اسلایدهای درس

پیاده‌سازی سیستم‌های پایگاه داده

دکتر پویا فوده

مهر ۰۱—۰۰

پیاده‌سازی سیستم‌های پایگاه داده

جلسه اول

آشنایی

مباحث امروز

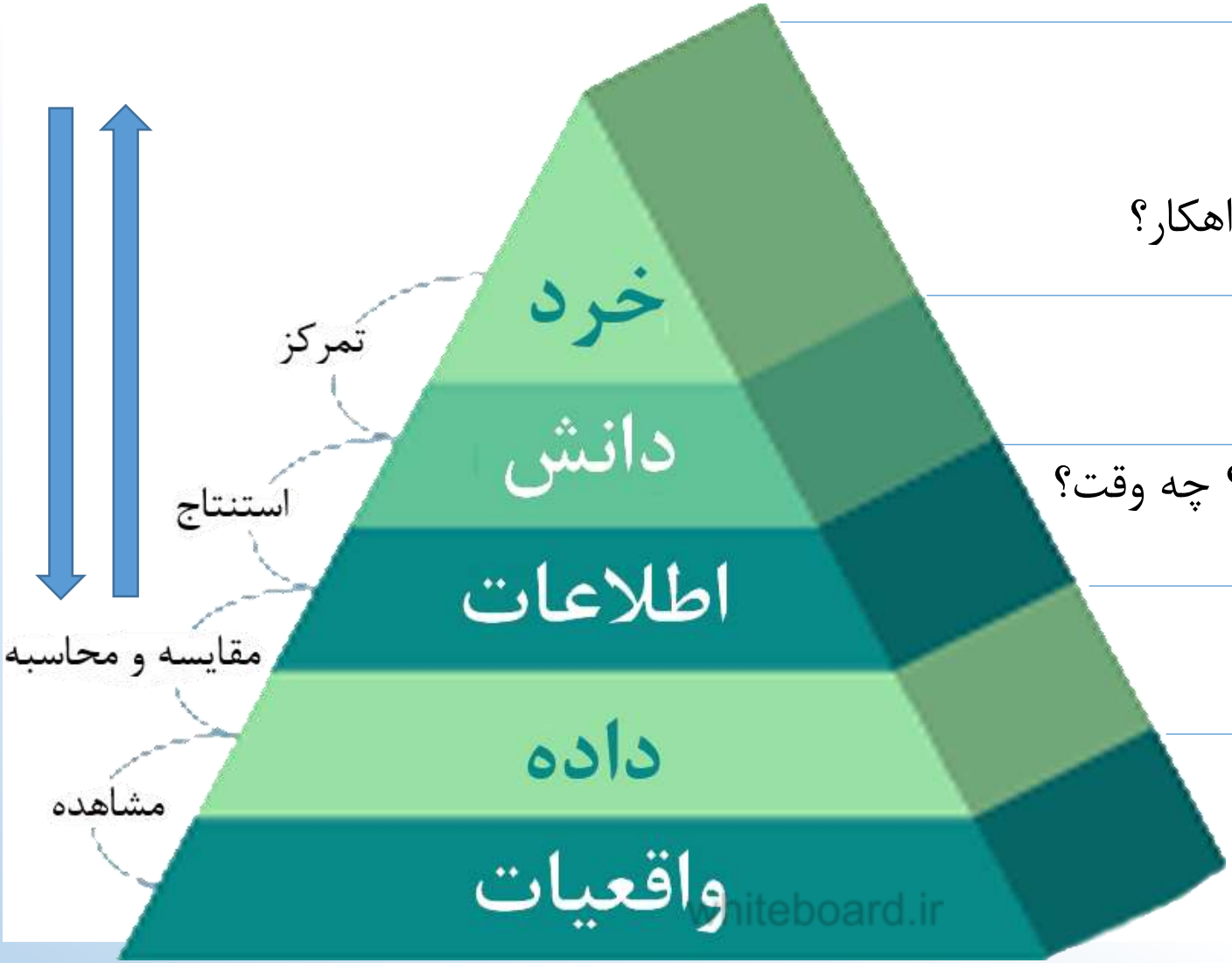
- مقدماتی در مورد داده و پایگاه داده.
- بعد از گذراندن درس پیاده‌سازی پایگاه داده، چه معلوماتی به ما اضافه می‌شود.
- منابع و کتاب‌های این درس.
- تمرینات و نحوه محاسبه نمره پایان ترم.

هرم داده، اطلاعات، دانش، خرد

ارزش حجم

هرم

ماهیت درخواست



چرا؟
بهترین راهکار؟

چگونه؟

آیا؟ کجا؟ چه وقت؟
چه ...؟

X

داده یا دیتا

- داده‌های که مورد علاقه‌اند با اندازه‌گیری، شمارش، پرسش‌گری از محیط جمع‌آوری می‌شوند.
- داده در شکل خالص آن شامل نشانه‌های ساده همچون اعداد و حروف الفباست. پردازش نشده و نامنظم است. از نظر معنایی ارزشی ندارد.
- چند نمونه از داده:

- ❑ تعداد نفرات خانوار که مامور سرشماری در فرم می‌نویسد.
- ❑ نمره‌ای که استاد بعد از مطالعه برگه امتحانی بالای برگه دانشجو می‌نویسد.
- ❑ نامی که بر روی یک برگ رای با یک شماره سریال نوشته شده.
- ❑ اعدادی که دستگاه آزمایش خون پس از پردازش یک نمونه خون بر روی کاغذ چاپ می‌کند.

اطلاعات

- اطلاعات حقایق معناداری هستند که مستقیم یا بعد از پردازش از داده‌های خام بدست می‌آیند.
- اطلاعات دارای نظم و ساختار هستند و از نظر معنایی هم دارای ارزش می‌باشند.
- چند نمونه از اطلاعات:

- ❑ متوسط جمعیت خانوار در شهر بناب چند نفر است.
- ❑ آیا دانشجو فارغ‌التحصیل است (همه درس‌ها را گذرانده)
- ❑ چه کسی رئیس‌جمهور منتخب است (اکثریت آرا را دارد)
- ❑ آیا صاحب نمونه سرطان خون دارد.

دانش

- دانش نتایج کلی است که از تجزیه و تحلیل و استنتاج بر روی اطلاعات و داده‌ها به دست می‌آید. پایگاه‌های دانش یکی از پایه‌های سیستم‌های پشتیبانی از تصمیم (که سازمان‌ها را در تصمیم‌گیری کمک می‌کنند) هستند.
- یک نمونه بارز از کاربرد دانش و پایگاه دانش، سیستم‌های فهم زبان‌های طبیعی انسانی و ترجمه یک زبان به زبان دیگر است. در این مورد داده‌ها و اطلاعات (دیکشنری) کفایت نمی‌کند چون هر لغت در هر زبان دارای طیف گسترده‌ای از معانی است و درک اینکه در یک جمله مراد کدام معنا بوده نیازمند دانش است نه اطلاعات لغت‌نامه‌ای. (WordNet پایگاه دانشی است برای زبان انگلیسی)

خرد یا شعور

- عالی‌ترین سطح و در عین حال دست نایافتنی‌ترین و گران‌بهاترین طبقه این هرم است.
- در این سطح تصمیم‌های حیاتی و مهم با استفاده از دانش و اطلاعات موجود گرفته می‌شوند.
- مثلا تصمیم‌گیری در مورد اینکه «با فلان شخص ازدواج کنم یا نه» یا اینکه «به کشور دیگری مهاجرت کنم یا نه» نیاز به خرد دارد. بدون شک طیف وسیعی از دانش و اطلاعات مانند اینکه: من الان سرطان خون دارم... وضعیت مالی من چگونه است.... رئیس جمهور کشور الان چه کسی است.... و هزاران قطعه اطلاعات مانند این لازمه عملکرد خرد در پاسخگویی به چنین سوالاتی است.
- عملکرد خرد نه تنها وابسته به قدرت استنتاج و اطلاعات زیادی است بلکه به عوامل دیگری چون قدرت قضاوت و اخلاقیات نیاز دارد.
  - از این رو برخی معتقدند این سطح در انحصار انسان است، اما
  - برخی دیگر معتقدند تمام کارهایی که انسان انجام می‌دهد را
  - ماشین نیز می‌تواند انجام دهد.

چرا داده‌ها را در کامپیوتر ذخیره کنیم

- فشردگی: فضای کمتر ازبایگانی‌های
- سرعت: دسترسی به پرونده‌ها فوری است و تهیه گزارش‌ها در زمان کوتاهی انجام می‌شود.
- هزینه: صرفه‌جویی مالی
- امنیت وجودی: داده‌ها در حوادث از بین نمی‌روند.
- امنیت دسترسی: فقط افراد مجاز می‌توانند اطلاعات را ببینند یا داده‌ها را تغییر دهند.

چرا پایگاه داده

- بجای استفاده از یک سیستم برنامه‌نویسی شده فایلی ما معمولا از پایگاه داده (سیستمی که اولا در یک DBMS ذخیره شده ثانیا شامل تمام داده‌های کال سازمان بصورت یکپارچه است) استفاده می‌کنیم چرا که:
- کارایی: یک DBMS (مثل Oracle یا SQLServer) که با هزینه بسیار توسط صدها نفر طی سال‌ها در یک شرکت بزرگ نوشته شده از نظر کارایی (سرعت) قابل مقایسه با سیستم فایلی که ما برنامه‌نویسی می‌کنیم نیست.
- استقلال ساختاری و داده‌ای: تغییرات در ساختار و حجم داده‌ها موجب تغییرات کلی در ساختار فایل‌ها و برنامه‌نویسی مجدد سیستم نخواهد شد.

چرا پایگاه داده

- جلوگیری از افزونگی: با یک پایگاه سراسری، از ثبت داده‌ها در چند جای مختلف یک از سازمان (مثلا تلفن یک دانشجو در گروه آموزشی و صندوق رفاه) جلوگیری می‌شود.
افزونگی غیر از هدر دادن منابع سیستم موجب آن می‌شود که داده‌ها همیشه در خطر ناهنجاری و ناپایداری باشند.

یک سوال مهم

- با وجود آنکه ما قبلا در درس پایگاه داده مواردی چون طراحی پایگاه داده رابطه‌ای، زبان SQL و احتمالا نرمال‌سازی و کار با یک DBMS را آموخته‌ایم، چه نیازی به این درس وجود دارد؟

پاسخ

- چون تقریبا هیچیک از اهداف سیستم پایگاه داده با طراحی ناقص و ضعیف محقق نمی‌شود.
- سرعت و کارایی: سیستم پایگاه داده تنها وقتی به حداکثر کارایی خود می‌رسد که DBMS مناسبی اختیار شده و طراح ضمن شناخت عمیق نظریه رابطه‌ای، اصول شاخص‌گذاری و بهینه‌سازی پرس‌وجوها را مد نظر قرار داده باشد.
- امنیت دسترسی: در سیستم پایگاه‌داده اگر اصول امنیتی توسط طراح اعمال نگردد، امنیت سیستم به مراتب ضعیف‌تر از بایگانی‌های کاغذی خواهد بود که دست کم به صورت فیزیکی حفاظت می‌شوند.

پاسخ

- امنیت وجودی: درست است که پایگاه‌داده در صورت پشتیبان‌گیری مناسب در برابر حوادث طبیعی مانند زلزله و آتش‌سوزی آسیب ناپذیر است، اما خطاهای سخت‌افزاری و نرم‌افزاری در کامپیوتری که پایگاه در آن است در هر لحظه ممکن‌اند، و داده‌ها را در خطر نابودی و خرابی قرار می‌دهد. مدیریت تراکنش‌ها، توالی‌پذیری، هم‌روندی و ترمیم پایگاه این خطرات را تا حدود زیادی دور می‌کند.
- جلوگیری از افزونگی: استفاده از DBMS ما را از شر افرونگی رها نمی‌کند. این مهم وابسته به درک عمیق قواعد نرمال‌سازی و نیز دانش استفاده صحیح از SQL است.

آنچه قاعدتا اکنون می‌دانید

- مفاهیم زیربنایی پایگاه داده‌ها
- طراحی پایگاه داده (رسم نمودار ER)
- مفاهیم اصلی مدل رابطه‌ای (همچون تاپل و ویژگی و کلید و جامعیت)
- کار با زبان SQL و کار عملی با حداقل یک DBMS
- و ... احتمالا وابستگی تابعی و نرمال سازی

- و در عین حال که اینها را می‌دانید هنوز برای ایجاد یک پایگاه که به صورت اصولی طراحی شده چیزهای دیگری را هم باید بدانید....

آنچه قرار است در این درس بیاموزید

- انتخاب DBMS مناسب، شروع کار با SQL Server. [4]
- اصول پایگاه رابطه‌ای با شناخت مفاهیم ریاضی(تاپل، رابطه، دامنه، متغیرهای رابطه‌ای، جامعیت و نرمال‌سازی) [3]
- برنامه و کدنویسی پایگاه‌داده (روال‌های ذخیره‌شده) [4]
- طراحی اصولی پایگاه‌داده (وابستگی‌ها و نرمال سازی)[3][1]
- قیود غیرعمومی و ایجاد آنها با تریگر. آشنایی با XML. [4]
- مدیریت تراکنش‌ها، توالی‌پذیری، هم‌روندی و ترمیم [2]
- ارتباط با اپلیکیشن‌ها [-]
- افزایش کارایی (کوئری‌های بهینه و شاخص‌گذاری) [2] [4]
- اصول امنیتی پایگاه داده و پیاده‌سازی آنها، پشتیبان‌گیری[4]
- پایگاه‌های توزیع‌شده [2 دیت] [4]

کتاب و منابع درسی

کتاب و منابع درسی

- کتاب روحانی رانکوهی انتشارات جلوه

جلد اول این کتاب سال‌هاست که بعنوان مهمترین کتاب درسی فارسی پایگاه داده تدریس می‌شود و جلد دوم آن نیز همچون جلد اول جامع، صحیح و شیواست.

- کتاب دیت نشر علوم رایانه

این کتاب نیز در جهان یکی از مهمترین کتاب‌های درسی پایگاه داده است، علاوه بر آن که دیت همکار کاد در ارائه نظریه رابطه‌ای بوده است، معلم خوبی هم هست و کتاب آموزنده‌ای نوشته است اما متاسفانه ترجمه کتاب در سطح خود کتاب و همچنین در سطح نوشتار روان رانکوهی نیست.

جلد اول این کتاب‌ها مباحث مقدماتی [1] و جلد دوم مباحث پیشرفته [2] را پوشش می‌دهد.

کتاب و منابع درسی

- [3] نگاه عمیق‌تر به پایگاه داده‌ها نقش سیمرغ

دیت در این کتاب (بر خلاف کتاب قبل که یک متن درسی است) نکات عملی طراحی را (تئوری در خدمت عمل) برای کسانی که می‌خواهند در کار پایگاه داده به صورت حافه‌ای مشغول شوند شرح داده است. حدود یازده سال از آخرین چاپ کاغذی کتاب گذشته و بجای تجدید چاپ، آن را برای دانلود گذاشته‌ام (به whiteboard.ir مراجعه کنید).

- [4] آموزش SQL Server 2016 نشردانشگاهی کیان

این کتاب پیاده‌سازی عملی اکثر مطالب این درس را شرح می‌دهد. نه تنها برای این درس، بلکه برای انجام پروژه‌های واقعی پایگاه داده به چنین منبعی نیاز خواهید داشت.

پیش ارائه جلسات درس

- مقدمه هر جلسه از درس اهمیت زیادی دارد. در این مقدمه گفته می‌شود که قرار است امروز چه مطالبی ارائه شود و این مطالب به چه کاری می‌آید.
- پیش ارائه به صورت فیلمی کوتاه (چند دقیقه‌ای) هر هفته در روز قبل از درس به ادمودو ارسال می‌گردد. توصیه می‌شود این فیلم را در زمانی که فرصت و تمرکز ذهنی دارید مشاهده کنید تا موضوع در ذهن‌تان جای گیرد و برای درک درسی که فردا در کلاس ارائه می‌شود آمادگی خوبی پیدا نمایید.

نمره پایان ترم

- **ارزیابی مستمر در طول کلاس (۱۳.۵ نمره):**

✓ تمرین‌های هفتگی ۷ نمره ۵ نمره
توجه: نمره تمریناتی که اصالت نداشته باشند در پایان ترم حذف خواهد شد. فرقی بین دهنده و گیرنده تمرین نیست. تقلب در تمرین و امتحان در نهایت مرتکبین و کل کلاس را متضرر خواهد کرد. توضیح بیشتر در وب سایت موجود است.

✓ امتحان میان ترم ۶.۵ نمره

- **ارزیابی پایان ترم:**

✓ امتحان پایان ترم ۶.۵ نمره ۱۵ نمره

✓ نمره ارفاقی به همه بطور مساوی (مقدار آن آخر ترم معلوم می‌شود)

✓ *در صورتی که امتحان پایان ترم بصورت حضوری برگزار شود نمره قرمز رنگ.

- **نمرات اضافه بر سازمان**
- تمرین برجسته (هر تمرین ۰.۵ هر دانشجو حداکثر ۲ نمره در ترم)
- مشارکت برجسته در کلاس (هر بار ۰.۵ هر دانشجو حداکثر ۲ نمره در ترم)
- کوئیز (در ترم معمولا ۳ یا ۴ کوییز خواهد بود که کلا ۱ نمره دارد)

نمرات اضافه بر سازمان

- تمرین یا مشارکت برجسته معمولا هر جلسه به یک نفر داده می‌شود که بهترین تمرین را ارائه یا بهترین مشارکت را در کلاس آنلاین داشته باشد. البته در شرایطی ممکن است به دو نفر داده شود یا به کسی داده نشود.
- تمرین برجسته تمرینی است که به نکاتی که کسی به آن دست نیافته دست یابد و بخشی که همه نادرست یا ناقص حل کرده باشند را حل نماید.
- مشارکت کننده برجسته (ویژه جلسات آنلاین) در کلاس کسی است که سوالی که دیگران نتوانند را پاسخ گوید یا سوال خوبی ی مطرح کند که به فهم درس کمک نماید. استفاده از میکروفون در این مورد می‌تواند نقش زیادی داشته باشد.
- کوییز تک سوالی است که بدون هماهنگی قبلی از درس همان روز یا درس‌های قبل یا مباحث مربوط به درس مطرح می‌شود و در زمان محدود در ادمودو ارسال می‌گردد.

کوییزهای انفرادی و گروهی

- برخی کوییزها انفرادی هستند و برخی بصورت گروهی برگزار می‌شوند.
- قوانین: دانشجویان بطور تصادفی در گروه‌هایی چند نفره قرار خواهند گرفت. می‌توانید با کمک هم به حل مساله بپردازید و در نهایت یک برگه با ذکر شماره گروه و نام همه اعضا گروه تحویل داده خواهد شد و به همه نمره یکسان تعلق می‌گیرد.
- در موارد زیر نمره صفر به همه اعضای گروه تعلق خواهد گرفت:
  ۱- عدم ارسال پاسخ در ادمودو.
  ۲- ارسال بیش از یک پاسخنامه از یک گروه.
  ۳- اگر نام گروه و اعضای آن با آنچه هنگام تعیین گروه‌ها اعلام شده متفاوت باشد (کم یا زیاد باشد.)
- در کلاس‌های حضوری گروه‌بندی توسط من در کلاس انجام می‌شود و گروه‌ها دور هم نشسته و سوال را با مشورت هم حل می‌کنند. در کلاس آنلاین اعضای گروه در یک اتاق مستقل مجازی قرار گرفته و در زمان جلسه همه امکان نوشتن در کادر چت و استفاده از میکروفون را دارند. کسی هر کس می‌تواند خودش را ارائه دهنده کند و روی تخته سیاه بنویسد یا آن را برای همه باز کند.

تمرین‌ها

- تمرین‌ها: هر هفته به صورت مرتب. زمان تحویل تا فقط هفته آینده. تمرین تاریخ گذشته ارزشی ندارد.
- برای تحویل تمرین‌ها، دریافت اسلایدها، امتحان میان‌ترم، و سوال در مورد درس از سیستم ادمودو edmodo استفاده خواهد شد.
- برای اطلاع از جزییات شرایط تحویل تمرین‌ها و دستورالعمل استفاده از ادمودو به سایت من مراجعه شود: whiteboard.ir
(هم از طریق وب سایت می‌شود و هم با نصب اپ اندروید یا IOS.)
- کد ادمودو (کلاس پیاده‌سازی پایگاه داده‌ها):
یکشنبه g9fnuk دوشنبه grhsny سه‌شنبه sa3ndp

نکته ۱: نام خود را فارسی وارد کنید. نه پنگلیش.
نکته ۲: اسلایدها و تمرین‌ها احتمال زیاد دارد که با ترم پیش متفاوت باشد. همیشه اسلایدهای جدید را بصورت هفتگی دریافت کنید.
نکته۳: تمرینات را PV و ایمیل نکنید. Zip و Rar نفرستید. راهنما را بخوانید.

پیاده‌سازی سیستم‌های پایگاه داده

جلسه دوم

انتخاب DBMS

شروع کار با آن

شروع نظریه رابطه‌ای

مباحث امروز

- شناخت معیارهای ارزیابی یک DBMS از نظر:
  آسانی نصب و کار
  نحوه اتصال و استفاده کاربران
  کارایی
  قیمت
  لهجه SQL
  هم خانوادگی با ابزار برنامه‌سازی
- نصب و کار با SQL Server
- آغاز مبحث نظریه رابطه‌ای

سیستم مدیریت پایگاه داده DBMS

- DBMSیک برنامه کامپیوتری است که به کاربر اجازه ایجاد و دسترسی پایگاه داده را می‌دهد.
- این برنامه جامع و بسیار پیچیده است و از همین رو (همانند سیستم عامل) فقط تعداد اندکی از آن وجود دارد که توسط شرکت‌ها یا سازمان‌های بزرگ کامپیوتری نوشته شده‌اند.
- پایگاه داده (دیتابیس) بدون DBMS معنا ندارد. به بیان دیگر هر سیستم پایگاه داده باید در یک DBMS پیاده سازی شده باشد.

انتخاب DBMS

- کسی که قرار است ایجاد کننده پایگاه داده باشد دارای تجربه و دانش محدودی است. هر مهندس کامپیوتر معمولا به یک یا حداکثر دو DBMS مسلط است. بنابراین اینکه در نقطه صفر و برای انجام نخستین پروژه‌ها انتخاب ما چه باشد، ممکن است برای سال‌ها (یا همیشه) ما را در همان مسیر قرار دهد.
- DBMS های موجود با هم **تفاوت‌هایی واقعی** دارند (بر خلاف زبان‌های برنامه‌سازی که تفاوتشان بیشتر ظاهری است). شناخت این تفاوت‌ها موجب می‌شود که انتخاب‌مان همراه با آگاهی باشد.

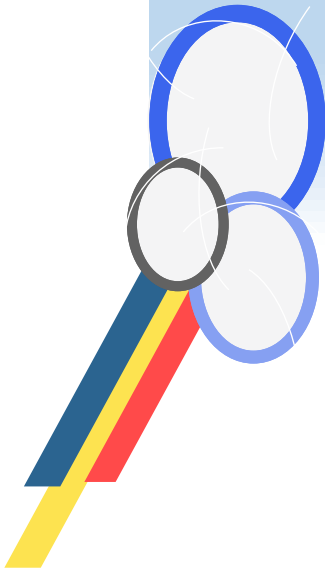
ORACLE®

DATABASE

Microsoft®
SQL Server®

Microsoft
Access

MySQL®



آسانی نصب و نگهداری

- DBMS ها از این لحاظ با هم متفاوت‌اند. Access یک سیستم بسیار آسان، Oracle دشوار و MySQL و SQLServer در سطح متوسط هستند.
- به طور کلی اگر نصب یک نرم‌افزار آسان باشد می‌توان این کار را به خود کاربر محول کرد و اگر متوسط باشد به پرسنل کامپیوتری سازمان استفاده کننده.
- اگر کار با DBMS آسان باشد وقت و انرژی طراح کمتر صرف تنظیمات آن می‌شود. همچنین نیروهای نیمه متخصص که در استخدام سازنده پایگاه یا سازمان خریدار سیستم هستند می‌توانند کار پشتیبانی و نگهداری پایگاه داده را عهده‌دار شوند.

آسانی نصب و نگهداری

- موتور پایگاه Access همراه مجموعه مایکروسافت آفیس نصب می‌شود. راه‌اندازی پایگاه به آسانی انتقال یک فایل است. طراحی پایگاه در آن تماما با منوهای خودکار و بدون حتی یک خط کدنویسی ممکن است.
- اما این سادگی زیاد به قیمت از دست رفتن بسیاری از امکانات تمام می‌شود، از جمله امنیت بسیار پایین و عدم کارایی با داده‌های حجیم.
- Oracle نقطه مقابل Access است. تنظیمات و ریزه‌کاری‌های آن بسیار زیاد است و انجام کارهای نسبتا ساده در آن نیازمند تکنسین‌های بسیار متخصص است اما در عوض امکانات و انعطاف زیادی هم به ما می‌دهد.

آسانی نصب و نگهداری

- دو سیستم MySQL و Server SQL از این نظر متعادل‌اند. نه آنقدر ساده‌اند که طراح را از امکاناتی که برای پیاده‌سازی یک سیستم جدی به آن نیاز است محروم می‌کنند، نه آنقدر پیچیده‌اند که وقت و هزینه کاری بسیاری را برای پیاده‌سازی یک سیستم معمولی مصرف کنند.
- با این حال کاربری که هیچ تخصصی در کامپیوتر ندارد نمی‌تواند به راحتی از پس نصب و اشکال‌یابی این سیستم‌ها برآید.

