهای newColumn، newColumn۱ و newColumn۲ را از جدول Authors حذف می‌کند.

```
ALTER  TABLE Authors
 DROP COLUMN newColumn, newColumn۱, newColumn۲
GO
```

مثال ۳۵ – ۱. پرس‌وجویی که جدول Test با دو فیلد x و y از نوع int را به بانک اطلاعاتی فعلی (PublishDB) اضافه می‌کند (فیلد x، مقدار تهی را نمی‌پذیرد).

```
CREATE TABLE Test  ( x int NOT NULL, y int)
GO
```

مثال ۳۶ – ۱. پرس‌وجویی که محدودیت PK_X را به جدول Test اضافه می‌کند که فیلد x را به عنوان کلید اولیه آن جدول در نظر می‌گیرد.

```
ALTER TABLE Test
  ADD CONSTRAINT PK_X PRIMARY KEY(x)
GO
```

نکته: محدودیت‌های جدول را نمی‌توانید تغییر دهید. برای تغییر محدودیت‌های جدول، ابتدا باید آن را حذف کرده، سپس آن را مجدداً ایجاد کنید. برای حذف محدودیت در دستور ALTER TABLE باید از پارامتر DROP CONSTRAINT استفاده نمود.

مثال ۳۷ – ۱. پرس‌وجویی که محدودیت FK_X را از جدول Test حذف می‌کند.

```
ALTER TABLE Test
  DROP CONSTRAINT PK_X
GO
```

۳–۴–۱. حذف جدول با دستور SQL

جدول‌های اضافی را باید از بانک اطلاعاتی حذف نمود. زیرا، از آنجایی که جداول داده‌ها را ذخیره می‌کنند، فضای زیادی از بانک اطلاعاتی را اشغال می‌نمایند. بنابراین، برای آزادسازی فضای بلااستفاده باید جداول اضافی را حذف کرد تا فضای بیشتری در اختیار بانک اطلاعاتی قرار گیرد. برای حذف جدول‌های اضافی بانک اطلاعاتی می‌توانید از دستور DROP TABLE به صورت زیر استفاده کنید:

```
DROP TABLE  table_name [۱,...,n ]
```

در این دستور پارامترهای [table_name[۱,...,n، لیست جداولی هستند که می‌خواهید از بانک اطلاعاتی حذف کنید. اگر تعداد جداولی که باید حذف شوند، بیش از یکی باشند، بین نام آن‌ها علامت، (کاما) قرار می‌گیرد.

مثال ۳۸ – ۱. دستوراتی که جدول Test را حذف خواهند کرد.

```
DROP TABLE Test
GO
```

نکته: اگر جدولی موجود نباشد و بخواهید آن را حذف کنید یا ساختار آن را تغییر دهید، از طرف SQL، پیغام خطا صادر می‌شود. ولی، اگر جدولی موجود باشد و بخواهید با دستور CREATE TABLE آن را مجدداً ایجاد کنید، SQL پیغام خطا مناسب را نمایش می‌دهد. با حذف جدول، ساختار جدول، داده‌های آن، ایندکس‌ها، تریگرها و محدودیت‌های تعریف شده برای آن نیز حذف خواهند شد.

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