اتصال کاربران به پایگاه

- سیستم Access تنها با اشتراک یک فایل بر روی ویندوز برای کاربران کامپیوترهای دیگر شبکه قابل استفاده است. در این سیستم فرم‌های ورود داده و گزارش‌های خروجی داده بدون کدنویسی (و نیاز به ایجاد اپلیکیشن) بصورت یکجا با پایگاه قابل ایجاد است.
- در بقیه سیستم‌هایی که نام بردیم، اتصال کاربران بصورت کلاینت-سرور (تحت ویندوز، لینوکس یا وب) انجام می‌شود که نیازمند یک شبکه ساخت‌یافته است. فرم‌ها (برای ورود داده) و گزارش‌ها (برای مشاهده و چاپ داده) نیز بصورت برنامه‌هایی مستقل هستند که بصورت جداگانه توسط یک زبان برنامه‌نویسی دلخواه ساخته شده‌اند.
- اتصال همچنین می‌تواند بصورت غیر مستقیم و از طریق وب سرور باشد.

کارایی

- وقتی که پایگاه داده قرار است روی داده‌های بزرگ کار کند یعنی کاربر تقاضای جستجو یا محاسبه را روی ده‌ها میلیون رکورد اجرا می‌کند و طبعا در کسری از ثانیه انتظار دریافت پاسخ دارد، و یا اینکه در آن واحد صدها کاربر از طریق شبکه داخلی یا اینترنت مشغول کار با پایگاه هستند و توقع دارند که احساس کنند خودشان به تنهایی مشغول به کار با سیستم‌اند و به دلیل زیاد شدن کاربران ابدا احساس کندی نکنند. در این صورت است که شما یک سیستم با کارایی بالا نیاز دارید.

کارایی

- Oracle از دیرباز یک سیستم با کارایی بسیار بالا بود (ابتدا برای مین فریم ساخته شد و سپس برای PC دوباره نویسی شد)، SQLServer و MySQL هم در طول زمان کارایی خود را بهبود دادند و هم اکنون هر سه کارایی بالا و مطلوبی دارند.
- باید در نظر داشت که حجم داده و کاربران یک پایگاه ممکن است در ابتدا بسیار کم باشد اما در آینده بسیار افزایش یابد. از ابتدا باید به فکر آینده بود.
- SQLServer و Oracle دارای (نگارش) ادیشن‌های متفاوتی هستند که در آنها حداکثر اندازه بانک و نیز حافظه کامپیوتر که توسط بانک مورد استفاده قرار می‌گیرد (و کارایی وابسته به آن است) در نگارش‌های مختلف فرق می‌کند و هر نگارش هم قیمتی دارد (ساده‌ترین نگارش که برای کارهای دانشجویی مناسب است رایگان است).

قیمت

- نگارش استاندارد Oracle ۱۷۵۰۰ دلار و نگارش استاندارد SQLServer ۱۰۰۰ دلار قیمت دارد. نگارش‌های پیشرفته آنها هم با قیمت چند ۴۷ هزار دلار و ۱۴ هزار دلار برای هر هسته پردازنده است. MySQL رایگان است. Access هم همراه بسته آفیس قیمت ارزانی دارد.
- قیمت نرم‌افزار شاید در شرایط کنونی برای ما عامل مهمی نباشد، اما در شرایطی که داده بر روی هاستینگ وب خارج از سازمان قرار می‌گیرد، هزینه سالانه MySQL که بر روی سرور لینوکس قرار دارد بسیار ارزانتر از SQLServer است که روی سرور ویندوز است.
- در مورد وب در اکثر هاست ها تنها دو انتخاب وجود دارد: MySQL و SQLServer

لهجه SQL (dialects)

- گرچه SQL زبانی است دارای استاندارد، اما تقریبا هیچکدام از DBMS ها این استاندارد رعایت نمی‌کنند.
- نتیجه این شده که زبان SQL متعلق به SQLServer با زبان MySQL و بقیه فرق دارد. کدهای SQL (برخلاف کدهای زبان‌های استاندارد برنامه نویسی) یکسان نیستند.
- این مساله باعث شده که طراحان پایگاه به سختی بتوانند به چند DBMS مختلف مسلط شوند و هر چند انتقال داده از یک DBMS به DBMS دیگر به راحتی ممکن است اما کدها غیر قابل انتقالند.

هم خانوادگی با ابزار برنامه‌سازی

- MySQL هم خانواده PHP است و SQLServer از خانواده .NET. است.
- در نتیجه این هم خانوادگی این است که هر DBMS برای ارتباط با ابزار برنامه‌سازی هم خانواده خود دارای امکانات و تسهیلات ویژه‌ای است (هر چند هر زبان برنامه‌نویسی با هر DBMS قابل اتصال است) .
- دستورات غیر SQL (پروسیجرها)، در SQLServer به دستورات زبان‌های .NET. و در Oracle به Java شباهت دارد و MySQL هم زبان خودش را دارد.

در نتیجه راحت‌تر است ابزار برنامه‌سازی و DBMS هم خانواده باشند.

DBMS انتخابی این درس

- با توجه به مزایایی که بر شمردیم، در آینده برای انتخاب یک DBMS مختاریم.
- اما اکنون و در جلسات آینده برای این درس لازم است که یکی از آنها را انتخاب کنیم تا برای آموزش و تمرین از آن بهره گیریم. از این رو SQL Server Express (free edition) را لازم است بر روی کامپیوتر تحت ویندوز نصب نمایید.

نصب DBMS

- این نرم‌افزار را از سایت ماکروسافت دریافت و نصب کنید.

https://www.microsoft.com/en-us/sql-server/sql-server-downloads

Or, download a free specialized edition

Developer

SQL Server 2017 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

Download now

Express

SQL Server 2017 Express is a free edition of SQL Server, ideal for development and production for desktop, web, and small server applications.

Download now

نصب DBMS

- پس از دریافت فایل نصاب نرم‌افرار باید ابتدا موتور DBMS و سپس رابط کاربری آن بنام Management Studio را نصب کنید.
- مراحل و انتخاب‌هایی هنگام نصب وجود دارد. برای نصب صحیح مراحل کار را از این ویدیو دنبال کنید (البته ویدئو متعلق به ورژن 2017 است اما برای 2019 هم تفاوت چندانی وجود ندارد):

https://www.aparat.com/v/bQeL7

تئوری رابطه‌ای

- یک تئوری (نظریه) است که یک مدل ریاضی را برای پایگاه‌های داده طرح و ارائه کرده است.
- این تئوری طی مقاله‌ای در سال ۱۹۷۰ توسط ای‌اف کاد منتشر گردید.
- در همان محدوده زمانی، مدل‌های دیگری نیز برای پایگاه‌های داده منتشر و برخی پیاده شدند اما همگی در برابر مدل رابطه‌ای شکست خوردند. امروزه مدل رابطه‌ای تنها و بی‌رقیب است و با توجه به گذشت مدت زمان طولانی بعید است در آینده (حتی دور) رقیبی برای آن پیدا شود.

برتری نظریه رابطه‌ای: پشتوانه ریاضی

- مدل رابطه‌ای از یک طرف برای کاربردهای عملی طراحی شده، اما از طرف دیگر دارای پشتوانه محکم ریاضی است.
- نظریه رابطه‌ای (مقاله ۱۹۷۰ کاد با ۱۲ هزار ارجاع) از این نظر شبیه فازی (مقاله ۱۹۶۵ لطفی‌زاده با ۸۴ هزار ارجاع) است که هر دو با وجود تئوری و ریاضی بودن کاربرد عظیمی در سیستم‌های عملی پیدا کرده‌اند.
- طرح یک مدل با پشتوانه ریاضی بسیار دشوار است اما چنین مدلی اگر طرح شود مستحکم خواهد بود. مدل‌های عملی که پشتوانه ریاضی ندارند در طول زمان آنقدر تغییر می‌کند که کم‌کم مدل اصلی از رده خارج می‌شود.

برتری نظریه رابطه‌ای: انتزاع زیاد

- مدل رابطه‌ای هیچ چیز در مورد پیاده‌سازی فیزیکی (روی دیسک) ندارد و در واقع دست سازنده DBMS را کاملا در ساخت سیستمی کارا، باز گذاشته است.
- در عوض این مدل به طراحان پایگاه این امکان را می‌دهد تا داده‌ها را به صورت سطح بالا ببینند و طراحی کنند. طراح باید مدل رابطه‌ای را بشناسد و کارش را بر مبنای آن انجام دهد

جدایی کامل مدل از پیاده‌سازی

- زمانی که طراح پایگاه با مدل رابطه‌ای کار می‌کند، هیچ کاری به موضوعات پیاده‌سازی ندارد. مثلا اینکه داده‌ها چگونه روی هارد دیسک ذخیره می‌شوند و یا اینکه ایندکس‌ها و جستجو به چه صورت پیاده می‌شود. این مباحث موضوع درسی بنام «ذخیره و بازیابی اطلاعات» که اخیرا از دروس کارشناسی حذف شده‌اند. DBMS همه این کارها را خودش انجام می‌دهد و طراح اجازه دخالت ندارد.
- مدل‌های رقیب مدل رابطه‌ای که اکنون ناپدید شده‌اند پیاده سازی را به کاربر می‌سپردند. کاربر با ایجاد ساختمان‌های داده پایگاه را بوسیله اشاره‌گر طراحی می‌کرد اما در مدل رابطه‌ای اشاره‌گر تنها نوع داده است که استفاده از آن ممنوع است.
- این سیستم کار طراحی را ساده‌تر می‌کند و از سوی دیگر موجب تربیت طراحان متخصص‌تر می‌شود (چون طراح درگیر مسائل ذخیره‌سازی سخت‌افزاری نمی‌شود).

انتزاع و سادگی، هم حسن و هم عیب

- بیش از حد انتزاعی نگاه گردن به پایگاه رابطه‌ای موجب می‌شود که طراحان رابطه‌ها را به شکل جدول‌های کشیده شده روی کاغذ، تاپل‌ها را به شکل سطر، ویژگی‌ها را به شکل ستون تصور کند و از درک خصوصیات و تفاوت‌های ظریف آنها عاجز شوند.
- این در حالی است که درک مدل ریاضی نظریه رابطه‌ای و به کار بردن دقیق آن لازمه طراحی یک پایگاه داده با کیفیت و استاندارد است.

انتزاع و سادگی، هم حسن و هم عیب

- سادگی کار با DBMS ها باعث شده که برخی کسانی که پایگاه طراحی می‌کنند نه فقط از اصول ریاضی لازم برای طراحی، که حتی از تنظیمات و ابزارهای خود DBMS هم بی اطلاع باشند. در واقع DBMS های امروزی مثل نرم‌افزار فتوشاپ شده‌اند، افراد حرفه‌ای و متخصص می‌توانند با استفاده از این ابزار قوی کارهای فوق‌العاده‌ای انجام دهند، در عین حال راه افراد بسیار مبتدی هم باز است تا با این ابزار کارهایی بسیار ضعیف را به انجام برسانند.
- در همین راستا زبان SQL استاندارد (و به پیروی از آن زبان تمام DBMSهای موجود) به شخص اجازه می‌دهد که برای ساده‌تر شدن کار اصول مسلم رابطه‌ای را زیر پا بگذارد.

ابزار ویرایش ساده، موجب ورود طراحان نابلد به کارها می‌شود.

در گذشته فقط عکاسان حرفه‌ای، عکس ادیت می‌کردند.

هدف از بخشی از این درس ...

آن است که با اصل مدل رابطه‌ای آشنا شویم. آنچه در اینجا خواهیم دید گرچه بیشتر تئوری است اما به شدت در عمل کاربرد دارند.

- لئوناردو داوینچی گفته: کسی که شیفته انجام تجربه بدون پشتوانه تئوری است مانند ناخدایی است که بدون سکان و قطب نما به کشتی می‌رود و نمی‌تواند اطمینان داشته باشد که از کجا سر در می‌آورد. تجربه بایستی همیشه بر پایه معلومات تئوری بنا شود.
- استاد جلال‌الدین همایی در مقدمه لغت‌نامه دهخدا می‌گوید: هر نوع علم و فنی هر چند در فطرت انسان نهفته باشد، چون جامه اصول و ضوابط پوشید جلوه و رونقی خواهد داشت و فهم آن برای کسی که نشانی از دانش و فرهنگ دارد یعنی از دانستن لذت می‌برد و از نادانی آزرده می‌شود، موجب نشاط و شکفتگی خاطرخواهد بود.

همواره گفته‌اند: قانون‌ها را باید بشناسیم و بفهمیم، حتی اگر که بخواهیم آنها را بشکنیم.

پیاده‌سازی سیستم‌های پایگاه داده

جلسه سوم

نظریه رابطه‌ای (ادامه)

پایگاه توزیع‌کنندگان و قطعات

ستون‌های مدل رابطه‌ای

- ساختار
مفهوم رابطه و سایر مفاهیم این مدل

- جامعیت
قیود و قواعدی است که در هر لحظه از زمان داده‌ها باید آن را رعایت کنند تا پایگاه درست باشد.

- کار با داده‌ها
چگونگی کوئری (درخواست) داده و به روزرسانی.

تاپل

- تعریف ریاضی: یک مجموعه است که تعداد اعضای آن مشخص بوده و اعضای آن دارای ترتیب مشخص باشند. اعضای تاپل را در پرانتز () می‌نویسند نه در آکولاد {}. مثلا: (۹۶۳۲۳۴۳، علی، رضایی)
- تعریف کاد: کاد برای نظریه رابطه‌ای تعریف تاپل را اندکی تغییر داد. تفاوت تعریف او آنست که اعضای مجموعه دارای ترتیب نیستند در عوض هر عضو دارای یک "نام ویژگی" است.
مثلا:
(شماره‌دانشجویی:۹۶۳۲۳۴۳، نام:علی، فامیل:رضایی)

رابطه = بدنه + عنوان

- بدنه مجموعه‌ای از **تاپل**هاست. به تعداد اعضای این مجموعه **کاردینالیتی** گویند. مثلا:

{stu1,stu2,stu3,....}

- **عنوان** مجموعه‌ای از ویژگی‌هاست که هر ویژگی دارای یک نام و یک دامنه است. به تعداد اعضای این مجموعه **درجه** می‌گویند. مثلا: {شماره‌دانشجویی:عدد نه رقمی، نام:رشته‌کاراکتری، فامیل:رشته‌کاراکتری}
- **رابطه** به یک بدنه و یک عنوان در کنار هم گفته می‌شود.
- رابطه یک مفهوم ریاضی است (مانند حلقه و گروه و ...) که از مدت‌ها قبل در ریاضیات نام‌گذاری شده و کاد آن را با تغییراتی در مدل خود وارد کرده است (همانند تاپل). بنابراین بر خلاف تصور عوام کلمه رابطه هیچ ارتباطی با رابطه دو جدول با ستون‌های مشترک ندارد.

رابطه از دید منطقی

- هر کدام از تاپل‌های رابطه نماینده یک گزاره هستند که با کمک عنوان تفسیر می شود (عنوان رابطه یک گزاره‌نما است، گزاره‌ای با پارامترهای نامعلوم).
- مثلا تاپل ذکر شده در مثال قبل می‌گوید:
در دانشگاه دانشجویی هست به شماره دانشجویی ۹۶۳۲۳۴۳ که نامش علی و فامیلش رضایی است.
- تمام گزاره‌هایی که از تاپل‌ها بدست می‌آیند درست هستند.
- فرض «جهان بسته» وجود دارد، یعنی تاپل‌هایی که می‌توانستند در رابطه باشند و نیستند همگی غلط فرض می‌شوند. مثلا غلط است که بگوییم دانشگاه دانشجویی با نام پژمان و فامیل اصغری دارد (وقتی تاپلی متناظر با آن در رابطه وجود ندارد).

شرط نرمال بودن

- رابطه‌ای که نرمال نباشد (در فرم اول نرمال یا 1NF) نباشد اصلا رابطه نیست.
- نرمال بودن به این معناست که هر کدام از مقدارهایی که در هر تاپل هستند اتمی (غیر قابل تجزیه) باشند. به بیان ساده در روی کاغذ در محل تلاقی هر سطر و ستون جدول فقط یک مقدار باشد نه بیشتر.
- اتمی بودن یعنی ما هرگز برای تجزیه آن مقدار اقدامی صورت ندهیم. مثلا می‌توان یک ویژگی برای نام و نام خانوادگی داشت و اگر هیچ دستوری هیچوقت در پایگاه سعی در جدا کردن نام و فامیل نکند این ویژگی اتمی است.

مطابقت داشتن با نظریه مجموعه‌ها

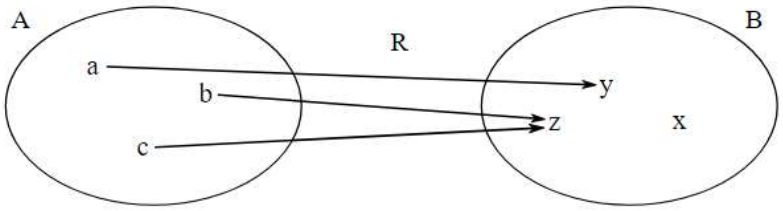
- همانطور که دیدیم رابطه بر اساس نظریه مجموعه‌ها تعریف می‌شود و این مطابقت در هنگام طراحی و اجرای پایگاه باید حفظ شود.
- در مجموعه‌ها وجود عضو تکراری بی‌معناست، در نظریه رابطه‌ای هم از این نظر نمی‌توان تاپل تکراری در رابطه داشته باشیم، هم از نظر منطقی تکرار کردن یک گزاره درست کاری بیهوده است، هم از نظر طراحی پایگاه را دچار افزونگی می‌کند.
- اما محصولات SQL از حضور تاپل تکراری (سطر تکراری) جلوگیری نمی‌کنند مگر با کلید این قانون را اعمال کنیم. با این حال کلید بر دیدها (که خودشان رابطه‌اند) اعمال نمی‌شود.
- اعضای مجموعه ترتیبی ندارند. در اینجا هم تاپل‌ها ذاتا ترتیبی ندارند مگر آنکه زمان چاپ بخواهیم آنها را مرتب کنیم.

جدول، سطر و ستون

- اینها نام‌های ساده‌ای است که به ترتیب به رابطه، تاپل و ویژگی داده شده است. در واقع وقتی این مفاهیم ریاضی روی کاغذ بیایند ظاهر آنها به شکل جدول و سطر و ستون خواهد بود در حالی که طراح پایگاه باید همیشه موقع کار مفاهیم ریاضی آنها را مد نظر داشته باشد، نه شکل کاغذی‌شان را.
- همانطور که ریه شما با عکس رادیوگرافی که از آن برداشته‌اید فرق دارد، ذات رابطه هم با تصویر آن که روی کاغذ به شکل جدول است فرق دارد.

جدول، سطر و ستون

- مثلا می‌دانیم که تاپل‌ها در رابطه فاقد ترتیب هستند (مثل تعدادی میوه که در یک کیسه ریخته شده و دائما بهم می‌خورند) اما وقتی آنها را به شکل جدول روی کاغذ می‌کشیم این بی‌ترتیبی نمی‌تواند تصویر شود و سطرهای جدول به هر حال ترتیبی دارند.
- همچنین این تصور که رابطه‌ها دقیقا همان فایل‌های معمولی کامپیوتری هستند (فقط با تجرید و انضباط) را باید از ذهن دور کرد.
- البته اشکالی در استفاده از اسامی ساده نیست، به شرطی که مفاهیم ریاضی را بدانیم و فکر نکنیم رابطه یعنی همان جدول. اگر کسی فرق جدول و رابطه و تاپل و سطر را اصلا نداند و اقدام به طراحی پایگاه کند کارش جاهلانه است.



R = {(a,y), (b,z), (c,z)}

دامنه یا نوع

- هر کدام از از مقادیر داده از یک دامنه معین هستند. به عبارت دیگر برای هر ویژگی یک استخر خیالی وجود دارد که مقدار داده را می‌توان از آن انتخاب کرد.
- مثلا در مورد جنسیت دانشجو در این استخر فقط دو توپ هست: زن و مرد. اما در مورد شماره دانشجویی تمام اعداد نه رقمی که دو رقم اول آن سال‌هایی که دانشگاه ورودی داشته، اینجا **دامنه** همان استخری خیالی است که همه شماره دانشجویی‌های ممکن در آن هست.
- در این نگاه رابطه‌ها دیگر جدول‌های دو بعدی نیستند، بلکه هر رابطه به تعداد ویژگی‌هایش (درجه‌اش) بعد دارد. وجود هر تاپل مترادف رسم یک نقطه در فضای n بعدی است و فضای خالی تاپل‌هایی است که وجود ندارند.
- شکل فوق یک رابطه دوتایی (با تعریف ریاضی آن) است که عنصر اول از دامنه A و دومی از دامنه B گرفته می‌شود.

دامنه یا نوع

- نوع داده (ویژگی‌ها) در پایگاه داده با انواع عمومی داده در کامپیوتر متفاوت است. مثلا داده‌های نوع شماره دانشجویی را ممکن است کسی از نوع عمومی عدد int تعریف کند. البته به علت محدود بودن کامپیوتر، انواع عمومی هم (مانند استخر) تعداد محدودی عضو دارند.
- ممکن است نوع دو ویژگی (مثلا شماره دانشجویی و کد ملی) به ظاهر با هم یکی باشد اما چون نوع داده آنها متفاوت است سیستم نباید اجازه مقایسه آنها را با هم بدهد (در برنامه نویسی به یاد بیاورید که 5 از نوع int و ’5‘ از نوع char با هم ظاهرا یکی بودند اما سیستم اجازه مقایسه آنها را نمی‌داد).
- استفاده کامل از نوع‌ها (مانند برنامه‌نویسی شی‌گرا) سیستم را مستحکم می‌کند. متاسفانه امکانات SQL در این مورد محدود است و طراحان غالبا فقط از انواع عمومی استفاده می‌کنند.
- استفاده کامل از دامنه همه جا ممکن نیست و خیلی جاها چاره‌ای جز استفاده از نوع‌های عمومی (مثل رشته حرفی) نیست. اما در جاهایی که می‌شود نوع را محدود کرد (مثل خانم/آقا) باید این کار را کرد.

تهی

- تهی به جای یک مقدار داده می‌تواند در پایگاه‌های داده به کار رود. استفاده از آن به دو دلیل کاملا متفاوت ممکن است. یکی زمانی که یک تاپل یکی از ویژگی‌ها را اساسا ندارد، مثلا ویژگی نام همسر در حالی که شخص اصلا زن (یا شوهر) ندارد. دوم زمانی که مقدار آن ویژگی نامعلوم است. مثلا نمی‌شود که کسی اصلا کدپستی نداشته باشد، اما برای کسی که کدپستی‌اش را اعلام نکرده هم می‌توان تهی وارد کرد.
- چون اکثرا معلوم نیست کدام حالت مورد نظر بوده، نمی‌توان گزاره مربوطه را از تاپل بدست آورد. همچنین در حالت دوم سیستم منطقی به سه سطحی تغییر می‌کند (درست/غلط/نمی‌دانم) و این در حالی است که مدل رابطه‌ای بر پایه منطق دوسطحی است.

تهی

- تهی یکی از موارد بحث‌انگیز در مدل رابطه‌ای است. تهی یکی از پایه‌های نظریه مجموعه‌ها است که خود زیربنای مدل رابطه‌ای است.
- کاد در ابتدا تهی را در مدل رابطه‌ای قرار نداد، اما حدود ده سال بعد آن را به مدل اضافه کرد. با این حال برخی همچون دیت دلایلی دارند که نباید از آن استفاده شود.
- حتی الامکان بهتر است که از تهی دوری کنیم. در جای خود نحوه حذف کامل تهی از پایگاه‌های داده را خواهیم دید.

تمایز بین متغیر رابطه‌ای و رابطه

- این دو مفهوم گاهی با هم اشتباه می‌شوند زیرا به خوبی جا نیافتاده‌اند.
- یک رابطه بدون توجه به محتوایش (که این محتوا در طول زمان تغییر می‌کند) یک متغیر رابطه‌ای است (مثل رابطه‌ای بنام students). این دقیقا مشابه برنامه‌نویسی است که یک متغیر integerی داشته باشیم (مثلا m که از نوع int است بدون توجه به محتوایش)
- متغیر رابطه‌ای را به اختصار مرابطه relvar می‌نامند.

تمایز بین متغیر رابطه‌ای و رابطه

- یک رابطه با توجه به محتوایی که در یک لحظه خاص دارد یک مقدار رابطه‌ای یا بطور خلاصه رابطه است (مثل students با همین دانشجوهای در این لحظه دانشگاه). مثال برنامه‌نویسی متغیر m با مقدار فعلی 7 .
- به یاد داشته باشید که کسی ممکن است بپرسد این چیست؟ یکی بگوید لیوان و دیگری بگوید آب طالبی. متغیر رابطه‌ای و رابطه هم به همین صورت است و مفهوم‌شان نباشد مخلوط شود.

جبر رابطه‌ای

- جبر سیستمی است ریاضی که در آن فورمول‌نویسی نه فقط برای محاسبه اعداد (2+4/2) بلکه همچنین برای محاسبه نمادهایی که بجای مقادیر نامشخص می‌نشینند (x=3+y/3) به کار می رود.
- زبان جبر برای فورمول‌نویسی‌های محاسباتی و جبری بسیار خلاصه‌تر و دقیق‌تر از زبان طبیعی (مثل فارسی) است ولی قدرت بیان آن (برای همه مطالب از جمله مطالب غیر ریاضی) از زبان طبیعی بسیار کمتر است.
- مشهورترین نوع جبر، جبر اعداد معمولی است اما سیستم‌های جبری دیگری نیز توسط ریاضی‌دان‌ها مطرح شده مانند جبر منطقی (بول) که پایه مدارات دیجیتال است.
- جبر رابطه‌ای در همان اولین مقاله توسط کاد ارائه شد. جبر رابطه‌ای مانند همه جبرها دارای یک مجموعه مقادیر مجاز (که رابطه‌ها هستند با تعریف گفته شده) و عملگرهای مربوطه (هشت عملگر جبری) می‌باشد.

عملگرهای جبر رابطه‌ای (یادآوری)

- در اینجا سه عملگر جبر رابطه‌ای که در عمل کاربرد گسترده دارند را روی یک پایگاه داده رابطه‌ای ساده (شامل دو رابطه) که مربوط به بخش‌ها و کارکنان یک سازمان است را می‌بینیم (از عملگرهای اجتماع، اشتراک، تفاضل، تقسیم و ضرب دکارتی صرفنظر می‌کنیم):

Dept	Dname	Budget
D1	بازاریابی	100 میلیارد تومان
D2	طرح و توسعه	120 میلیارد تومان
D3	پژوهش	50 میلیارد تومان

Emp	Ename	Dept	Salary
E1	لطفی	D1	33 میلیون تومان
E2	جالنگی	D1	35 میلیون تومان
E3	فیضی	D2	25 میلیون تومان
E4	سعادت	D2	29 میلیون تومان

گزینش (محدودیت)

- این عملگر تکی فقط بعضی تاپل‌های رابطه را برمی‌گرداند (مثلا کارمندانی که بیش از ۳۰ میلیون تومان حقوق می‌گیرند) نوشته می‌شود
WORKERS WHERE (salary>30)

Emp	Ename	Dept	Salary
E1	لطفی	D1	33 میلیون تومان
E2	جالنگی	D1	35 میلیون تومان
E3	فیضی	D2	25 میلیون تومان
E4	سعادت	D2	29 میلیون تومان

پرتو

- مانند آن است که فقط بر روی برخی ستون‌ها پرتوی از نور بتابانیم تا فقط آنها دیده شوند. مثلا فقط نام کارمند و حقوق او.
نوشته می‌شود WORKERS [Ename,Salary]

Emp	Ename	Dept	Salary
E1	لطفی	D1	33 میلیون تومان
E2	چالنگی	D1	35 میلیون تومان
E3	فیضی	D2	25 میلیون تومان
E4	سعادت	D2	29 میلیون تومان

پیوند

- یعنی دو جدول را بر اساس مقدارهای مشترک که در ستون مشترک هست ارتباط دهیم (با ضرب دکارتی تمام تاپل‌ها با هم زوج شده سپس فقط آنهایی که مقدار مشترک دارند گزینش می‌شوند).

Emp	Ename	Dept	Salary
E1	لطفی	D1	میلیون تومان 33
E2	چالنگی	D1	میلیون تومان 35
E3	فیضی	D2	میلیون تومان 25
E4	سعادت	D2	میلیون تومان 29

Dept	Dname	Budget
D1	بازاریابی	میلیارد تومان 100
D2	طرح و توسعه	میلیارد تومان 120
D3	پژوهش	میلیارد تومان 50

Workers JOIN Departments

Dept	Dname	Budget	Emp	Ename	Salary
D1	بازاریابی	میلیارد تومان 100	E1	لطفی	میلیون تومان 33
D1	بازاریابی	میلیارد تومان 100	E2	چالنگی	میلیون تومان 35
D2	طرح و توسعه	میلیارد تومان 120	E3	فیضی	میلیون تومان 25
D2	طرح و توسعه	میلیارد تومان 120	E4	سعادت	میلیون تومان 29

متغیرهای رابطه‌ای پایه و دیدها

- رابطه‌هایی پایه هستند بصورت اولیه در سیستم بوده‌اند یعنی از رابطه‌های دیگر بدست نیامده‌اند.
- در مقابل رابطه‌های پایه، رابطه‌های مشتق شده را داریم. مثلا در یک پرتو یا یک پیوند حاصلش یک رابطه جدید است، اما این جدول مشتق شده است نه پایه.
- در سیستم‌های رابطه‌ای می‌توان رابطه‌های مشتق شده را تعریف و با یک نام ذخیره نمود که «دید» یا view نامیده می‌شود.
- این مجموعه (جدول‌ها و دیدها) تشکیل یک سیستم ریاضی می‌دهد (مثل هندسه). می‌توان فرض کرد رابطه‌های پایه اصول موضوعه (دانسته‌های بدون استدلال) هستند و با قواعد استنتاج (فورمول دیدها و کوئری‌ها) می‌توان قضایای جدید (دانش جدید یعنی نتیجه دیدها و کوئری) را بدست آورد.

متغیرهای رابطه‌ای پایه و دیدها

- دیدها مرابطه مشتق شده یا مرابطه مجازی هم نامیده می‌شوند.
- دیدها دقیقا مانند رابطه‌های پایه هستند با این تفاوت که مقدار آنها بصورت یک کپی جداگانه در سیستم ذخیره نشده و از رابطه‌های پایه در همان لحظه محاسبه می‌شود.
- اصل تعویض‌پذیری می‌گوید اینکه کدامیک از مرابطه‌ها پایه و کدامیک مجازی باشند، اختیاری است. بنابراین نبایستی هیچ برتری دلخواه و غیر ضروری میان مرابطه‌های پایه و مجازی وجود داشته باشد (مثلا می‌توان برای دانشجویان هر دانشکده یک رابطه تعریف کرد و بعد آنها را اجتماع کرد یا اینکه برای همه دانشجوها یک رابطه تعریف کرد و بعد با عملگر گزینش آنها را به رابطه‌های مجزا تبدیل نمود).

پایگاه‌داده توزیع کنندگان و قطعات

- این پایگاه مثال استاندارد تمام کتاب‌های دیت است که با وجود سادگی به دقت طراحی شده و برای یادگیری و تمرین مناسب است.
- تعدادی قطعه (مثل پیچ و مهره و ...) با رنگ و وزن مشخص تولید می‌شوند و هر کدام نیز یک شهر دارند که در آن تولید و انبار شده‌اند.
- تعدادی توزیع کننده که دارای نام و رتبه (مثل تعداد ستاره) و شهر هستند، وجود دارند.
- تعدادی سفارش ثبت شده هست که نشان می‌دهد کدام قطعه به چه تعدادی از کدام توزیع کننده سفارش داده شده است.

تمرین: ساخت پایگاه داده کامل

- بعنوان تمرین هفته قبل، پایگاه را بصورت ساده در SQLServer ساختید (اگر نساخته‌اید الان بسازید چون تمام تمرین‌های از این به بعد وابسته به آن خواهد بود و رابطه‌ها باید در آن ایجاد شود.)
- طرح پایگاه باید به گونه‌ای انجام شود که مناسب پذیرش داده‌های فارسی باشد. جدول‌های جدید اضافه کنید. فقط داده‌ها فارسی‌اند. برای جلوگیری از مشکلاتی آتی مشخصات سیستمی مثل اسم رابطه و ویژگی (جدول و ستون) انگلیسی نوشته شود.
- نوع CHAR برای داده انگلیسی (کد اسکی) با طول ثابت است. افزون VAR به ابتدای آن طول را متغیر (حداکثر به اندازه طول معین شده) و افزودن N آن را از نوع مناسب برای الفبای غیر انگلیسی (یونی کد) می‌نماید.
- پس از تکمیل عنوان جدول‌ها داده‌ها را مطابق آنچه در اسلاید است وارد کنید.

کلید (اصلی و خارجی) را تعریف نکنید. این موضوع جلسه بعد است. برای ساخت رابطه‌ها و ورود دیتا لازم نیست حتما کوئری بنویسید.

عنوان و نام رابطه‌ها

DESKTOP-OM1TN1B\...sParts - dbo.Part

Column Name	Data Type	Allow Nulls
PID	int	☐
PName	nvarchar(10)	☐
Color	nvarchar(5)	☐
Weight	real	☐
City	nvarchar(10)	☐

DESKTOP-OM1TN1B\...ts - dbo.Supplier

Column Name	Data Type	Allow Nulls
SID	int	☐
SName	nvarchar(10)	☐
Status	int	☐
City	nvarchar(10)	☐

DESKTOP-OM1TN1B\...s - dbo.Shipment

Column Name	Data Type	Allow Nulls
SID	int	☐
PID	int	☐
Qty	int	☐

پایگاه توزیع‌کنندگان و قطعات

DESKTOP-OM1TN1B\...sParts - dbo.Part

PID	PName	Color	Weight	City
1	مهره	قرمز	12	لاهیجان
2	پیچ	سبز	17	پاکدشت
3	پیچ خودکار	آبی	17	اسکو
4	پیچ خودکار	قرمز	14	لاهیجان
5	پیچ چفتی	آبی	12	پاکدشت
6	چرخ دنده	قرمز	19	لاهیجان

DESKTOP-OM1TN1B\...ts - dbo.Supplier

SID	SName	Status	City
1	عصمتی	20	لاهیجان
2	جنانی	10	پاکدشت
3	بلاغتی	30	پاکدشت
4	کلانتری	20	لاهیجان
5	آوانسیان	30	آشتیان

TOP-OM1TN1B\...s - dbo.Shipment

SID	PID	Qty
1	1	300
1	2	200
1	3	400
1	4	200
1	5	100
1	6	100
2	1	300
2	2	400
3	2	200
4	2	200
4	4	300
4	5	400

پیاده‌سازی سیستم‌های پایگاه داده

جلسه چهارم

جامعیت

روال‌های ذخیره شده

Stored Procedures

جامعیت

- جامعیت ترجمه نارسایی است برای کلمه integrity ، معادل فارسی کاملی برای این کلمه وجود ندارد اما به معنای شخص یا سیستمی است که با تمام وجود سعی دارد به درستی کار کند (در مورد شخص معادل کاردرستی، شرف و یکرویی است).
- قیدهای جامعیت قواعدی (گزاره‌هایی) هستند که در طول عمر پایگاه داده باید همواره برقرار باشند. داده‌ها (مقدار رابطه‌ها) دائما در حال تغییر است اما این تغییرات در هیچ لحظه‌ای باعث نمی‌شود این قیدها زیر پا گذاشته شوند.

جامعیت

- پایگاه داده نمی‌تواند درستی داده‌ها را تضمین کند چون بر مرحله نمونه‌برداری از دنیای واقعی نظارتی ندارد. اما می‌تواند جامعیت را تضمین کند (از ورود داده‌های ناجور جلوگیری کند). نداشتن جامعیت به معنای وجود داده‌های آشکارا نادرست در پایگاه است.
- **قیدهای جامعیت عمومی** (برای همه پایگاه‌ها) دو تا هستند: جامعیت وجودی و جامعیت ارجاعی. اولی به معنای نقض نشدن کلید اصلی و دومی به معنای نقض نشدن کلید خارجی است.
- **قیدهای جامعیت اختصاصی** (خاص برای هر پایگاه) هم می‌تواند وجود داشته باشد.

کلید

- مرابطه R را در نظر بگیرید. اگر K زیرمجموعه‌ای از عنوان آن باشد، K کلید کاندید (یا به اختصار کلید) R است اگر و تنها اگر دارای هر دو خاصیت زیر باشد:

یکتایی
غیر ممکن است در R دو تاپل مجزا باشد که دارای مقدار یکسان برای K باشند. (مثلا شماره دانشجویی تکراری نداریم)

کاهش ناپذیری
هیچ زیرمجموعه‌ای از K خاصیت یکتایی ندارد. (مثلا شماره دانشجویی همراه شماره تلفن نمی‌تواند کلید باشد)

- کد ملی و (شماره شناسنامه، محل صدور، سال صدور) نمونه کلید یک یا چندتایی هستند.

کلید

- کلید خاصیتی است که متعلق به مرابطه است نه رابطه، چرا که مقدار فعلی (بدنه) رابطه نیست که کلید را تعیین می‌کند، بلکه کلید است که می‌گوید چه چیزی می‌تواند یا نمی‌تواند در (بدنه) رابطه باشد.
- کلید (= کلید کاندید) خاصیتی است که بطور ذاتی در مرابطه (= جدول) هست و طراح در وجود آن نقشی ندارد. کلید اصلی یکی از کلیدهای کاندید است که طراح آن را بطور دلخواه بعنوان اصلی انتخاب می‌کند. این انتخاب اکثر اوقات دلیل مدیریتی و غیر فنی دارد (مثل انتخاب شماره دانشجویی و نه کد ملی در سیستم دانشگاه).
- هر مرابطه به طور ذاتی دارای کلید است، چون اگر هیچ زیر مجموعه‌ای از عنوان خاصیت کلید نداشته باشد، دست کم کل عنوان چنین خاصیتی دارد (چون در رابطه هیچوقت تاپل تکراری نداریم).
- جامعیت وجودی در صورتی برقرار است که مقدار کلید اصلی (آن کلیدی که ما به صورت دلخواه انتخاب می‌کنیم) در هر تاپلی نه تهی باشد نه با تاپل دیگر یکسان.

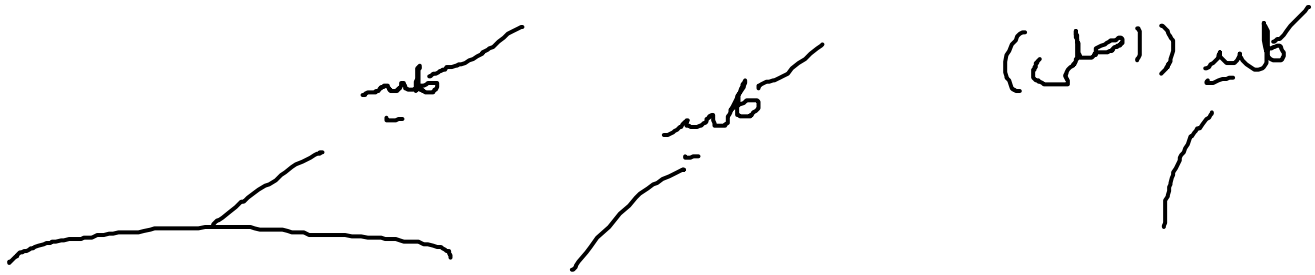
کلید خارجی

- زمانی که در R1 تاپلی که دارای متناظر در R2 است می‌خواهد حذف شود یا مقدار کلید آن به روزرسانی شود سیستم برای نقض نشدن قاعده کلید خارجی دو راهکار دارد:

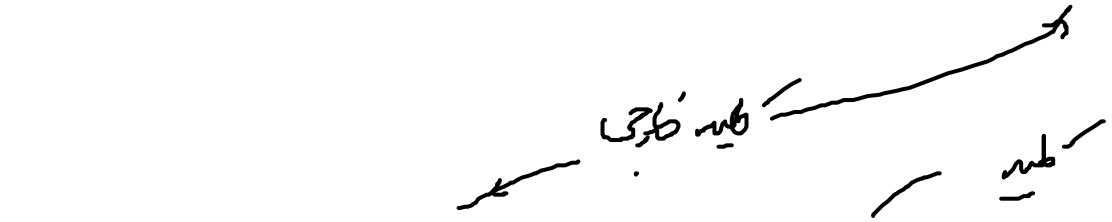
کلید خارجی

- زمانی که در R1 تاپلی که دارای متناظر در R2 است می‌خواهد حذف شود یا مقدار کلید آن به روزرسانی شود سیستم برای نقض نشدن قاعده کلید خارجی دو راهکار دارد:
یکی اینکه جلوی انجام این کار را بگیرد.
دیگری اینکه کلیه تاپل‌های متناظر را حذف یا بروزرسانی کند.

مثالی از کلید



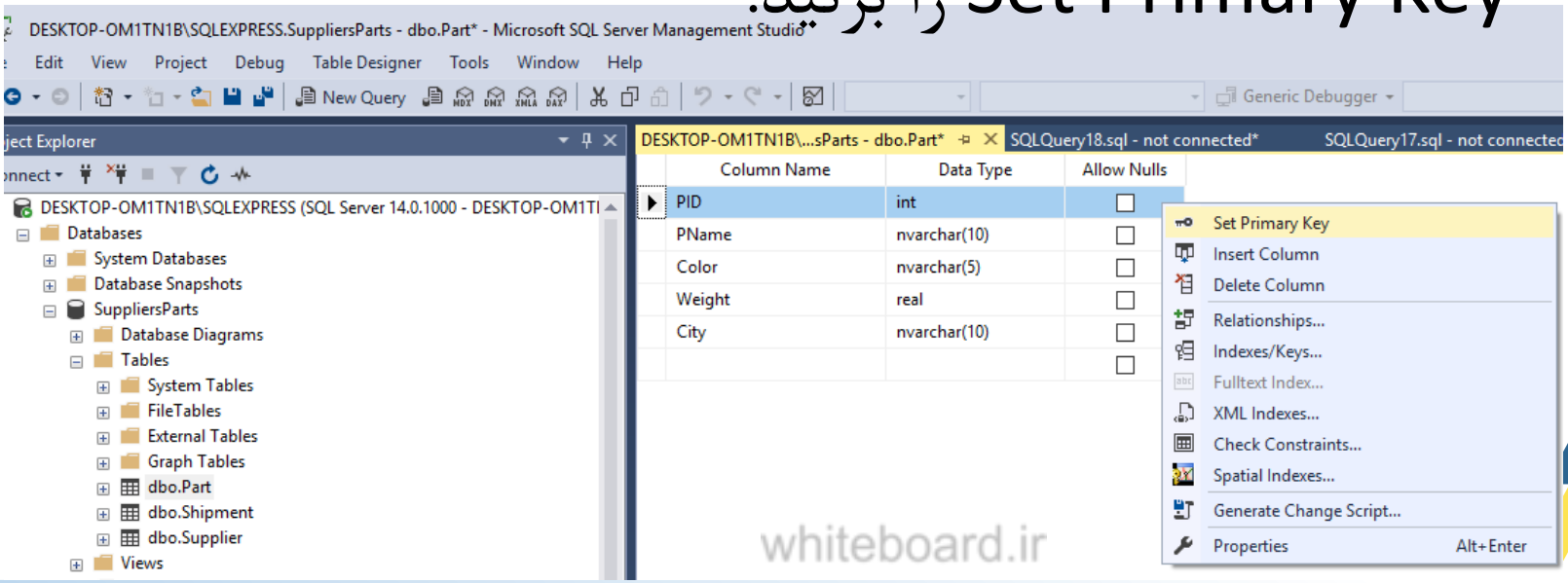
شماره دانشجویی	نام	نام خانوادگی	کد ملی	شماره شناسنامه	محل صدور	سال تولد



سریال پرداخت	شماره دانشجویی	مبلغ	تاریخ

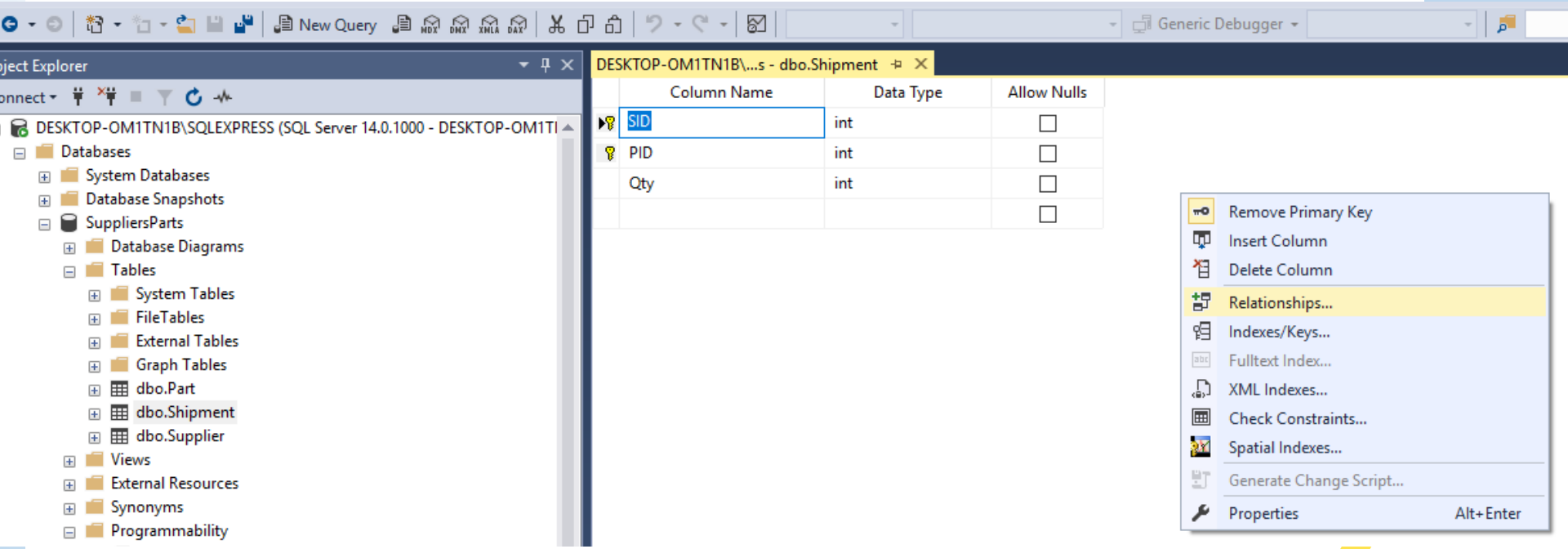
الزام به قید جامعیت وجودی

- الزام به قید جامعیت وجودی ساده است. سیستم مدیریت پایگاه DBMS هیچ سطر تازه ای (یا به روزرسانی‌ای) را قبول نمی‌کند اگر بخشی از کلید اصلی آن تهی باشد. همچنان که اگر مقدار تکراری باشد هم آن را نمی‌پذیرد.
- در SQLServer برای تنظیم کلید اصلی روی ویژگی مربوطه (یا انتخاب چند ویژگی با ctrl) کلیک کنید و Set Primary Key را بزنید.



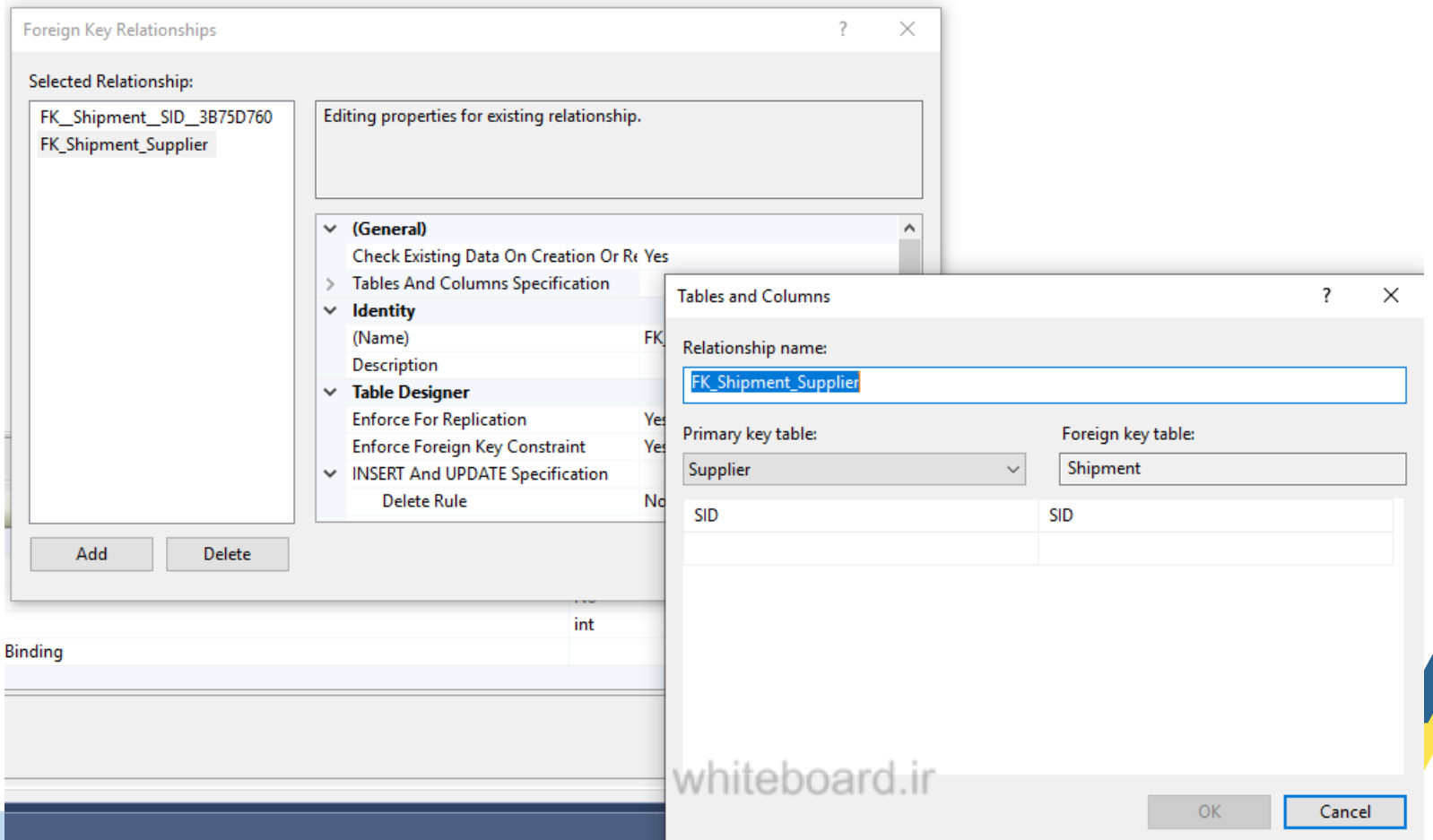
الزام به قید جامعیت ارجاعی

- در SQLServer زمانی که در قسمت طراحی جدول هستید روی صفحه کلیک راست کنید و Relationships را انتخاب کنید.



الزام به قید جامعیت ارجاعی

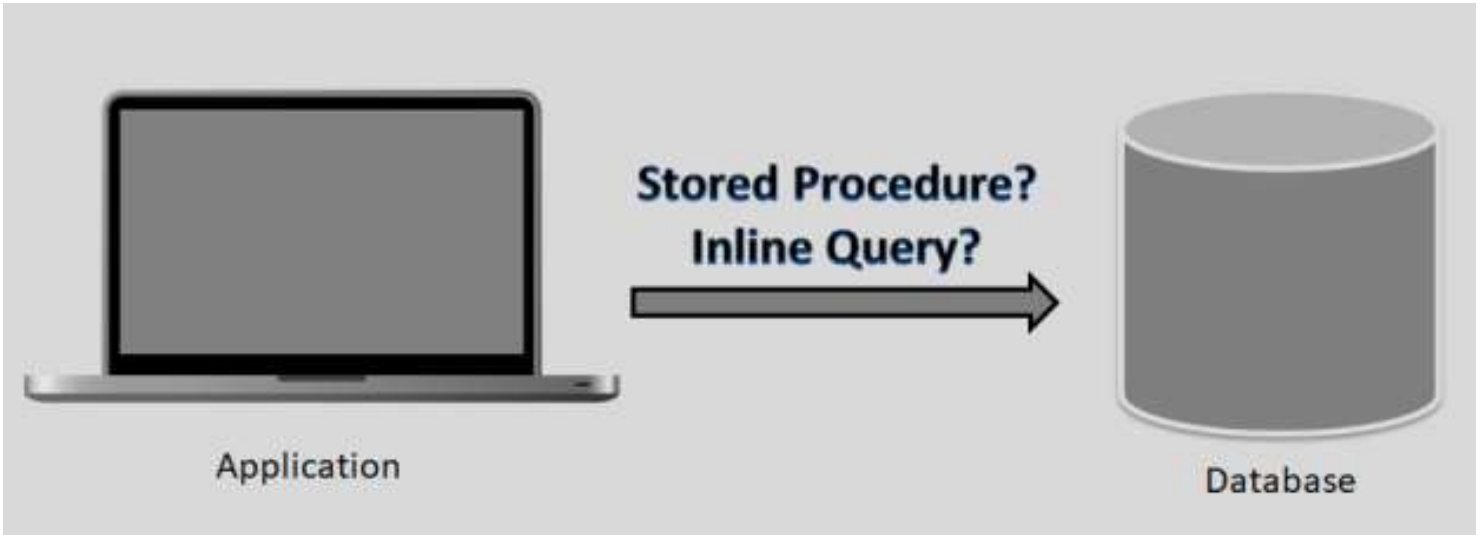
- Add را بزنید و سپس با زدن "..." در table and columns specification ویژگی‌های جدول کلیدخارجی و کلید اصلی را انتخاب کنید.
- قسمت INSERT and UPDATE Specification جایی است که در آن سیاست به روزرسانی (اسلاید صفحه ۸) را تعیین می‌کنید (جلوگیری یا حذف).



استور(د) پروسیجر چیست

- استور پروسیجر روالی است که به عنوان جزئی از پایگاه داده در DBMS ذخیره می‌شود و می‌تواند با اجرای کوئری‌ها در قالبی شبیه زبان‌های برنامه‌نویسی با پایگاه داده کارکند.
- کوئری‌ها را می‌توان به صورت ساده در پایگاه ذخیره کرد که به آن ویو (دید) می‌گویند، اما استور پروسیجر (علاوه بر کدهای SQL) دارای دستورات مرحله به مرحله بوده و دستورات برنامه‌نویسی همچون if و while و تعریف و مقداردهی متغیر می‌توان در آن وجود داشته باشد.

تفاوت استورپروسجر و کوئری



- Query:
  Select …. From …. Where …
- Procedure:
  run p1()

مزایای استورپروسجر نسبت به کوئری

برنامه‌های کاربردی از دو راه با پایگاه ارتباط برقرار می‌کنند یکی ارسال دستورات SQL یا اجرای ویوها و دیگری اجرای استور پروسجرها.

مزایا:

۱- کارایی: با اولین اجرا محتوای پروسیجر parse شده و query plan ساخته می‌شود و دفعات بعد همین‌ها مورد استفاده قرار می‌گیرند (شبیه کامپایل شدن برنامه)، اما هنگام اجرای کوئری هر بار این کارها باید انجام گیرد.

۲- بار شبکه و بار کاری کامپیوترهای کاربران: بدون استفاده از استور پروسجر در بسیاری موارد باید حجم بیشتری داده از کامپیوتر کاربر به پایگاه منتقل شود و برنامه کاربردی پردازش‌هایی انجام دهد.

مزایای استورپروسجر نسبت به کوئری

۳- محافظت از کپی رایت: استورپروسیجر ها را می‌توان رمزگذاری کرد که در این صورت هیچکس نخواهد توانست کد آنها را ببیند و تغییر دهد (بر خلاف اپلیکیشن‌ها قابل قفل شکستن نیست) و از این راه با برخی تکنیک‌ها مانند کنترل قفل سخت افزاری در پروسیجر، محدودیت در تاریخ یا حجم داده یا قرار دادن نام مشتری در جواب برخی کوئری‌ها کل سیستم حفاظت خواهد شد.

۴- ارسال و دریافت پارامتر: کوئری‌ها (بصورت ویو) مستقیما امکان دریافت پارامتر ندارند و فقط یک رابطه (جدول) را بر می‌گردانند.

۵- امنیت: می‌توان دسترسی مستقیم کاربر به جدول‌ها را ممنوع کرد و فقط از طریق اجرای استورپروسجرها به او اجازه کار با پایگاه را داد. در این صورت طراحی نقشه امنیتی پایگاه داده آسانتر می‌شود و فقط کاربر به آنچه لازم است دسترسی داشته باشد، دسترسی دارد.

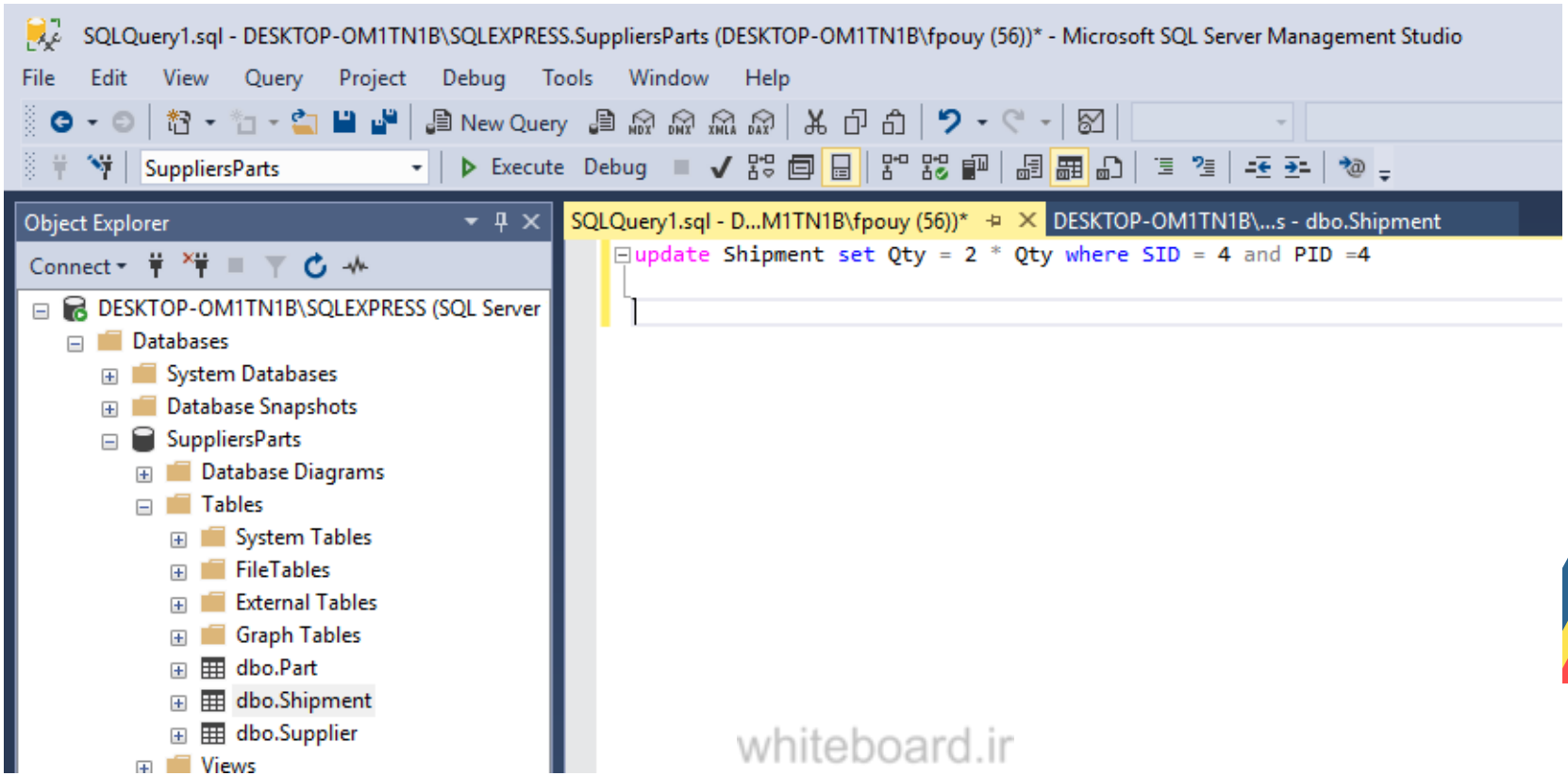
همچنین یکی از خطرات بالقوه که پایگاه را تهدید می‌کند، حملات از طریق تزریق کد SQL، از پایگاه داده دور خواهد شد.

مثال

- یک کوئری بنویسید که در جدول سفارش‌ها تعداد سفارش‌های توزیع‌کننده شماره ۴ قطعه شماره ۴ را دو برابر کند (در قسمت new query نوشته و با execute اجرا می‌شود.).

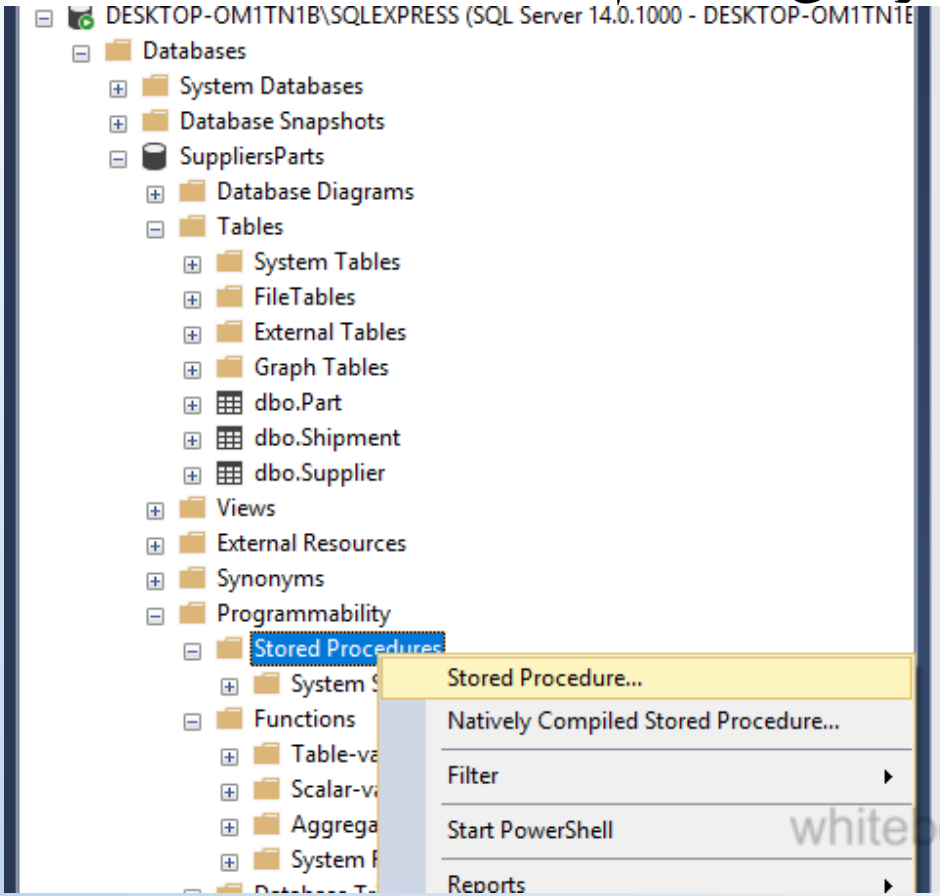
مثال

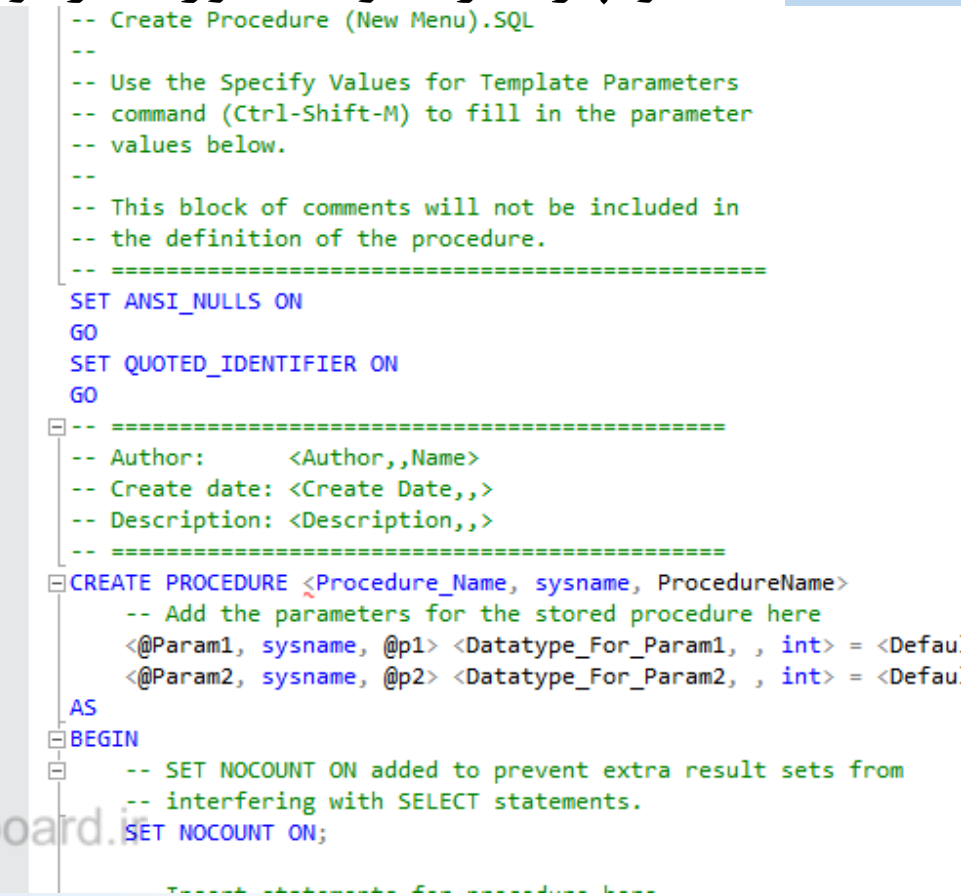
- یک کوئری بنویسید که در جدول سفارش‌ها تعداد سفارش‌های توزیع‌کننده شماره ۴ قطعه شماره ۴ را دو برابر کند (در قسمت new query نوشته و با execute اجرا می‌شود.).



نوشتن استور پروسیجر

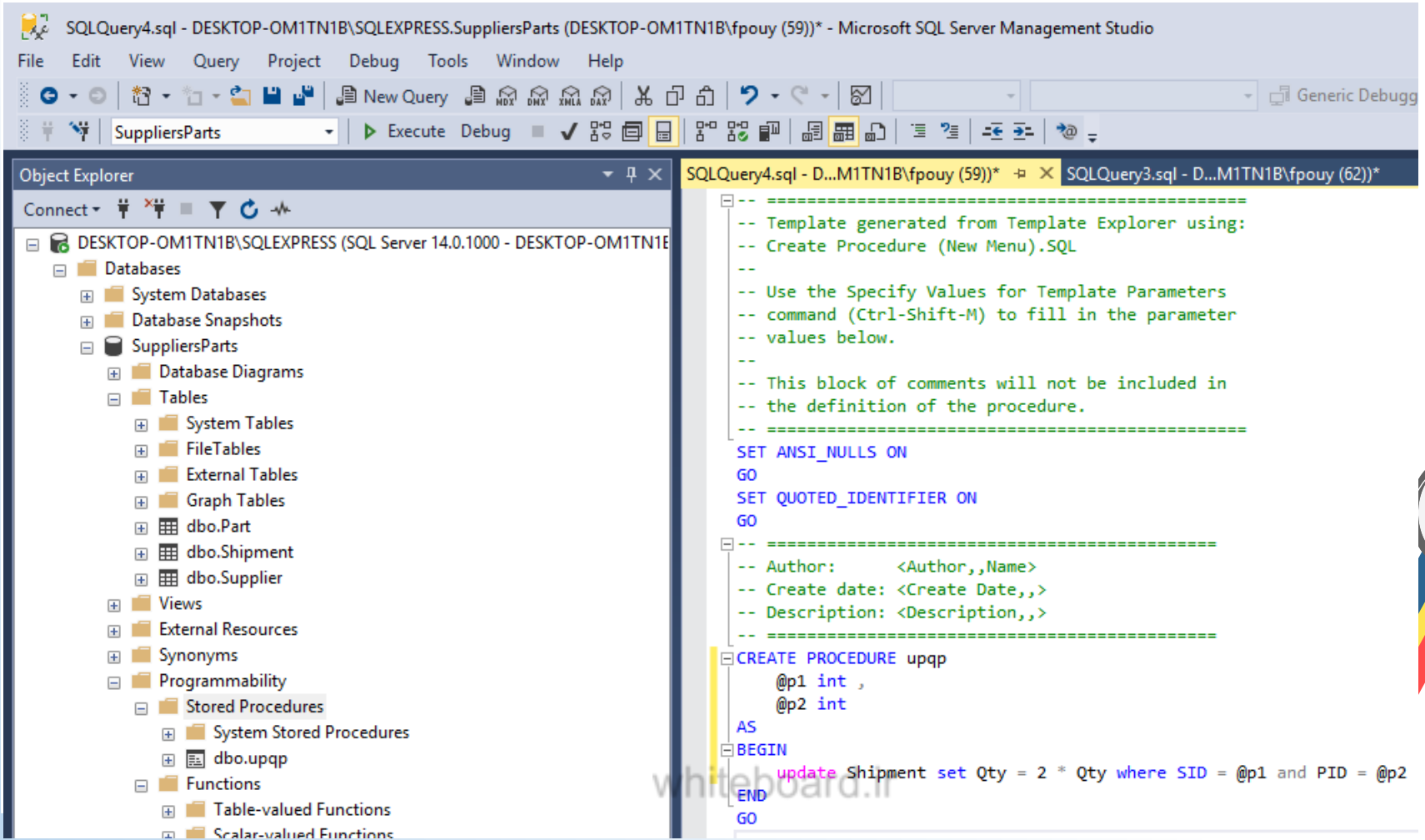
- با کلیک راست روی stored procedures و بعد انتخاب stored procedure... یک پنجره با یک پروسیجر نمونه باز می‌شود که دارای دو پارامتر نمونه و یک دستور select نمونه است. باید ابتدا برای پروسیجرمان یک نام مناسب انتخاب کنیم و پارامترها و دستورات را ویرایش نماییم.





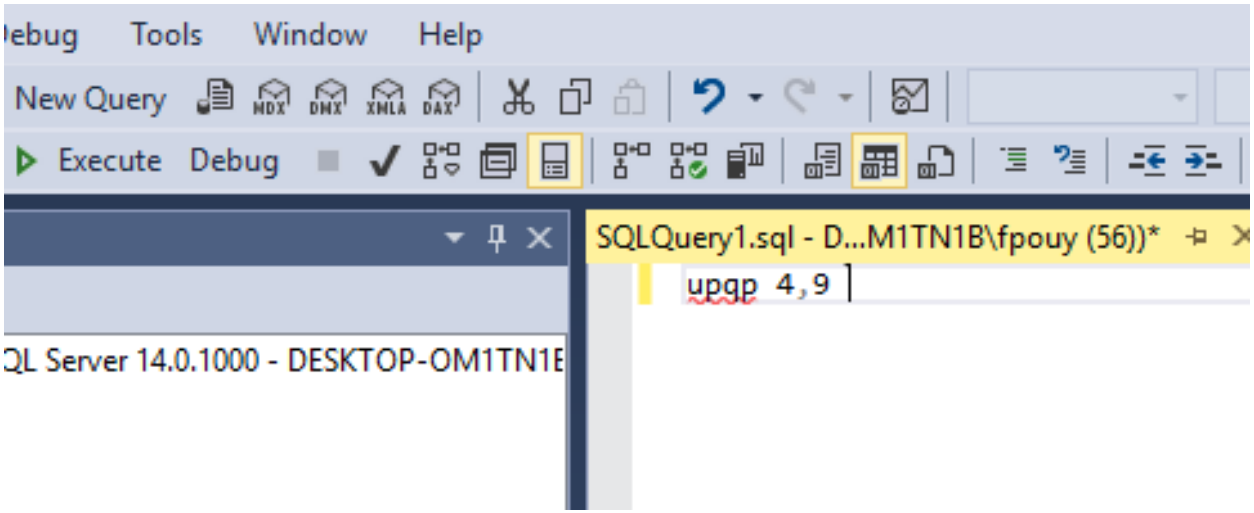
مثال

- برای مثال دو اسلاید قبل استور پروسجر بنویسید (شماره قطعه و توزیع‌کننده پارامتری باشد).



ادامه مثال

- همانطور که دیدید این پروسیجر را با نام upqp و با دو پارامتر نوشتیم، یعنی کوئری می‌توانست فقط همان رکورد را تغییر دهد اما اینجا می‌توان شماره توزیع کننده و شماره قطعه را به صورت پارامتر فرستاد، برای اجرای پروسجر کافیست این کوئری ساده را اجرا کنیم:



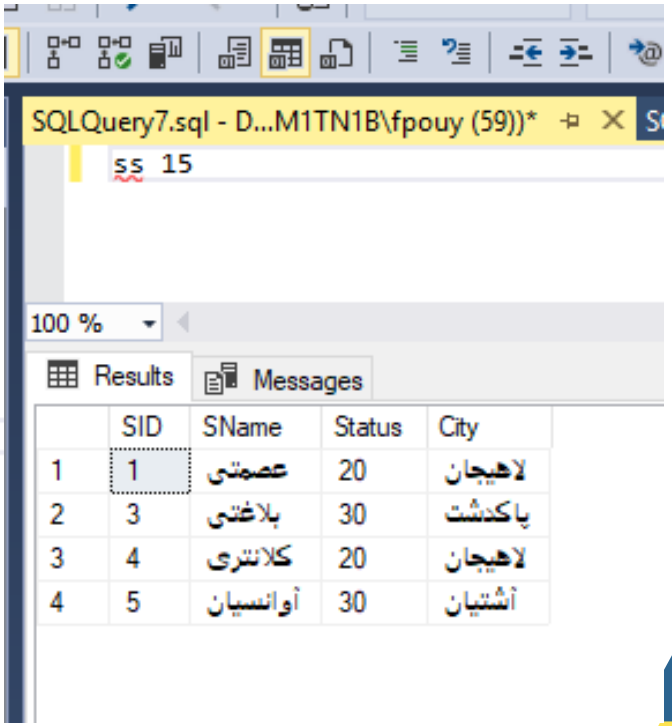
پروسیجرهای درخواست اطلاعات

- استور پروسیجرها فقط برای به روزرسانی اطلاعات نیستند. آنها می‌توانند یک رابطه را برگردانند. مثلا پروسیجر زیر یک پارامتر عددی دارد و پس از اجرا جدولی را بر می‌گرداند که حاوی تمام توزیع‌کنندگانی است که اعتبارشان بیشتر از آن عدد است.

```
CREATE PROCEDURE ss
    -- Add the parameters for the stored procedure here
    @p1 int= 0
    AS
BEGIN

select * from Supplier where Status >= @p1

END
GO
```



	SID	SName	Status	City
1	1	عصمتی	20	لاهیجان
2	3	بلاغتی	30	پاکدشت
3	4	کلانتری	20	لاهیجان
4	5	آوانسیان	30	آشتیان

تعریف متغیر در پروسیجر

- در استور پروسیجر می‌توان متغیر اسکالر (تک مقداری) داشت (به عنوان پارامتر یا غیر پارامتر). به غیر از آنها متغیرهای رابطه‌ای (جدولی) هم می‌توانیم داشته باشیم.
- همانطور که در جلسات قبل دیدیم، جدول‌های موجود در پایگاه هر کدام در واقع یک متغیر رابطه‌ای هستند.
متغیرهای قابل تعریف پروسیجر هم به این متغیرها اضافه می‌شوند. از آنجا که متغیرهای تعریف شده همیشه با @ شروع می‌شوند، هرگز با جدول‌ها و دیدهای (که دائمی و ذخیره شده در سیستم هستند) موجود مخلوط نمی‌شوند.
- اینجا داده نوع آرایه نداریم چون "نوع جدول" همان کار را بسیار بهتر انجام می‌دهد.

مثال

- پروسیجری بنویسد که یک عدد را دریافت کند و تعداد سفارشاتی که کوچکتر از آن عدد هستند را به همراه توزیع کننده‌ای که سفارش به او داده شده را بازگرداند. رابطه‌های توزیع‌کنندگان و سفارش‌ها باید ابتدا در دو متغیر ذخیره شوند و بعد بقیه اعمال روی آن دو متغیر صورت پذیرد.

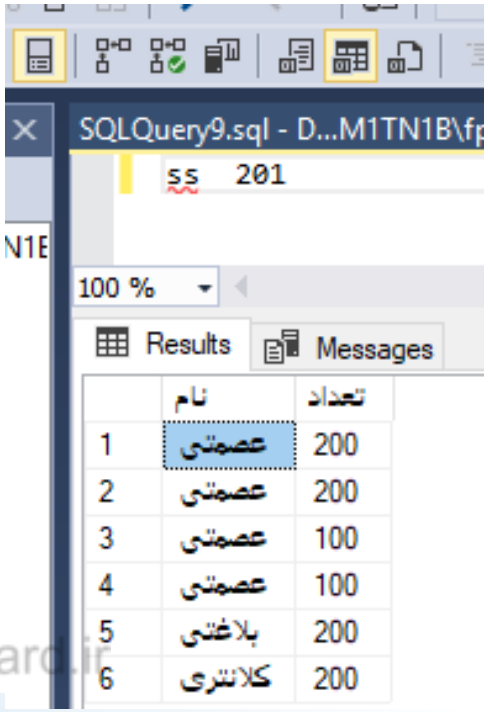
```
ALTER PROCEDURE [dbo].[ss]
    -- Add the parameters for the stored procedure here
@p1 int= 0
AS

BEGIN
declare  @t1 table (a int,b nvarchar(10))
declare  @t2 table (a int,b int, c int)

insert into @t1 select SID, Sname from Supplier
insert into @t2 select * from Shipment

select X.b as نام, Y.c as تعداد
from @t1 X inner join @t2 Y on X.a = Y.a
where Y.c < @p1

END
```



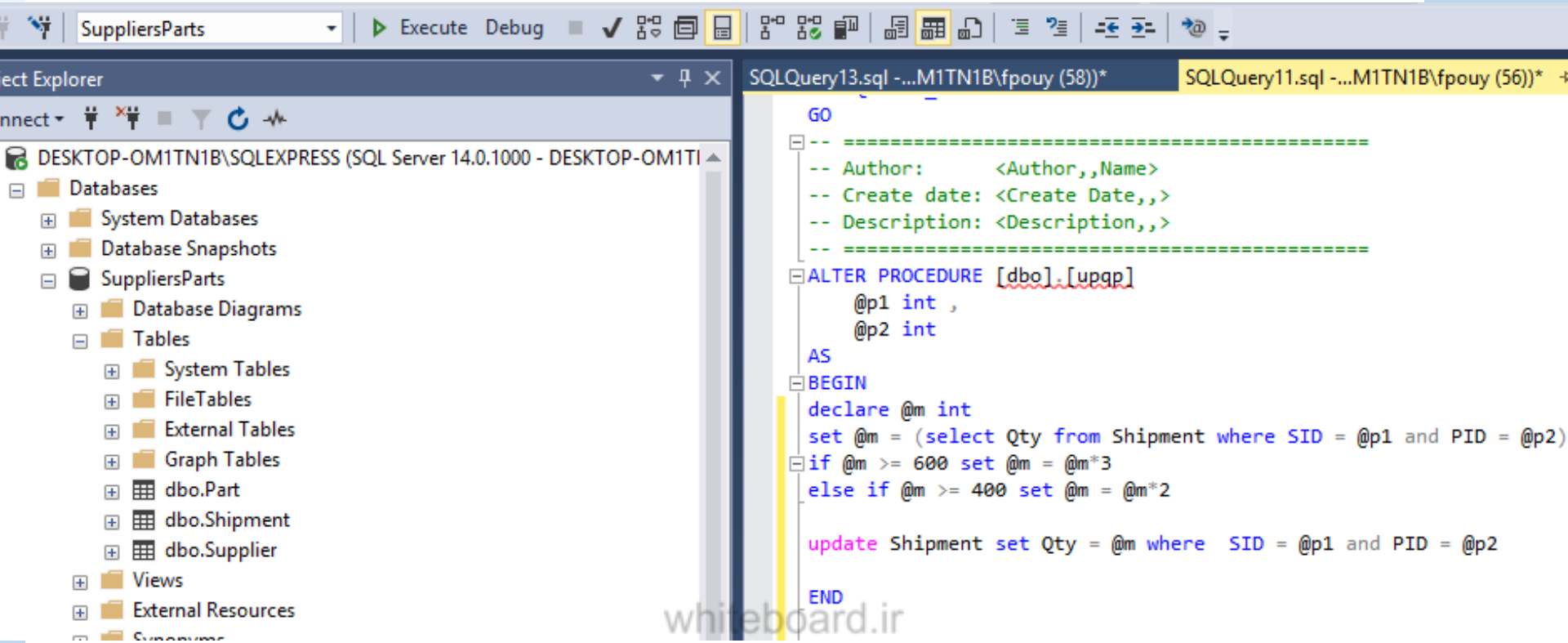
	نام	تعداد
1	عصمتی	200
2	عصمتی	200
3	عصمتی	100
4	عصمتی	100
5	بلاغتی	200
6	کلانتری	200

دستورات کنترل جریان برنامه

- استور پروسیجرها مخلوطی هستند از کوئری‌ها و دستورات برنامه‌نویسی. وقتی برنامه داشته باشیم، دستورات کنترل جریان برنامه هم خواهیم داشت. این دستورات از جمله while و if به صورت شرطی اجرای دستورات را کنترل می‌کنند و همچنین دستور goto برای کنترل غیر شرطی و waitfor برای تاخیر انداختن اجرای دستور (بر اساس ساعت و دقیقه) به کار می‌روند.
- بلاک‌های دستورات (که در C داخل آکولاد بودند) در اینجا بین begin و end قرار می‌گیرند.

مثال

- برنامه‌ای بنویسید که شماره توزیع کننده و قطعه را دریافت کند و اگر تعداد سفارش مربوطه بیش از ۶۰۰ است آن را سه برابر و اگر بیش از ۴۰۰ است آن را دو برابر کند.



توابع اسکالر

- استور پروسیجر فقط کاری روی دیتا انجام می‌دهد ولی مقداری بر نمی‌گرداند (فقط اگر دارای دستور select باشد یک رابطه برمی‌گرداند).
- در مقابل یک تابع اسکالر یک مقدار تکی (نه جدولی) را برمی‌گرداند. عملکرد تابع اسکالر شبیه عملکرد تابع‌های معمولی در زبان‌های برنامه‌سازی است.
- ساخت تابع اسکالر شبیه ساخت استور پروسیجر است با این تفاوت که scalar-valued function را انتخاب می‌کنیم.

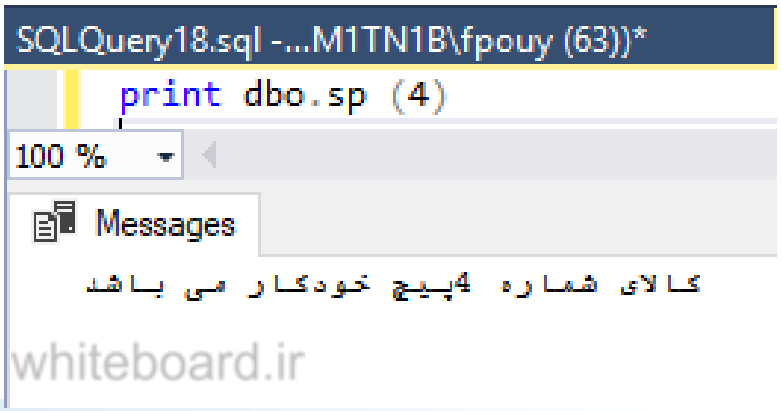
مثال

- تابعی بنویسید که یک شماره قطعه را بگیرد و یک جمله فارسی برگرداند که در آن شماره قطعه و نام آن بیان شده باشد.

```
ALTER FUNCTION [dbo].[sp]
(
@p1 int
)
RETURNS nvarchar (100)
AS
BEGIN
    -- Declare the return variable here
    DECLARE @rs nvarchar (100)

    -- Add the T-SQL statements to compute the return value here
    set @rs = (select PName from Part where PID = @p1)
    set @rs = N'کالای شماره ' + CAST(@p1 as nvarchar(20)) + @rs + N' می باشد'
    -- Return the result of the function
    RETURN @rs

END
```



پیاده‌سازی سیستم‌های پایگاه داده

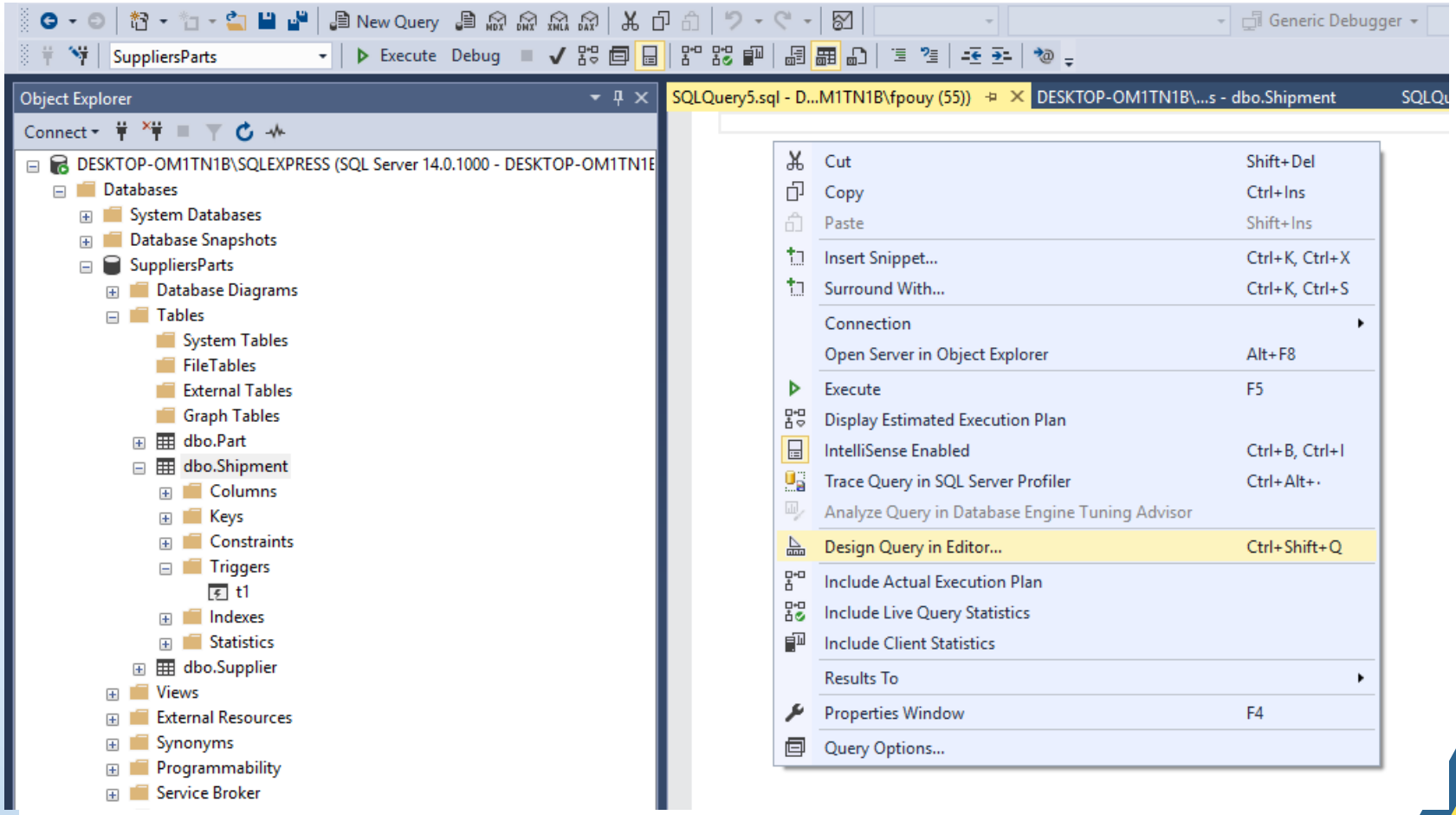
جلسه پنجم

قیود غیر عمومی
ساختار XML

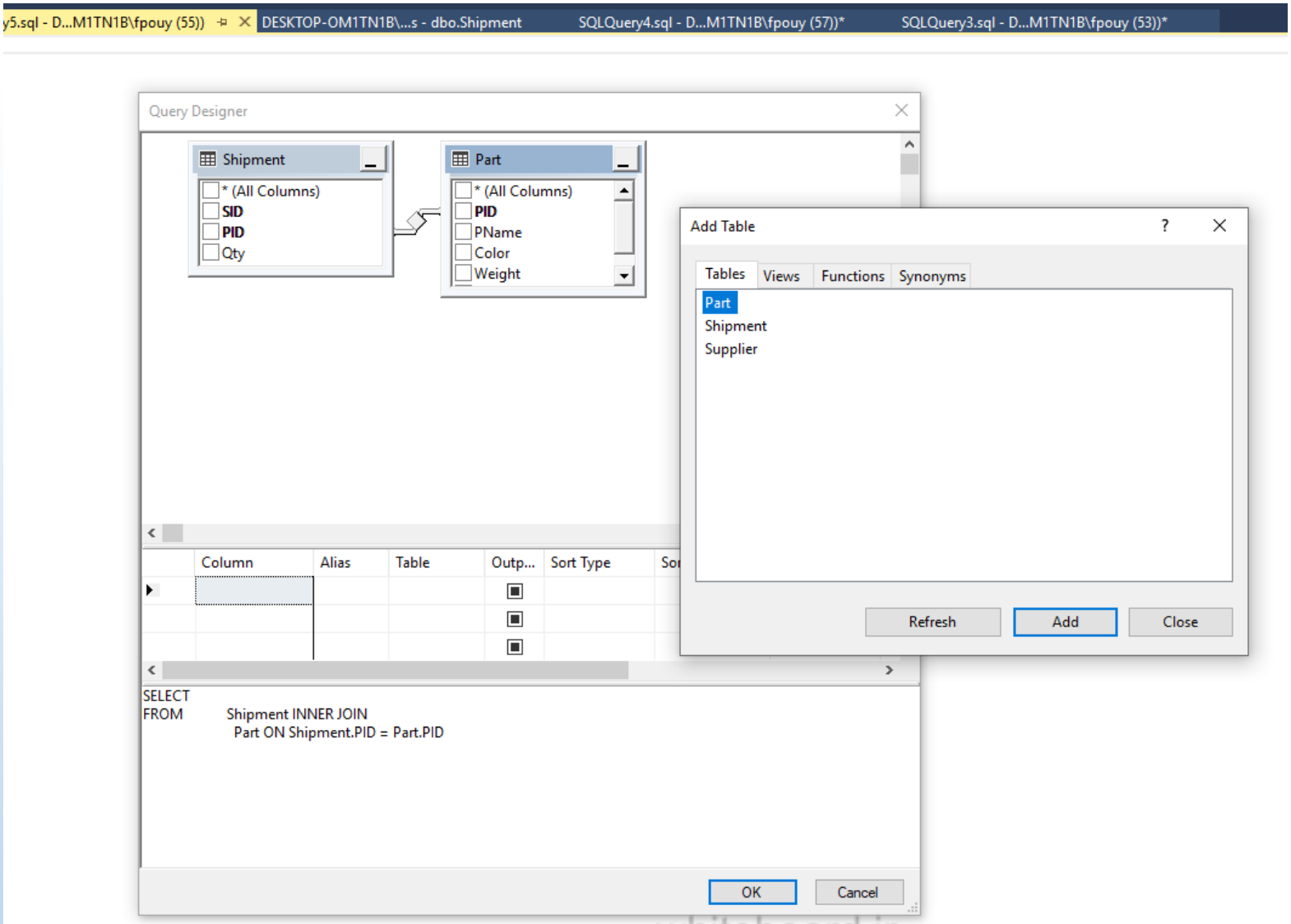
ایجاد دستورات SQL با ویزارد

- هنگام گذراندن درس پایگاه داده‌ها ما با سینتکس و نحوه نوشتن کوئری‌ها به زبان SQL آشنا شدیم.
- اما برخی از ما هنگام کار ترجیح می‌دهیم بجای تایپ کردن، این دستورات را با ویزارد گرافیکی (در SSMS یا هر رابط دیگری) ایجاد کنید.
- مثلا می‌خواهیم اسامی قطعات را و کل تعداد سفارش‌هایی که برای آنها ثبت شده را پیدا کنیم.

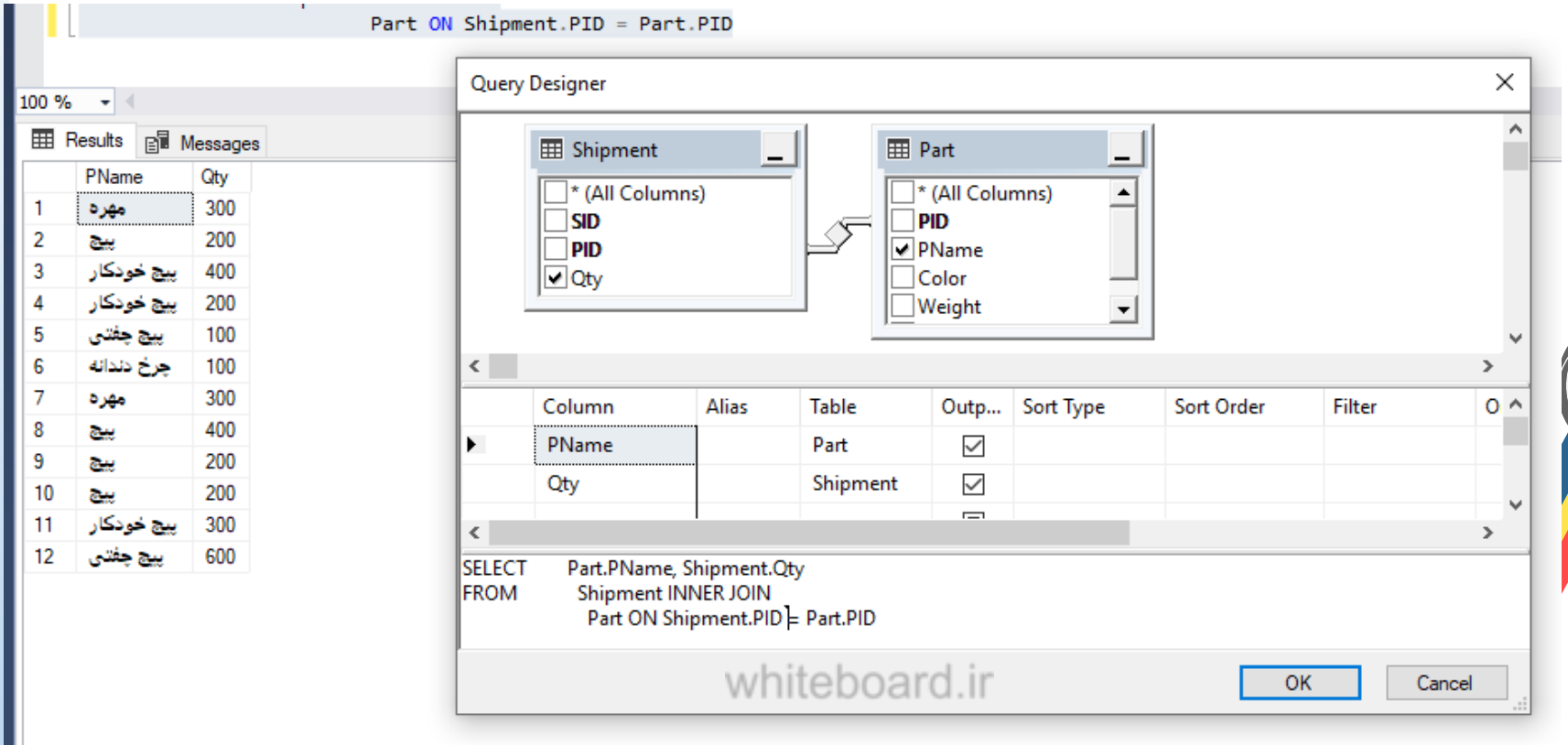
- یک کوئری جدید (new query) باز می‌کنیم و بجای تایپ کردن در فضای سفید کلیک راست می‌کنیم و ....



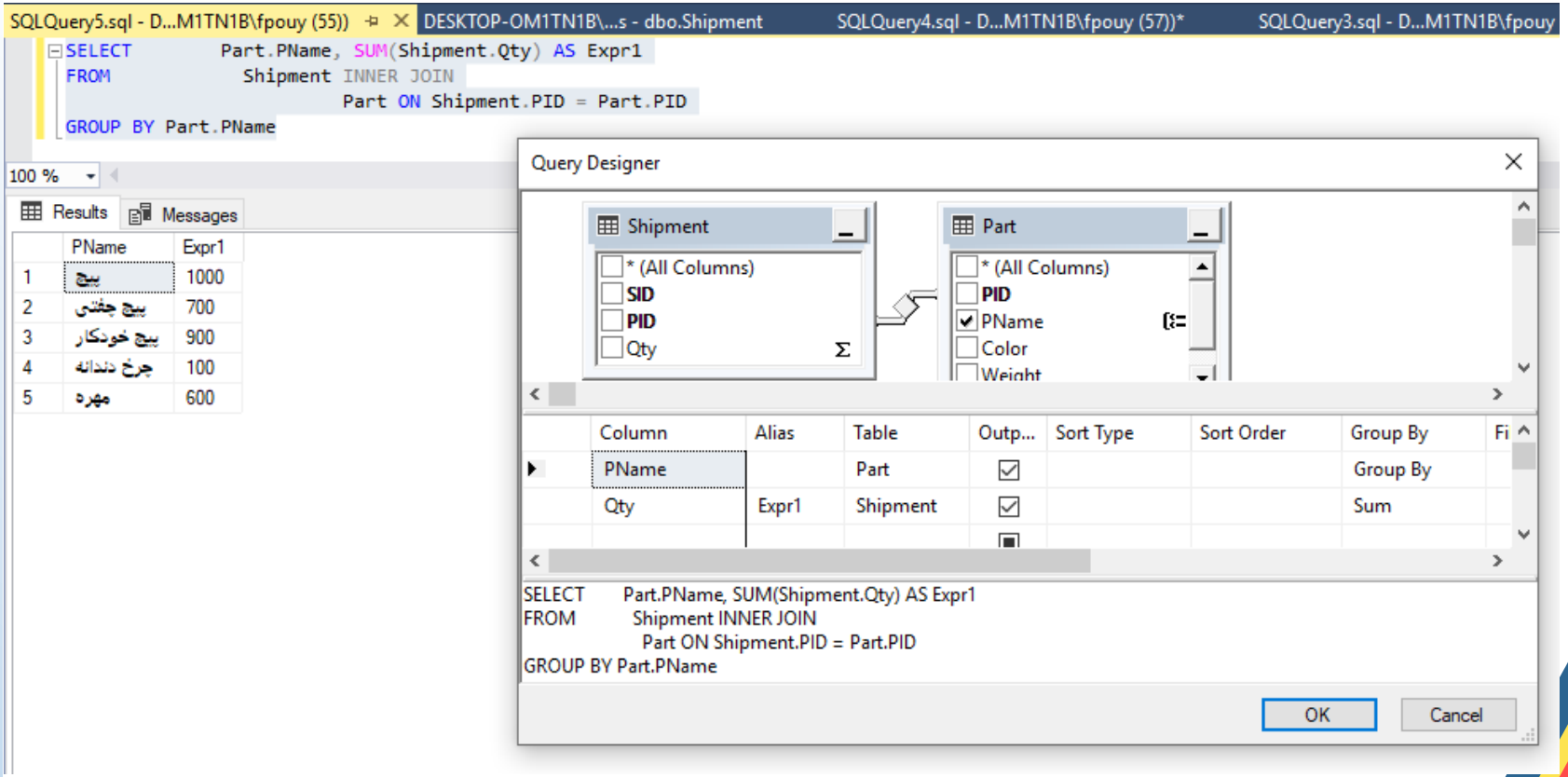
- روی جدول هایی که نیازشان داریم دبل کلیک می‌کنیم تا در بالای دیزاینر ظاهر شوند (توزیع‌کنندگان و قطعات)



- ما به نام قطعه و تعداد سفارش نیاز داریم. از آنجا که قبلا جدول‌ها با کلید خارجی مرتبط شده‌اند، با تیک زدن آنها دو جدول پیوند می‌خورند (OK کنید و نتیجه اجرا را ببینید. بعد می‌توانید دستور SQL را با ماوس select کنید و با کلیک راست دوباره به دیزاینر بیایید و تغییراتی ایجاد کنید)



- برای بدست آورن جمع تعدادها، نیاز به تابع جمعی داریم. کلیک راست کنید و add group by را بزنید. ستون تعداد را به sum تغییر دهید.



قیدها

- قید (به معنای بند و زنجیر): داده‌هایی که قرار است در پایگاه ثبت شوند را مجبور می‌کند که از مقرراتی پیروی کنند.
- در غیر اینصورت پایگاه تغییرات درخواست شده (درج، به‌روزرسانی، حذف) را قبول نمی‌کند.

الزام پایگاه به رعایت قیود غیر عمومی

- قبلا دیدیم که چگونه پایگاه را تنظیم کنیم تا داده‌ها را ملزم به رعایت **قیود جامعیت عمومی** (کلید اصلی و خارجی) کند. قیدهای عمومی در همه پایگاه‌های داده بدون استثنا برقرارند.
- در برخی موارد پایگاه دارای قواعد جامعیتی غیر عمومی (غیر دو مورد فوق) است. مثلا در سیستم دانشگاه، دانشجو بیش از ۲۴ واحد در ترم ندارد و یا در سیستم حسابداری، در هر لحظه (بعد از هر تراکنش) باید حساب‌ها تراز باشند. یعنی:

موجودی + هزینه‌ها + بدهکاران = سرمایه + بستانکاران

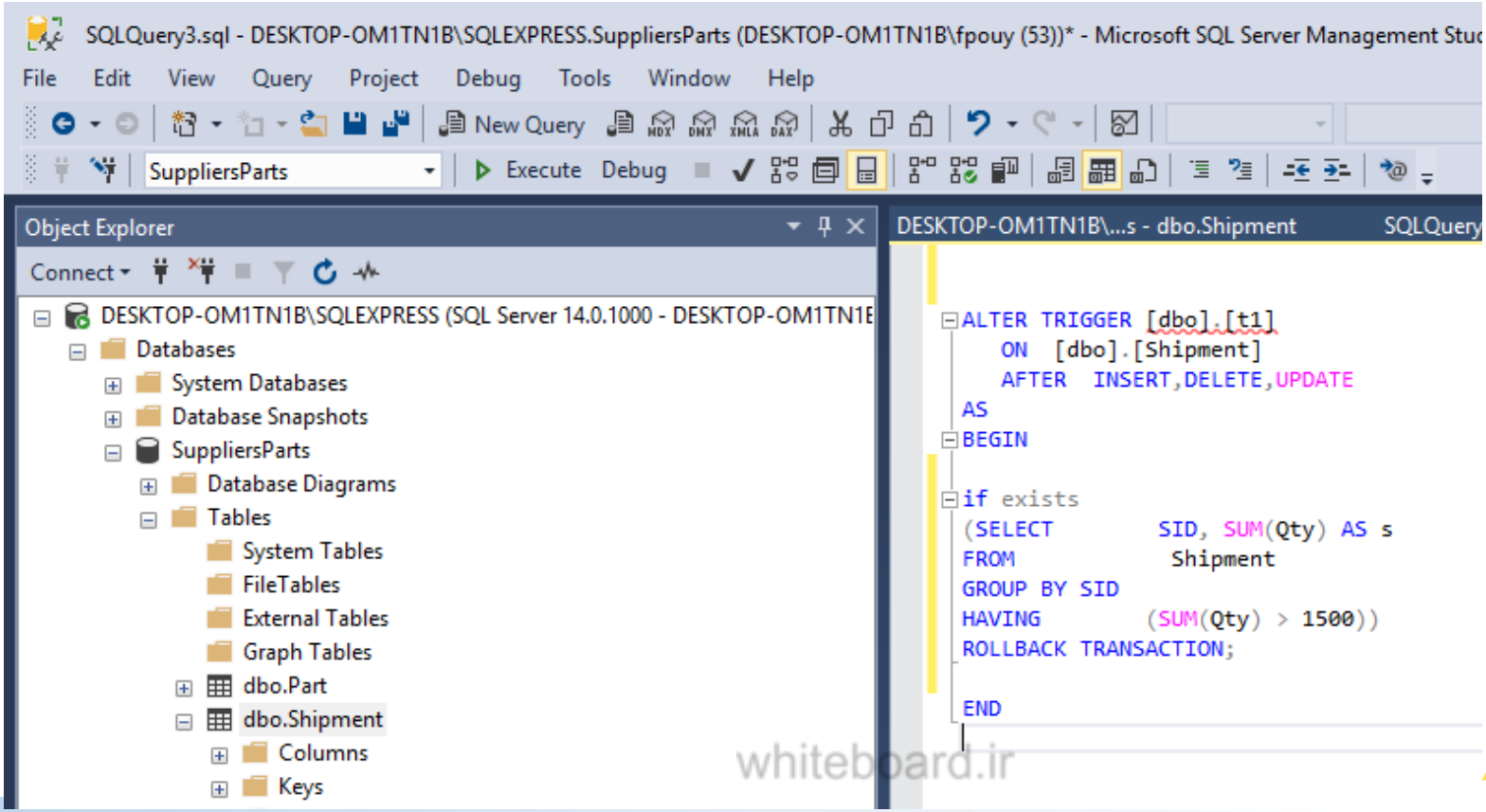
الزام پایگاه به رعایت قیود غیر عمومی

- هر تراکنشی که این معادله را بهم بریزد (مثلا برداشتی از صندوق انجام شود که معلوم نشود پولش کجا رفته) غیر قابل قبول است.
- هر چند در سطح اپلیکیشن می‌توان این موارد را کنترل کرد، اما برای اطمینان از اینکه هیچ اشتباهی صورت نمی‌گیرد، پایگاه بلافاصله بعد از دستورات تغییر داده‌ها (درج، به‌روزرسانی، حذف) می‌باید شرایطی را کنترل و در صورت تخطی، دستور به روز رسانی اجرا شده را کان لم یکن کند.

استفاده از تریگر

- برای تعریف قیودی که DBMS ملزم به رعایت آن است روی جدول‌ها تریگر تعریف می‌کنیم.
- تریگر کدی است که بعد از تغییر جدول اجرا می‌شود. در حین اجرا شرایطی کنترل می‌شود و در مواردی دستور rollback transaction کل تغییرات در جدول را واگردان می‌کند، انگار که هیچ دستور بروزرسانی نرسیده است.
- **مثلا می‌خواهیم جمع کل تعداد سفارش‌های هیچ توزیع‌کننده‌ای بیش از ۱۵۰۰ نشود.**

- همانطور که می‌بینید تریگری تعریف کردیم بنام t1 روی جدول سفارشات که هنگام هر سه عمل درج و حذف و تغییر روی این جدول اجرا می‌شود.
- بعد از هر اجرای تریگر اگر توزیع‌کننده‌ای باشد که در مجموع بیش از ۱۵۰۰ سفارش داشته باشد، عمل انجام شده بی اثر شده و جدول به حال قبل بر می‌گردد.



زبان‌های نشانه‌گذاری (مقدمه XML)

- **زبان نشانه‌گذاری متنی** روشی است برای نشانه‌گذاری (یا حاشیه‌نویسی) درون متن، به گونه‌ای که به صورت سیستماتیک از متن اصلی قابل تفکیک باشد.
- از حدود چهل سال پیش استفاده محدودی از این روش آغاز شد مثلا در نوشتن کدهای ژنتیک و نیز نرم‌افزارهای ویرایش متن و صفحه‌بندی که سرانجام به لاتک منتهی شد.
- با ایجاد شبکه اینترنت، نیاز به روشی بود که بر روی کامپیوترهای متفاوت و نرم‌افزارهای browser مختلف نمایش نوشتارها و متون را به شکل قابل قبولی نشان دهد. در اواسط دهه هفتاد شمسی برای این منظور HTML ارائه شد که باز هم نوعی Markup Language بود.

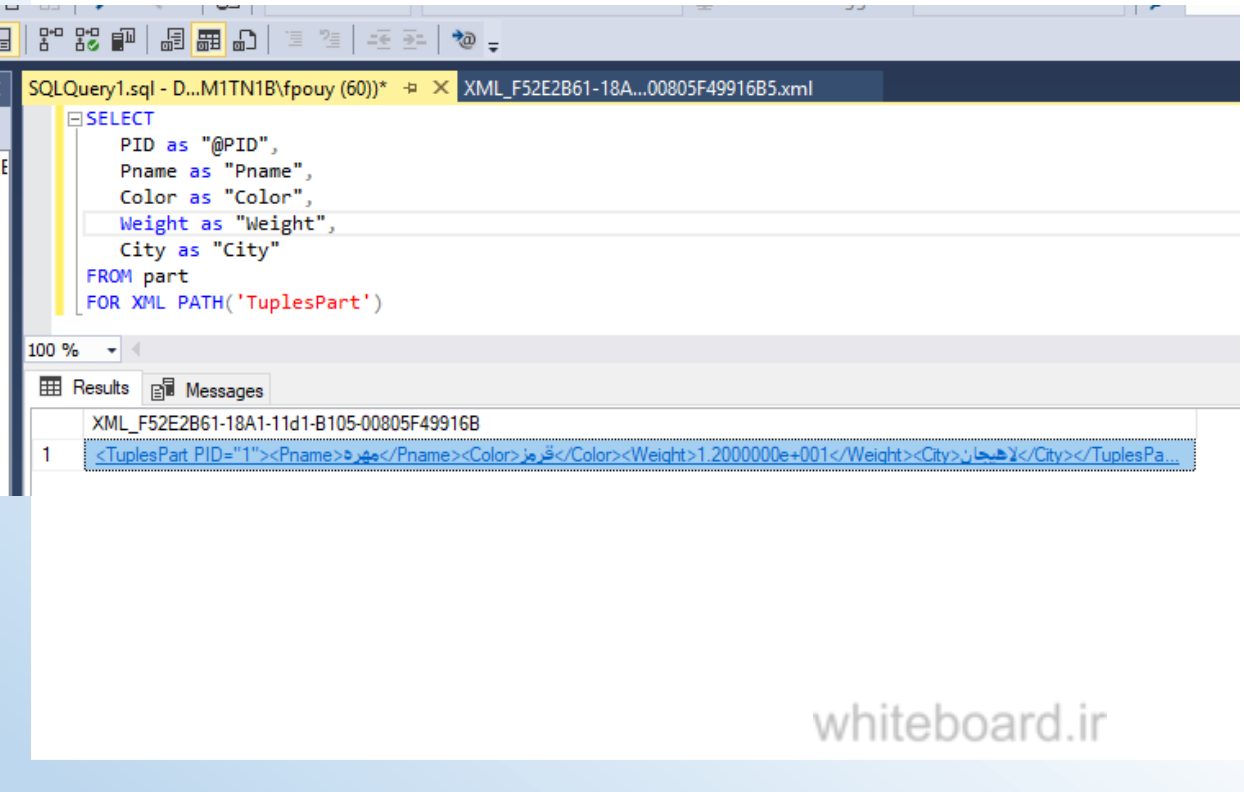
XML

- ۱۳ سال بعد از ارائه HTML ، روشی بنام XML ارائه شد که یک روش عمومی برای نشانه‌گذاری هر متنی بود. XML می‌خواست جایگزین HTML هم بشود اما این اتفاق هرگز به طور کامل روی نداد و هنوز هم فایل‌های HTML که XML نیستند در اینترنت فراوان استفاده می‌شوند.
- اما XML کاربردهای فراوان دیگری یافت. یکی از این کاربردها انتقال داده‌های موجود در پایگاه داده است: **زمانی که از DBMS (و احتمالا سیستم کامپیوتر) طرف مقابل اطلاعی نداریم ولی می‌خواهیم همه بتوانند دیتا را بخوانند.** (بدیهی است فایل‌ها و بک‌آپ‌های SQLServer فقط توسط یک SQLServer دیگر قابل خوانده شدن هستند).

XML

- فایل‌های اطلاعاتی XML (مانند سایر متن‌های markup شده) متن‌های ساده‌ای هستند. آنها می‌توانند اطلاعات جدول‌ها (رابطه‌ها) را در خود داشته باشند به گونه ای که **هم توسط انسان وهم توسط کامپیوتر قابل فهم** هستند. تمام متن معمولا در یک خط پشت سر هم میآید اما برخی نمایشگرها کد XML را بصورت خط به خط جدا شده نشان می‌دهند تا بوسیله انسان قابل فهم‌تر (خواناتر) باشد.
- کل سیستم وب یک پایگاه داده بسیار عظیم است. وب معنایی از این دیدگاه اطلاعات را پردازش می‌کند. وجود اطلاعات بصورت XML این کار را آسان‌تر می‌کند.
- هم کامپایلرها و هم DBMS ها ابزار ساخت فایل XML از روی جدول‌های داده را دارند. تبادل داده بین دو DBMS ناهمگون و همچنین بین اپلیکیشن و DBMS معمولا از این راه انجام می‌شود.

تبدیل جدول قطعات به XML



تمرین

- یک تریگر بنویسید که از قبول هر سفارش بیش از ۱۰،۰۰۰ تایی جلوگیری کند اما توزیع کننده‌ای که نامش کلانتری است استثناست (select کوئری مربوطه را می‌توانید با ویزارد بنویسید).
- آن را امتحان کنید (سعی کنید دو سفارش یکی مجاز و یکی غیر مجاز وارد کنید)

پایگاه داده‌ها

جلسه ششم

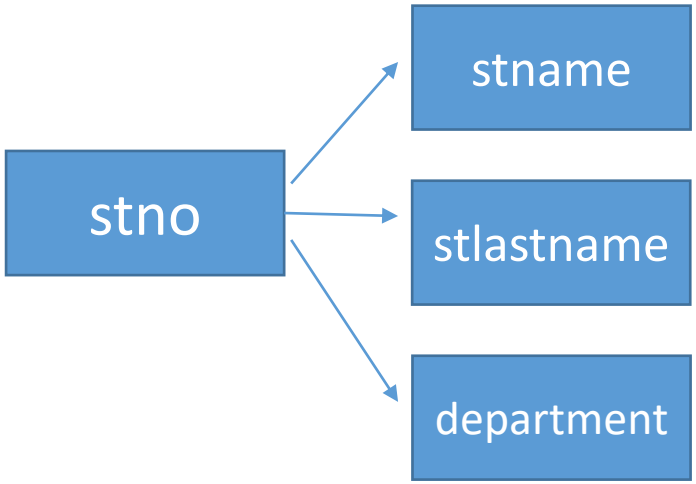
وابستگی و
تشخیص رابطه نرمال

تئوری وابستگی

- برای درک مباحث پیشرفته پایگاه داده از جمله نرمال‌سازی و فهم دقیق کلید، باید با تئوری وابستگی آشنا باشیم.
- در یک رابطه ممکن است یک ویژگی به یک ویژگی (یا مجموعه‌ای از چند ویژگی) دیگر وابستگی تابعی داشته باشد.
- یک مثال ساده: اگر مثلا در یک رابطه ویژگی رنگ اتومبیل به شماره پلاک ماشین وابستگی تابعی داشته باشد با دادن یک شماره پلاک خاص
(مثلا ۴۵ه۱۴۵) می‌توان رنگ را دانست (برعکس نه!).

مثلا...

stno	stname	stlastname	depatment
972134	حسن	محمدی	کامپیوتر
963432	پیمان	یساری	مکانیک



stno → stname

stno → stlastname

stno → department

سوال: آیا stlastname → stname نیز برقرار است؟

مفاهیم وابستگی

- stno → stname خوانده می‌شود
stname به stno وابستگی تابعی دارد یا
stno فِلش (می‌دهد) stname.
- در مثالِ قبل چون شماره دانشجویی کلید است بقیه به آن وابستگی دارند، اما گاهی ممکن است به غیر کلید هم وابستگی‌هایی دیده شود.
- وابستگی مفهومی **سمانتیک** است نه تجربی. یعنی از محتوای داده‌های فعلی جدول بدست نمی‌آید بلکه از درک معنای عنوان جدول بدست می‌آید. وابستگی روی عنوان تعریف می‌شود و در واقع مربوط به متغیر رابطه‌ای است نه مقدار لحظه‌ای رابطه.

مفاهیم وابستگی

- مثلا در جدول زیر

انتخاب واحد (شماره دانشجویی، نام دانشجو، کد رشته، نام رشته، معدل دیپلم)

شماره دانشجویی کلید است و طبعا همه ویژگی‌های دیگر به او وابسته‌اند.

- اما کد رشته با اینکه کلید نیست اما نام رشته به او وابسته است.
- برخی وابستگی‌ها را می‌توان به صورت معنایی پیدا کرد. برخی وابستگی‌های دیگر را هم می‌توان با محاسبه از روی وابستگی‌های شناخته شده بدست آورد.
- برخی وابستگی‌ها بی‌اهمیت هستند یعنی هیچ راهی برای زیر پا گذاشتن‌شان نیست. مثلا: کد رشته→(نام دانشجو، کد رشته)

قواعد آرمسترانگ

- به مجموعه **تمام وابستگی‌های ممکن** بستار می‌گویند و با F+ نشان داده می‌شود.
- برای بدست آوردن وابستگی‌های جدید از وابستگی‌های فعلی از سه اصل موضوعه آرمسترانگ استفاده می‌شود:

۱. بازتابی: اگر $B \subseteq A$ آنگاه $A \rightarrow B$

۲. بسط‌پذیری: اگر $A \rightarrow B$ آنگاه $(A,C) \rightarrow (B,C)$

۳. تراگذری: اگر $A \rightarrow B$ و $B \rightarrow C$ آنگاه $A \rightarrow C$

قواعد دیگر

- با استفاده از اصول موضوعه (= عقلانی ولی غیر قابل اثبات)، آرمسترانگ سپس تعدادی قاعده دیگر به دست آورد (که با استفاده از اصول موضوعه قابل اثبات هستند):

- اگر $A \rightarrow (B,C)$ آنگاه $A \rightarrow B$ و $A \rightarrow C$ (تجزیه)
(عکس قاعده تجزیه هم صادق است = قاعده اجتماع)

- مثال اگر (نام، محل تولد) → کد ملی آنگاه محل تولد → کدملی و .....
مثال اگر نام → (ش شناسنامه، محل صدور، تاریخ تولد) آنگاه هیچچی

- اگر $C \rightarrow D$ و $A \rightarrow B$ آنگاه $(A,C) \rightarrow (B,D)$ (ترکیب)

- اگر $(B,C) \rightarrow D$ و $A \rightarrow B$ آنگاه $(A,C) \rightarrow D$ (شبه‌تعدی)

- برای ما اصول موضوعه و قضایا فرقی ندارند و از همه بصورت قاعده استفاده می‌کنیم.

وابستگی‌های مینیمال

با اعمال قواعد آرمسترانگ بر روی جدول‌ها، مجموعه‌ای عظیم از وابستگی‌ها پدید می‌آید. برای فراهم کردن مجموعه‌ای کاهش‌ناپذیر از وابستگی باید شرایط زیر تامین شود:

۱. سمت راست هر FD (وابستگی تابعی) فقط یک ویژگی باشد.

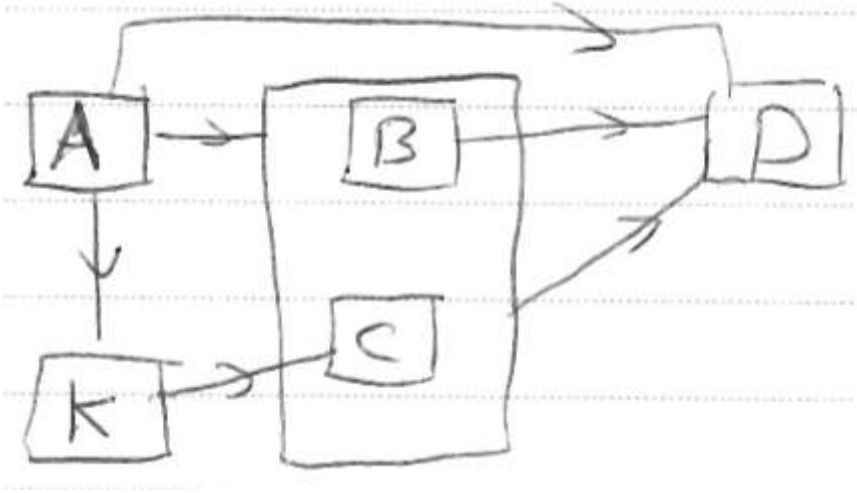
۲. سمت چپ FD مجموعه‌ای کاهش‌ناپذیر از ویژگی‌ها باشد.

۳. وابستگی‌هایی که بود و نبودشان فرق نمی‌کند (بی اهمیت‌ها یا آنها که از بقیه بدست می‌آید) حذف شوند.

تعریف کلید

- قبل از آشنایی با مفهوم ریاضی وابستگی، کلید را تعریف کردیم. اکنون می‌توان تعریفی ساده‌تر و دقیق‌تر ارائه کرد.
- هر چه که همه ویژگی‌های رابطه به آن وابسته باشند فراکلید است و اگر مینیمال باشد کلید (= کلید کاندید) است.
- **نکته مهم: در تمام مباحث مربوط به وابستگی و نرمال‌سازی، هر جا صحبت از کلید است منظور کلید کاندید می‌باشد. کلید اصلی و خارجی در این مبحث هیچ کاربردی ندارند.**

نمودار وابستگی‌های زیر را تا حد امکان ساده کنید (مراحل: نوشتن فورمول از روی نمودار، ساده کردن فورمول، کشیدن دوباره نمودار). از روی نمودار کلید را پیدا کنید.



فورمول بدست آمده

$A \rightarrow D$

$A \rightarrow B,C$

$A \rightarrow K$

$K \rightarrow C$

$B \rightarrow D$

$B,C \rightarrow D$

فورمول ساده شده

$A \rightarrow B$

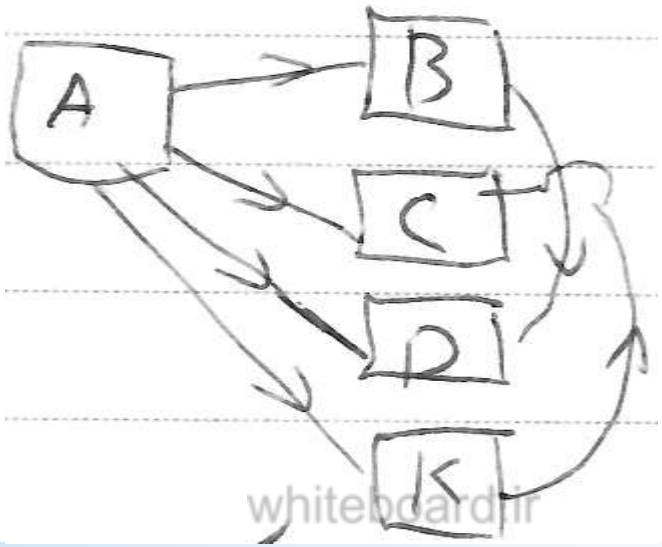
$A \rightarrow C$

$A \rightarrow D$

$A \rightarrow K$

$K \rightarrow C$

$B \rightarrow D$



افزونگی در ساختار جدول‌ها

- در طراحی زیر به جای نگهداری مشخصات قطعات در یک جدول جداگانه، آن را به داخل جدول سفارش‌ها آورده‌ایم.
- همانطور که واضح است، مشخصات قطعه شماره 1 چند بار تکرار شده است (در واقع همه قطعات). اینجا افزونگی کاملا مشخص است. در هر به روزرسانی و درج ممکن است اطلاعات متناقض شوند و نیز قطعاتی که سفارش‌شان حذف شود، مشخصات‌شان هم از دست می‌رود.

شماره قطعه	شماره توزیع‌کننده	نام قطعه	رنگ قطعه	تعداد
1	5	پیچ	زرد	200
1	7	پیچ	زرد	300
3	1	مهره	سفید	500

قواعد تشخیص رابطه نرمال

- قواعد نرمال نمی‌گویند که پایگاه چطور باید طراحی شود. آنها می‌گویند که رابطه‌های «متعارف» چگونه‌اند و نیز می‌گویند اگر جدولی به صورت «نامتعارف» طراحی شده چگونه می‌توان آن را نرمال‌سازی کرد تا از مشکلات مربوطه (مثل افزونگی، آنومالی و مشکلات به‌روزرسانی داده‌ها) خلاص شویم.
- کاد در مقاله خود فرم‌های اول و دوم و سوم نرمال را معرفی کرد (با این حال نرمال سازی جزئی از مدل رابطه‌ای نیست).
- کاد ابتدا فرض کرد که هر رابطه فقط یک کلید دارد.
- فرم‌های اول و دوم برای خود ارزشی ندارند و هدف از بیان آنها رسیدن به فرم نهایی یعنی سوم است.

فرم اول نرمال 1NF

- رابطه‌ای در فرم اول نرمال است که ویژگی‌های آن اتمیک (تجزیه‌ناپذیر) باشند.
- به بیان دیگر باید در هر خانه از جدول فقط یک مقدار ذخیره شده باشد نه بیشتر.
- تجزیه‌پذیری مفهومی عملی است. مثلا اگر در یک ستون نام و نام‌خانوادگی را داشته باشیم اما هرگز اقدامی برای جدا کردن آنها انجام ندهیم این مقدار اتمیک است ولی اگر در مواردی نام را از نام خانوادگی جدا کنیم آنگاه این مقدار اتمیک نیست.
- طبق تعریف کاد جدولی که 1NF نباشد اصلا رابطه نیست.

فرم دوم نرمال 2NF

- رابطه‌ای در فرم دوم نرمال است که در فرم اول نرمال باشد و هیچ کدام از ویژگی‌های غیر کلید به کلید وابستگی ناقص نداشته باشد (یعنی حتما وابستگی کامل، به تمام اجزای کلید با هم، وجود داشته باشد).
- می‌دانیم که کلید باید کاهش‌ناپذیر باشد، اما این مانع از آن نیست که برخی (نه همه) ویژگی‌ها به آن وابستگی ناقص داشته باشند.
- جدول سفارش‌ها با افزودن رنگ قطعه، دیگر نرمال نیست چون کلید <شماره قطعه،شماره توزیع‌کننده> است و رنگ فقط به بخشی از آن یعنی شماره قطعه وابسته است.

شماره توزیع‌کننده	شماره قطعه	تعداد	رنگ قطعه
1	2	300	سبز
5	2	200	سبز

فرم سوم نرمال3NF

- رابطه‌ای در فرم سوم نرمال است که در فرم دوم نرمال باشد و ویژگی‌های غیر کلید هیچ وابستگی به همدیگر نداشته باشند.
- با افزودن نام انگلیسی قطعه رابطه قطعات از 3NF می‌افتد چرا که نام فارسی و انگلیسی قطعه به همدیگر وابسته‌اند، بدون آنکه هیچکدام کلید باشند.

شماره قطعه	نام قطعه	نام قطعه به انگلیسی	رنگ قطعه
1	پیچ	screw	سبز
2	پیچ	screw	زرد

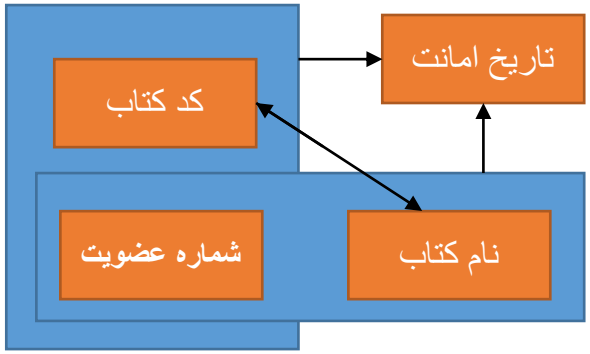
فرم نرمال بویس-کاد BCNF

- گفتیم وابستگی بی‌اهمیت، وابستگی‌ای است که هیچ راهی برای زیر پا گذاشتنش وجود ندارد. مثلا

{کد کتاب} → {کد کتاب ، شماره عضویت}

- رابطه ای BCNF است که در فرم اول نرمال باشد و برای تمام وابستگی‌های بااهمیت موجود در آن که به شکل A → B هستند، A فراکلید رابطه باشد.

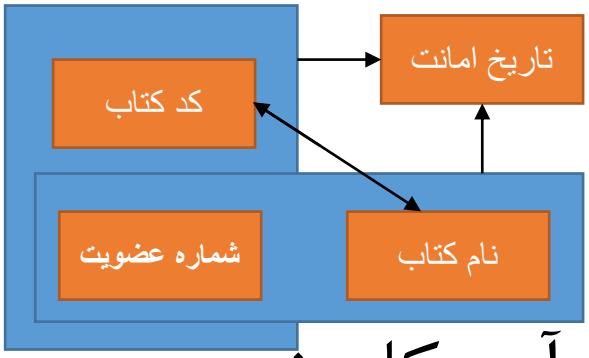
(یعنی روی شکل هر فلش فقط از فراکلید خارج شده باشد)



- اگر رابطه نرمال نباشد چه باید کرد؟

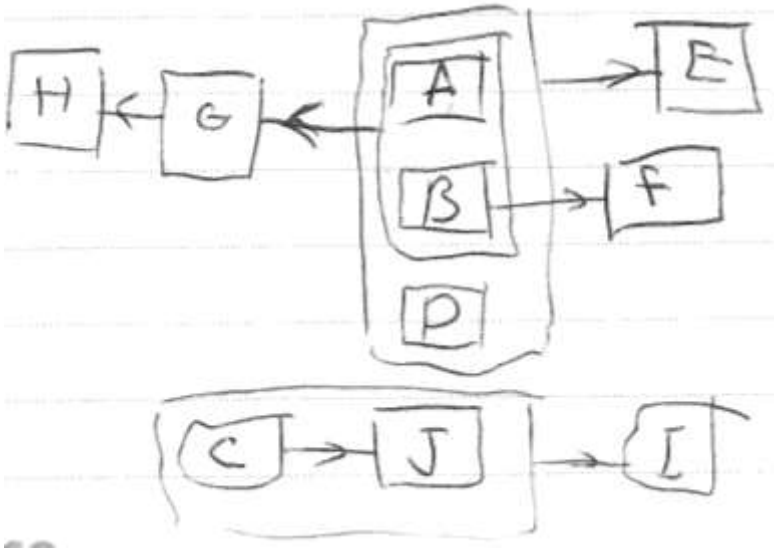
فرم نرمال بویس-کاد BCNF

- در مقاله اول کاد (که سه فرم نرمال در آن ارائه شد)، فرض بر آن بود که رابطه فقط یک کلید دارد (یا اگر چند کلید دارد این کلیدها مجزا هستند).
- زمانی که چند کلید مرکب در رابطه وجود دارد و این کلیدها دارای ویژگی مشترک هستند، فرم سوم نرمال دیگر نمی‌تواند از افزونگی جلوگیری کند.
- رابطه مقابل به وضوح افزونگی دارد (کد و نام کتاب موجب افزونگی است) اما در فرم سوم تشخیص داده نمی‌شود. کادرهای آبی کلیدند.



تمرین

- وابستگی‌ها را پیدا و تا حد مینیمال ساده کنید و کلید (ها) را پیدا کنید.



- جدول زیر تا چه سطحی نرمال است؟ (نام شهرها در سطح کشور یکتا هستند. نام خیابان‌ها یکتا نیستند)

کدپستی	استان	شهر	خیابان
2156874451	البرز	کرج	طالقانی

پیاده‌سازی سیستم‌های پایگاه داده

جلسه هفتم

طراحی پایگاه داده:
نرمال‌سازی و تعامد

طراحی پایگاه‌داده

- روش طراحی پایگاه‌داده در سطح منطقی بدین معناست برای پایگاه چه رابطه‌ها (جدول‌ها)یی بسازیم و هر کدام از این رابطه‌ها چه ویژگی‌ها (ستون‌ها)یی داشته باشند.
- قواعد طراحی بر روی مرابطه‌ها (متغیرهای رابطه‌ای) اعمال می‌شوند نه رابطه‌ها (مقدار لحظه‌ای جدول‌ها) اما در این بحث برای سادگی از کلمه رابطه استفاده می‌کنیم.
- طراحی پایگاه بیشتر یک هنر و یک فن است نه یک علم. یعنی روشی سیستماتیک برای طراحی وجود ندارد و نمودار ER هم فقط تا حدودی می‌تواند به طراح کمک نماید، نه آنکه بتوان پایگاه را دقیقا بر اساس آن ساخت. طراحی پایگاه نیاز به هوش و تجربه دارد.

نرمال‌سازی و تجزیه

- وظیفه طراح است که رابطه‌های پایه را به گونه‌ای طراحی کند که حداقل در سطح BCNF باشد (مگر در موارد استثنایی که مورد آن گفته خواهد شد).
- در مواردی که یک رابطه نرمال نیست، باید آن را به دو (یا چند) رابطه تجزیه کرد به گونه‌ای که از پیوند آنها دقیقا همان رابطه اولیه به دست آید.
- اگر از پیوند رابطه‌های حاصل از تجزیه همان رابطه بدست نیاید (مثلا تعدادی تاپل زائد ایجاد شود) تجزیه نامطلوب است.

تجزیه رابطه

- برای تجزیه به دو رابطه باید ویژگی‌های عنوان را سه دسته کرد. گروهی برای رابطه اول، گروهی برای رابطه دوم و گروهی که بین هر دو رابطه مشترک است. به این صورت با دو پرتو می‌توان رابطه را تجزیه کرد (و با پیوندکردن آنها دوباره رابطه اول را بدست آورد).
- **قاعده ریسانن** می‌گوید: تجزیه نباید باعث از بین رفتن هیچکدام از وَابستگی‌های تابعی شود یعنی مجموع وابستگی‌های تابعی موجود در دو رابطه حاصل از تجزیه باید دقیقا همان وابستگی‌های موجود در رابطه اولیه باشد.
- طبق **قضیه هیث** اگر ویژگی‌های رابطه را به سه گروه A , B , C تقسیم کنیم، و A مشترک باشد، تجزیه‌ای مطلوب و بی‌کم‌وکاست است که $A \rightarrow B$ (و احتمالا $A \not\rightarrow C$)

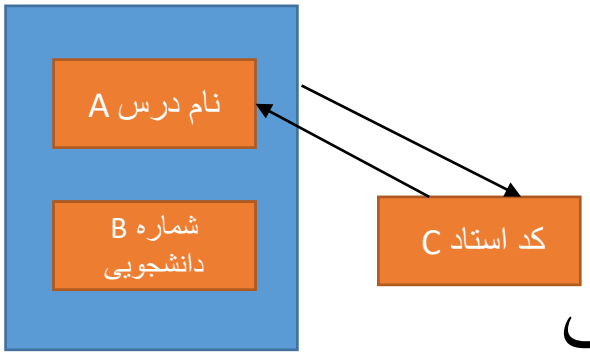
لزوم رعایت قواعد تجزیه

- عدم رعایت قاعده ریسانن موجب می‌شود که نتوانیم تاپل‌های یک رابطه را به صورت مستقل درج و به روزرسانی کنیم.
- اما عدم رعایت قضیه هیث موجب مشکل بزرگتری می‌شود که ایجاد تاپل‌های زائد هنگام پیوند دوباره است.

شماره قطعهA	شماره توزیع‌کننده B	نام قطعه C	رنگ قطعه D	تعداد E
1	5	پیچ	زرد	200
1	7	پیچ	زرد	300
3	1	مهره	سفید	500

رابطه‌های غیر قابل تجزیه

- مواردی وجود دارد که تجزیه باعث از دست رفتن وابستگی‌ها می‌شود، که در این صورت باید افزونگی را تحمل کنیم و نمی‌توان به سطح نرمال رسید.
- شکل: کادرهای آبی کلیدند، همانطور که معلوم است در این دانشگاه هر استاد فقط یک درس ارائه می‌دهد و دانشجو فقط یکبار یک درس را می‌گیرد.
- رابطه مقابل BCNF نیست و قابل BCNF شدن هم نیست. اگر رابطه را به دو رابطه (کد استاد، نام درس) و (شماره دانشجویی، کد استاد) تجزیه کنیم، وابستگی فلش بالایی از بین می‌رود.



سطوح بالاتر نرمال

- سطوح نرمال بالاتر یعنی 4NF,5NF (وابستگی پیوندی) مربوط به شرایط خاص هستند و بسیار بعید است در طراحی پایگاه‌های عادی پیش بیاید که رابطه‌ای BCNF باشد ولی 5NF نباشد.
- اگر رابطه‌ای کلید مرکب (کلیدی با بیش از یک ویژگی) نداشته باشد، 3NF برای آن سطح نهایی است و با بودن در این سطح 5NF هم هست.
- اهمیت وابستگی پیوندی بیشتر در چالش‌ها و مسائل تئوری و پژوهشی مربوط به پایگاه داده است.
- 5NF بالاترین سطح برای کاهش افرونگی است و تجزیه بیشتر از 5NF کمکی به کاهش افزونگی نمی‌کند.

سطح ششم نرمال 6NF

- سطح 6NF بالاترین سطح نرمال است به نحوی که رابطه‌ها در این سطح قابل تجزیه بیشتر نیستند.
- در این سطح در هر رابطه به غیر از کلید، فقط یک ویژگی دیگر وجود دارد.
- در این فرم نرمال «تهی» به طور کلی حذف می‌شود و پایگاه از مشکلات آن از جمله منطق سه سطحی رهایی می‌یابد. این فرم همچنین در برخی دیگر از موارد (از جمله پایگاه‌های داده زمانمند) کاربرد دارد.
- در مثال 6NF زیر، رتبه یک توزیع کننده نامعلوم است و این بدون تهی در پایگاه نشان داده شده است.

SN

SNO	SNAME
S1	Smith
S2	Jones

SS

SNO	STATUS
S1	20

SC

SNO	CITY
S1	London
S2	Paris

مزایای نرمال‌سازی

- رابطه نرمال یک نمایش خوب از دنیای واقعی خواهد بود (درک طراحی آسان‌تر می‌شود).
- نرمال سازی موجب کاهش افزونگی می‌شود. البته تضمین نمی‌کند که حتما افزونگی را از بین ببرد، چون اولا گاهی برای حفاظت از وابستگی‌ها نمی‌توانیم به سطح نرمال کافی برسیم ثانیا برخی افزونگی‌ها اصلا ربطی به نرمال‌سازی ندارند.
- کاهش افزونگی تا حدودی موجب جلوگیری از آنومالی (= مشکلات به روزرسانی داده) می‌شود. همچنین ساده‌سازی کوئری‌ها و انجام‌پذیری الزام به رعایت برخی قیدهای جامعیت از نتایج نرمال‌سازی است.

کم‌کردن درجه نرمال (دینرمال سازی)

- با وجود تمام محاسن و ضرورت‌هایی که برای نرمال بودن رابطه برشمردیم، گاهی هیچ چاره‌ای نداریم جز اینکه رابطه‌ای که به صورت نرمال طراحی کرده‌ایم را به سطوح نرمال پایین‌تر برگردانیم و در واقع درجه نرمال را کاهش دهیم.
- خواندن اطلاعاتی که در سطوح بالای نرمال هستند نیاز به عمل پیوند دو یا چند رابطه دارد. پیش می‌آید که حجم فوق‌العاده زیاد داده‌های دو رابطه در کنار تعداد فراوان کاربرانی که همزمان نیاز به خواندن داده‌های پایگاه دارند کارایی پایگاه را به شدت پایین می‌آورد به گونه‌ای که زمان پاسخگویی به درخواست‌ها به نحو غیر قابل قبولی زیاد می‌شود.

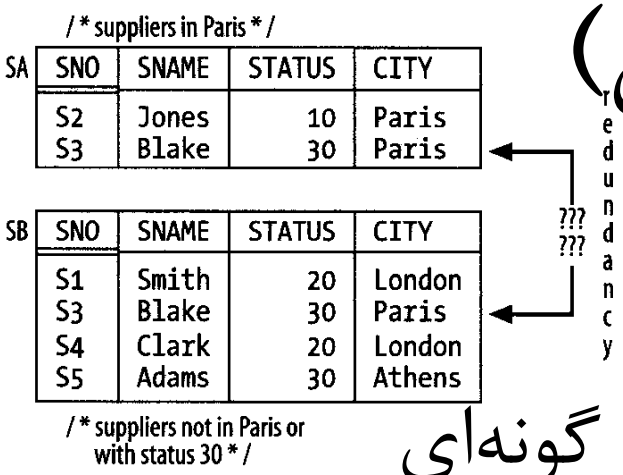
دینرمال سازی

- **وقتی تمام تکنیک‌های دیگر** از جمله تیونینگ پایگاه با **شکست** مواجه شد به عنوان آخرین راه حل می‌توان به ذخیره جدول‌های پایه در سطح پایین نرمال فکر کرد.
- در چنین شرایطی با ایجاد قیود جامعیت خاص می‌توان تا حدودی از ناسازگاری داده‌ها (آنومالی) جلوگیری کرد. اما این قیود بسیار پرهزینه هستند و سرعت به روزرسانی داده‌ها بسیار کندتر خواهد شد. اما زمانی که تعداد به روزرسانی‌ها نسبت به تعداد خواندن‌ها خیلی کمتر باشد این هزینه قابل تحمل خواهد بود. همچنین نوشتن کوئری‌ها در شرایط غیر نرمال سخت است و در صورت بی‌دقتی احتمال نوشتن کوئری بصورت نادرست زیاد خواهد بود (خصوصا توابع جمعی مثل شمارش و معدل).

ضعف پایگاه‌های رابطه‌ای امروزی

- مدل رابطه‌ای اساسا ارائه شده تا سطوح ذخیره فیزیکی و طراحی منطقی از هم مجرد باشند، اما چون این تجرید تا امروز کامل انجام نشده و ساختار فیزیکی (فایلی) انعکاسی از طراحی منطقی (جدولی) هستند، نحوه طراحی منطقی بر کارایی اثر می‌گذارد و دقیقا به همین دلیل ما گاهی مجبور به پایین آوردن سطح نرمال هستیم.
- اگر روزی DBMS های پیشرفته به اندازه کافی هوشمند شوند و بتوانند این مشکل را حل کنند، ما همیشه طراحی را بدون فکر کردن به کارایی در سطح BCNF یا حتی 6NF انجام می‌دهیم و مطمئن خواهیم بود که سیستم خودش بهترین کارایی ممکنه را به ما خواهد داد.

طراحی متعامد (تفکیکی)



- در تصویر مقابل می‌بینید که بجای یک مرابطه، دو مرابطه به توزیع‌کنندگان اختصاص یافته به گونه‌ای که توزیع‌کنندگان پاریسی در یک جدول و توزیع‌کنندگان غیرپاریسی یا آنها که رتبه‌شان ۳۰ است در جدول دیگر ذخیره می‌شود.
- چنین طراحی‌ای به طور واضح خوب نیست و علاوه بر آن موجب افزونگی هم می‌شود ولی چنین طرحی کاملا نرمال‌است (مشابه چنین طراحی‌هایی در پایگاه‌های توزیع‌شده دیده می‌شود).

طراحی متعامد (تفکیکی)

- اصل طراحی متعامد (orthogonal) : قیدها نبایستی اجازه دهند که یک تاپل در دو مرابطه یک پایگاه داده ظاهر شود.
- کلمه تفکیکی (متعامد) در اینجا به این واقعیت اشاره دارد که طبق این اصل مرابطه‌ها بایستی مجزا از هم باشند -که در مثال اسلاید قبل نیستند- حال اگر قیدهایی برای «منع همپوشانی» داشته باشند چنین اتفاقی نمی‌افتد.
- در بسیاری موارد عدم تفکیک آشکار است، یعنی اگر دو رابطه سازگار (هم‌عنوان) نباشند نمی‌توانند این اصل را نقض کنند. اما گاهی این امر نه در کل رابطه بلکه در بخشی از آن مستتر است و با تجزیه آشکار می‌شود.

تمرین

- یک سازمان خیریه در دانشگاه کمک‌های نقدی دانشجویان را جمع‌آوری و در جدول زیر ثبت می‌نماید. یک دانشجو در هر روز فقط یک‌بار می‌تواند کمک کند. برای تشویق به کمک، یک دستگاه ترازوی دیجیتال متصل به سیستم در محل نصب شده و هر دانشجویی که به خیریه کمک نماید می‌تواند وزن خودش را به رایگان اندازه بگیرد.
- ۱. تمام وابستگی‌هایی در این جدول هست را بنویسید و کلید (ها) را پیدا کنید.
- ۲. آیا جدول تا سطح BCNF نرمال است؟
- ۳. اگر نیست آن را به صورت رابطه‌های نرمال در این سطح در آورید و نشان دهید که این کار آسیبی به داده‌ها و وابستگی‌ها نمی‌رساند.

A شماره دانشجویی	B تاریخ پرداخت	C نام دانشجو	D مبلغ پرداختی	E وزن دانشجو

تمرین امتیازی (اختیاری)

- پایگاهی برای این منظور طراحی کنید: موجودیت‌ها عبارت‌اند از بازاریاب‌ها، مناطق فروش و محصولات. هر بازاریاب می‌تواند در یک یا چند منطقه فعالیت داشته باشد و هر منطقه یک یا چند بازاریاب دارد، هر بازاریاب یک یا چند محصول برای فروش دارد و هر محصول توسط یک یا چند بازاریاب فروخته می‌شود. هر کالا حتما در هر منطقه فروخته می‌شود ولی دو بازاریاب یک کالا را در یک منطقه نمی‌فروشند. هر بازاریاب در تمامی مناطق زیرپوشش خود کالاهای یکسانی را می‌فروشد.
(پیاده‌سازی لازم نیست. رسم جدول‌ها روی کاغذ و تعیین کلیدها کفایت می‌کند.)

پیاده‌سازی سیستم‌های پایگاه داده

جلسه هشتم

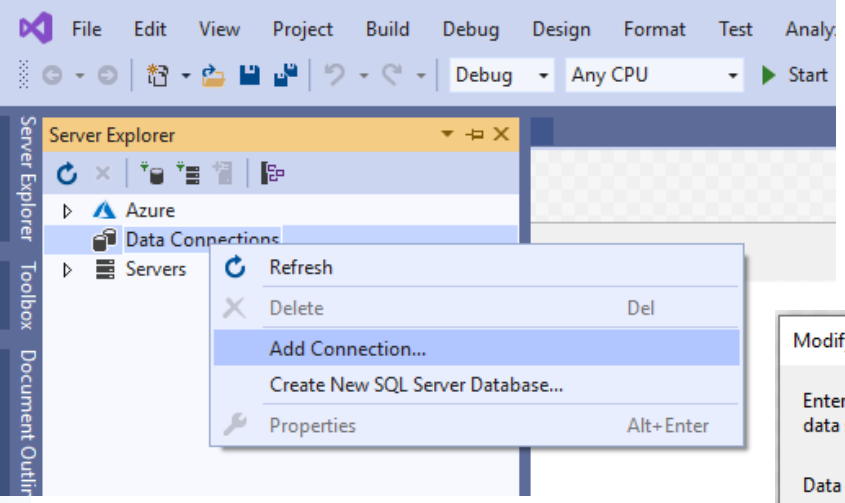
ارتباط با اپلیکیشن‌ها
افزایش کارایی

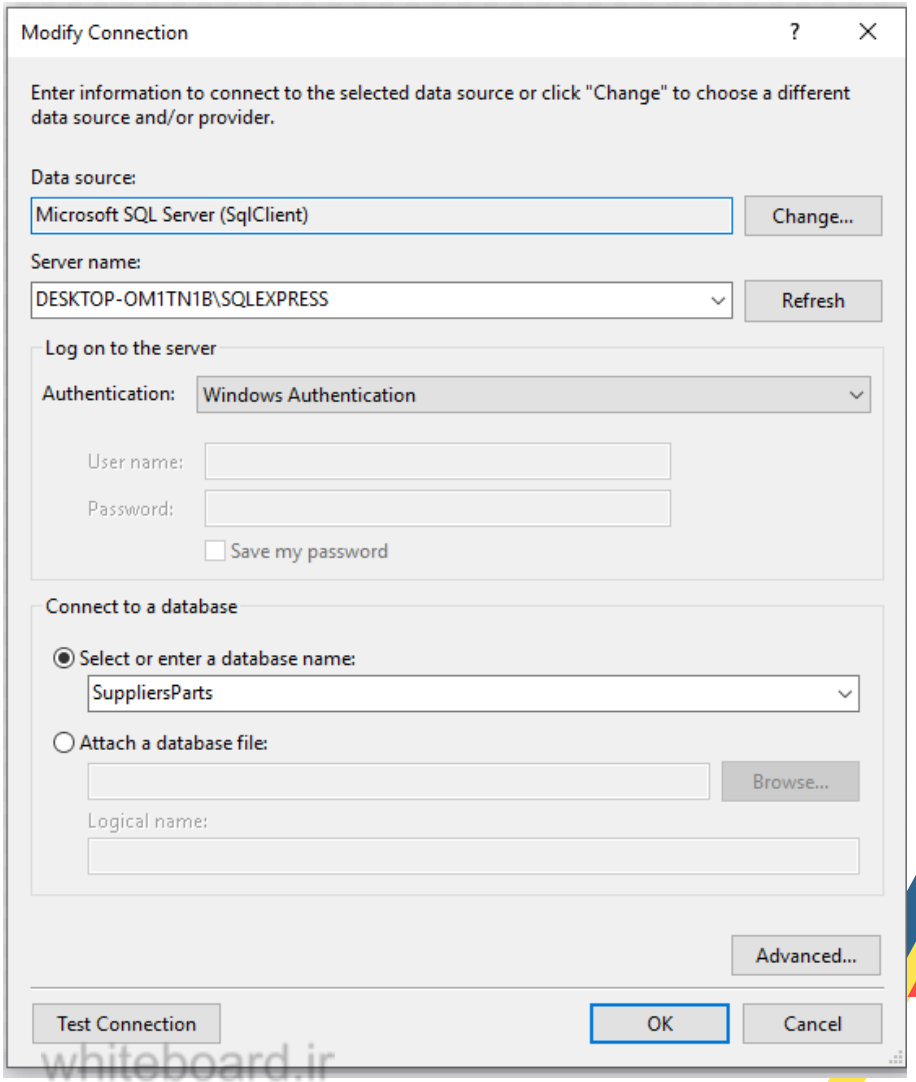
ارتباط با اپلیکیشن‌های ویندوزی

- دستورات SQL که از طریق قسمت‌های مختلف Management Studio وارد می‌شوند برای متخصصان کامپیوتر که کار طراحی و یا پشتیبانی از پایگاه داده را عهده‌دار هستند، مناسب‌اند.
- کاربران ساده میتوانند با اپلیکیشن‌ها کار کنند و طبیعتا از همین طریق با پایگاه داده هم ارتباط برقرار می‌کنند.
- در محیط ویندوز، تمام کامپایلرهای امروزی می‌توانند برنامه‌هایی تولید کنند که با تمام DBMS های امروزی ارتباط برقرار کنند (از طریق ADO).
- اما SQLServer و Visual Studio که هر دو محصول ماکروسافت‌اند، راه مستقیم و راحتی برای ارتباط دارند.
- دستور نصب VS2019 در: https://www.aparat.com/v/TdWFl

ایجاد کانکشن

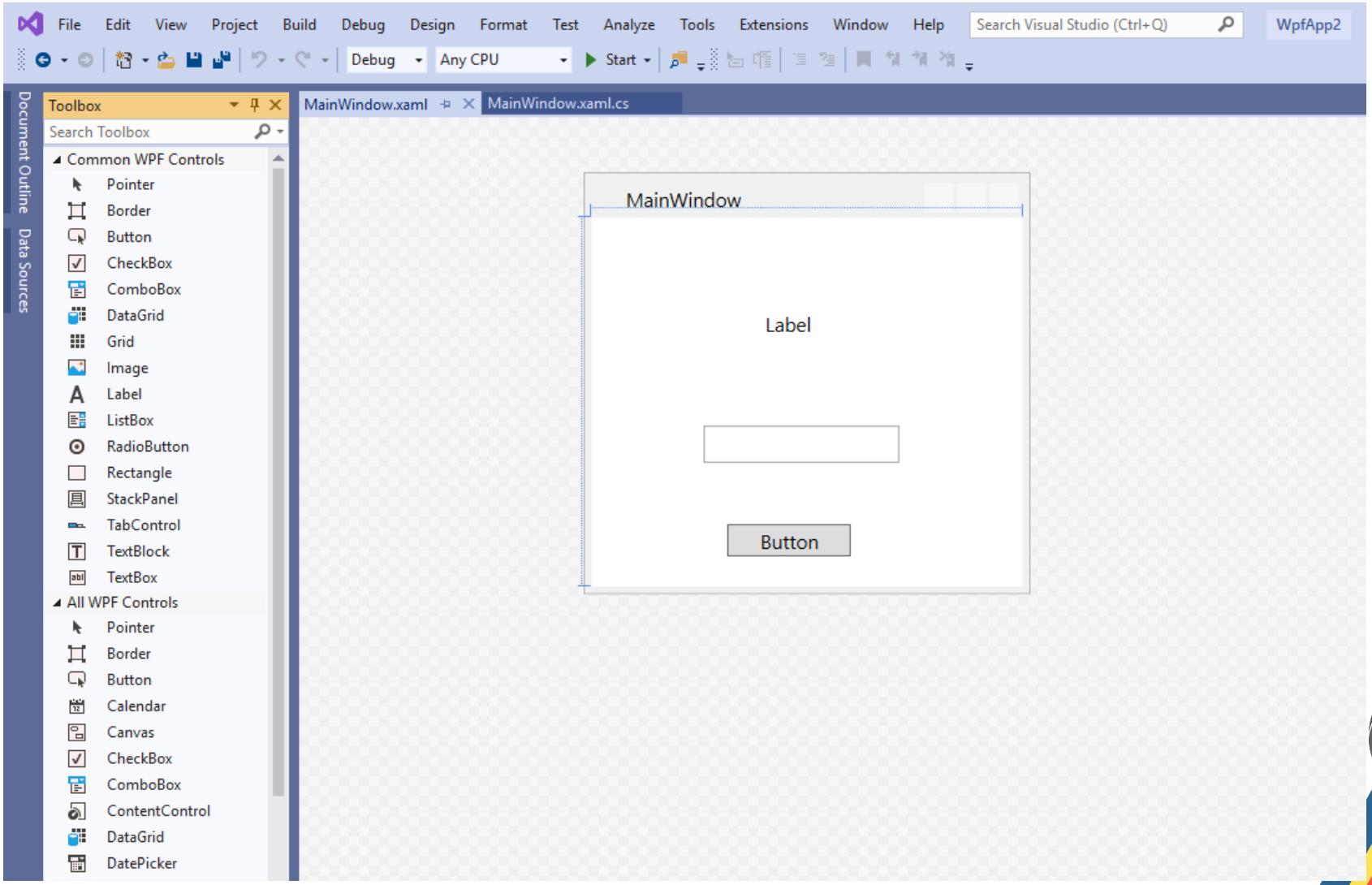
- در شروع کار باید یک کانکشن (ارتباط) از برنامه کاربردی بر روی پایگاه داده (که نام پایگاه و نام سرور آن معلوم است) باز کرد و پس از پایان کار هم باید کانکشن را بست.
- مطابق تصویر اسلاید بعد یک کانکشن اضافه کنید. نام سرور را وارد کنید (خیلی وقت‌ها سیستم نمی‌تواند لیست سرورها را بیاورد و باید آن را تایپ کنیم. اگر نام سرور را نمی‌دانیم از کنار فلش سبز management studio می‌توان آن را دید و کپی کرد) و سپس نام دیتابیس را انتخاب کنید.





یک پروژه ساده

- می‌خواهیم یک فرم ویندوزی بسازیم (در یک پروژه C#) که کاربر در آن یک شماره قطعه وارد کند، سپس باید نام قطعه به او نشان داده شود.
- در فرم‌های ویندوز ابتدا ظاهر فرم طراحی می‌شود و سپس برای آن کد نوشته می‌شود. یک تکست‌باکس می‌خواهیم برای دریافت عدد، یک لیبل می‌خواهیم برای درج نام قطعه و یک باتن که فرمان اجرای برنامه را می‌دهد. با دبل کلیک کردن روی باتن کد برنامه را می‌نویسیم.



یک پروژه ساده

- using System.Data.SqlClient را به سرآیندها اضافه کنید.
- یک SqlConnection جدید تعریف کنید، منبع آن را می‌توانید از استرینگ کانکشنی که قبلا تعریف کرده‌اید، کپی، پیست کنید. کانکشن را باز کنید.
- یک SqlCommand تعریف کنید که دستور پیدا کردن نام قطعه را در خورد داشته باشد.
- کامند را بصورتی که یک مقدار اسکالر (تکی) برگرداند اجرا کنید، مقدار برگشتی را داخل لیبل بریزید.
- کانکشن را ببندید.

ug Test Analyze Tools Extensions Window Help Search Visual Studio (Ctrl+Q) WpfApp2

bug Any CPU Start

xaml MainWindow.xaml.cs*

WpfApp2.MainWindow textBox

```
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Data.SqlClient;

namespace WpfApp2
{
    2 references
    public partial class MainWindow : Window
    {
        0 references
        public MainWindow()
        {
            InitializeComponent();
        }

        1 reference
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-OM1TN1B\SQLEXPRESS;Initial Catalog=SuppliersParts;Integrated Security=True");
            con.Open();
            SqlCommand findpart = new SqlCommand("select PName from Part where PID =" + textBox.Text, con);
            label.Content = findpart.ExecuteScalar();
            con.Close();
        }
    }
}
```

یک پروژه ساده

- حال می‌توان پروژه را اجرا کرد. فرم ظاهر می‌شود و با وارد کردن عدد، نام قطعه نشان داده می‌شود.

اهمیت کارایی

- امنیت و جامعیت (درستی) در پایگاه داده اهمیت زیادی دارند، اما موضوع دیگری که بسیار پراهمیت است کارایی است. هیچکس دوست ندارد که کوئری را روی داده‌ها اجرا کند بعد برود یک چای بخورد به امید اینکه وقتی برمی‌گردد جواب آماده شده باشد. هیچکس هم دوست ندارد بعد از اینکه سیستم تحویل مشتری شد از تلفن بشنود که این سیستم به درد ما نمی‌خورد چون خیلی کند است.
- افزایش کارایی با بهینه‌سازی و تنظیم (تیونینگ) سیستم به دست می‌آید.

شاخص‌گذاری

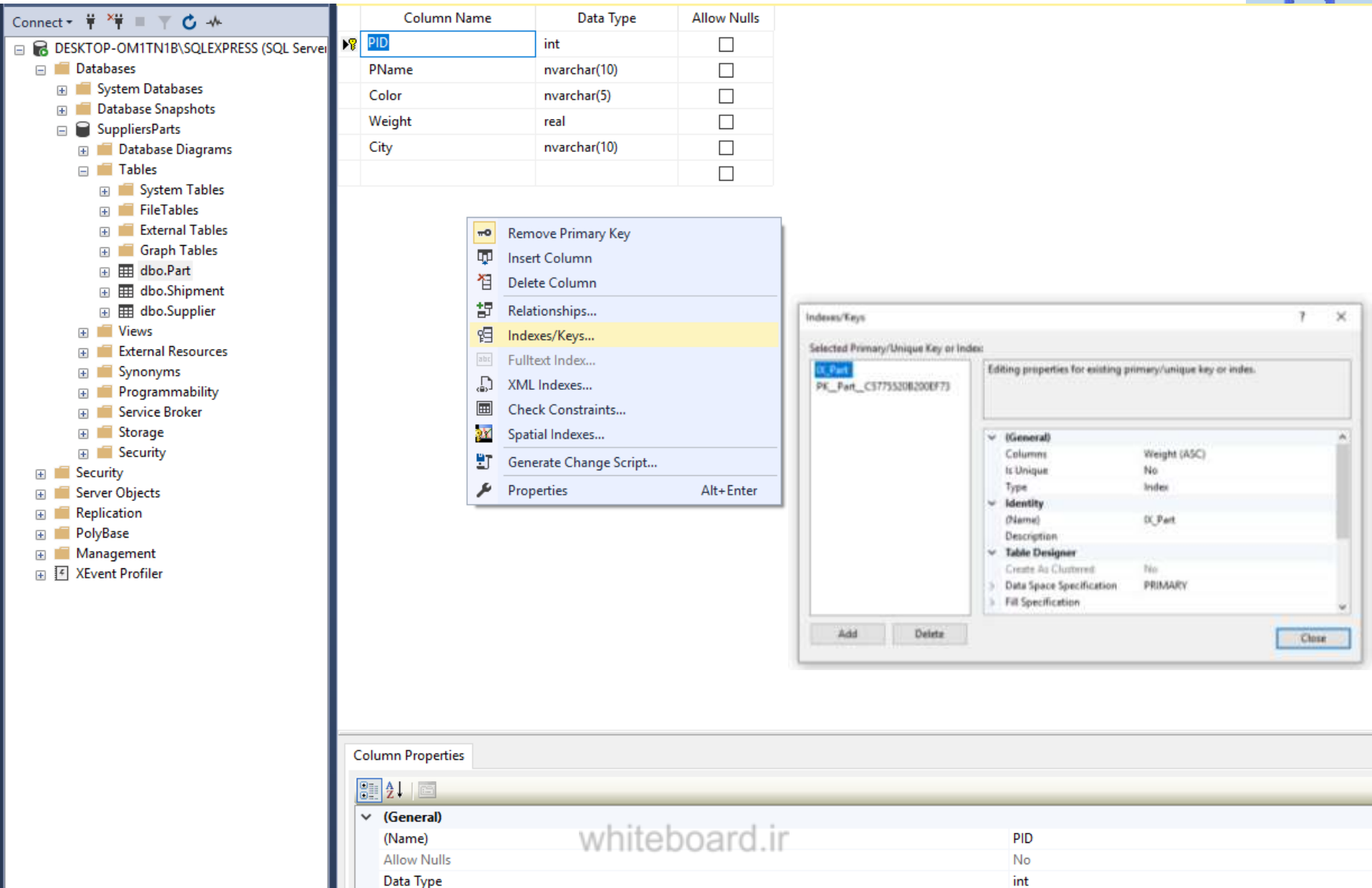
- جستجو در میان داده‌ها، به طور کلی، عملیات وقت‌گیری است. استفاده از شاخص index این عمل را تسریع می‌کند.
- شاخص یک ساختار داده‌ای اضافی در پایگاه داده است که به منظور تسریع در جستجو در پایگاه ساخته می‌شود. با استفاده از شاخص، بجای جستجوی سطر به سطر داده‌ها، مستقیما به جایی هدایت می‌شویم که داده ممکن است آنجا باشد.
- شاخص تقریبا مثل حروف کنار دفترچه تلفن عمل می‌کند.

عیب‌های شاخص‌گذاری

- این تکنیک بی‌هزینه نیست. به غیر از حافظه و امکانات سیستم که برای آن مصرف می‌شود، ضمن سریعتر کردن جستجوها، عملیات تغییر داده‌ها یعنی درج و حذف و به‌روزرسانی را کندتر می‌کند. چون باید شاخص‌ها نیز به روزرسانی شوند.
- با توجه به این موارد شاخص باید فقط روی داده‌هایی که واقعا آن را لازم دارند گذاشته شود. یعنی آنها که عملیات جستجو بر رویشان زیاد است و احتمالا تغییرات زیادی هم ندارند.

شاخص‌گذاری در SQLServer

- SQL Server بر روی ویژگی‌های کلید اصلی و خارجی به صورت خودکار شاخص می‌گذارد چون فرض می‌کند بیشتر عملیات جستجو را دارند.
- اگر طراح بخواهد می‌تواند بر روی فلیدهای دیگر جدول‌ها هم شاخص بگذارد. مثلا در اینجا نحوه شاخص گذاری روی ویژگی وزن در جدول قطعات را می‌بینید. هنگام شاخص‌گذاری می‌توان آن را یکتا تعریف نمود که این کار معمولا روی کلیدهای جانشین (کاندیدهایی که بعنوان اصلی انتخاب نشده‌اند) انجام می‌شود.


Connect
DESKTOP-OM1TN1B\SQLEXPRESS (SQL Server
Databases
System Databases
Database Snapshots
SuppliersParts
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.Part
dbo.Shipment
dbo.Supplier
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler
Column Name
Data Type
Allow Nulls
PID
int
PName
nvarchar(10)
Color
nvarchar(5)
Weight
real
City
nvarchar(10)
Remove Primary Key
Insert Column
Delete Column
Relationships...
Indexes/Keys...
Fulltext Index...
XML Indexes...
Check Constraints...
Spatial Indexes...
Generate Change Script...
Properties
Alt+Enter
Indexes/Keys
Selected Primary/Unique Key or Index:
IX_Part
PK__Part__C3775520B200EF73
Editing properties for existing primary/unique key or index.
(General)
Columns
Weight (ASC)
Is Unique
No
Type
Index
Identity
(Name)
IX_Part
Description
Table Designer
Create As Clustered
No
Data Space Specification
PRIMARY
Fill Specification
Add
Delete
Close
Column Properties
(General)
(Name)
PID
Allow Nulls
No
Data Type
int
whiteboard.ir

بهینه‌سازی کوئزی‌ها

- گفتیم در زبان‌های غیر رویه‌ای مثل SQL ما روش انجام کار را به سیستم نمی‌گوییم. مثلا نمی‌گوییم مرتب سازی را به کدام روش sort انجام بده.
- اما در مواردی نحوه نوشتن کوئری بر نحوه اجرای آن در سیستم تاثیر می‌گذارد.
- به عبارت دیگر نحوه نوشتن دستور SQL ممکن است بر زمان اجرای آن اثر منفی یا مثبت بگذارد.

یک مثال

- پیدا کردن نام کسانی که قطعه شماره دو را فروخته‌اند:

((shipment join supplier) where pid = 2) {sname}

- فرض کنید ۱۰۰ توزیع کننده و ۱۰،۰۰۰ سفارش هست که فقط ۵۰ تای آنها مربوطه به قطعه ۲ می‌باشد. همچنین فرض کنید بافری در حافظه وجود ندارد و همه خواندن‌ها یکی‌یکی از حافظه جانبی (هارد دیسک) انجام می‌شود.
- بنابراین برای ساخت پیوند و ذخیره آن روی دیسک برای هر سفارش باید یکبار همه توزیع‌کننده‌ها خوانده شود که می‌شود ۱،۰۱۰،۰۰۰ بار و نتایج پیوند که ۱۰،۰۰۰ عدد هستند هم باید نوشته شوند.
- در مرحله بعد باید بین ۱۰،۰۰۰ نتیجه حاصل از مرحله قبل جستجو شود تا ۵۰ تاپل جدا شوند.
- پرتو بر روی ۵۰ نتیجه صورت می‌گیرد.
- جمعا می‌شود ۱،۰۳۰،۰۵۰ عمل ورودی خروجی دیسک.

یک استراتژی دیگر

((shipment join supplier) where pid = 2) {sname}

- این بار ابتدا جستجو (شرط) را روی جدول سفارشات انجام می‌دهیم. می‌شود ۱۰،۰۰۰ عمل خواندن و ۵۰ نوشتن نتیجه در حافظه.
- نتیجه قبلی را با توزیع‌کنندگان پیوند می‌دهیم. می شود ۵۰ بار خواندن ۱۰۰ توزیع‌کننده که می‌شود ۵،۰۰۰ و بعد هم نوشتن ۵۰ نتیجه در حافظه.
- این بار فقط ۱۵،۱۰۰ عملیات حافظه‌ای داشتیم که اگر زیر ۲۰۰ رکورد را در حافظه اصلی (بافر) نگه داریم تازه از این هم خیلی کمتر می‌شود.

فرم‌های متفاوت نوشتن کوئری

- در مثال قبل فرض بر آن بود که سیستم دستور نوشته شده چند جور اجرا شود حال آنکه گاهی خود دستور را می‌شود چند جور نوشت.
- اسامی توزیع‌کننده‌هایی را بدهید که قطعه شماره ۲ را عرضه می‌کنند.

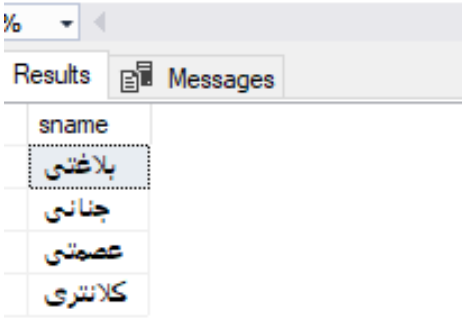
```
Select distinct sname
from supplier inner join shipment
on supplier.sid = shipment.sid
where pid = 2
```

Results | Messages

sname
بلاغتی
جنانی
عصمتی
کلانتری

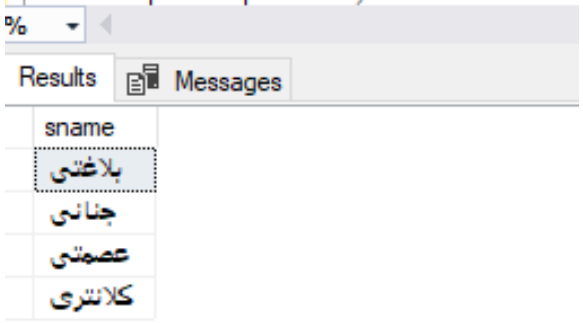
- جای پیوند با انتخاب فرعی هم می‌شود نوشت (خوب):

```
Select distinct sname
from supplier
where sid in
(
select sid
from shipment
where pid = 2
)
```



- و این یکی هم با استفاده از exists همین کار را می‌کند (بد).

```
Select distinct sname
from supplier
where exists
(select *
from shipment
where
shipment.sid = supplier.sid
and shipment.pid = 2)
```



بررسی کارایی هر دستور

- دیدم که در صورتی که هیچ حافظه‌ای برای بافر نداشته باشیم دستور به فرم اول اگر با استراتژی دوم اجرا شود حدود ۶۰،۰۰۰ عمل خواندن و نوشتن دارد.
- زمانی که کار را بر عهده سیستم بگذاریم شاید برایمان معلوم نباشد که کدام استراتژی را انتخاب می‌کند اما اگر دستور را به فرم دستور بالای صفحه قبل بنویسیم، استراتژی دوم انجام می‌شود (select دومی فقط یک بار اجرا می‌شود).
- دستور پایینی اما استراتژی اول را اجبار می‌کند که در اینجا کارایی خیلی کمتری دارد، اما اگر جدول توزیع‌کنندگان خیلی بزرگ و جدول سفارش‌ها کوچک بود قضیه فرق می‌کرد.

بهینه‌سازی کوئری‌ها در SQLServer

- SQLServer به صورت خودکار کوئری‌ها را بهینه‌سازی می‌کند. بنابراین در بسیاری موارد (نه همیشه) فرقی نمی‌کند که آن را چطور بنویسیم.
- یعنی حتی اگر دستور را به شکل دستور پایینی بنویسیم بعید نیست که سیستم آن را بهینه کند و به صورت دستور بالایی (کارایی بهتر) انجام دهد.
- اگر داده‌های واقعی در دسترس‌مان باشد ساده‌ترین کار آن است که کوئری را به چند صورت مختلف بنویسیم و زمان اجرا را ببینیم. اینگونه مطمئن می‌شویم که کدام یک کارایی بهتری دارند.

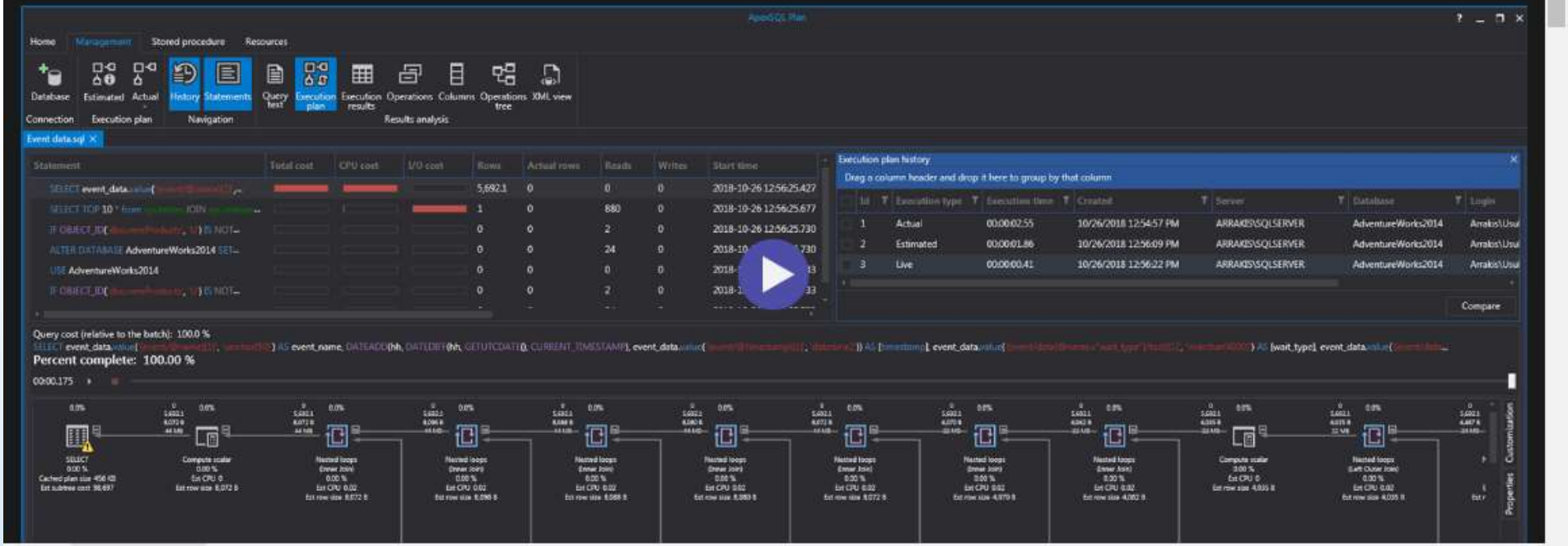
بهینه‌سازی کوئری‌ها در SQLServer

- خود SQL Server ابزاری دارد بنام Database Engine Tuning Advisor که البته در نگارش رایگان (express) در دسترس نیست (و فقط هم بهینه‌ساز نیست).
- یک ابزار خوب برای ارزیابی کوئری‌ها که هم رایگان است و هم راحت به SQLServer وصل می‌شود ApexSQL Plan است.

SQL execution plan viewing and analysis

Open, view, analyze and explore an execution plan in SQL Server, profile SQL code and more

Free product



- Analyze and optimize SQL query execution plans
- Understand query performance characteristics
- Customize graphical query execution plan view
- Analyze query live performance statistics
- Profile stored procedure performance
- Identify query performance issues and deadlocks
- Analyze query waits & review query execution plans
- See the actual relative cost of each plan operator
- Compare estimated and actual query execution plans
- Generate lazy profiling reports

برخی دیگر از اقدامات برای تیونینگ

- تیونینگ پایگاه داده فقط تعیین شاخص ها و بهینه‌سازی دستورها نیست.
- در بعضی موارد استثنایی دینرمال کردن رابطه‌ها (ذخیره‌سازی در سطح پایینتر نرمال) می‌تواند راه حل باشد (زمانی که چاره دیگری نداریم).
- ارتقا سخت‌افزار و نیز نرم‌افزار (نگارش خاصی از DBMS) یکی دیگر این اقدامات است. مثلا در جایی که کوئری‌ها دارای پردازش موازی‌اند افزایش هسته‌های پردازنده کارایی را به شدت افزایش می‌دهد.

تمرین

- الف: برنامه‌ای بنویسید که شماره توزیع‌کننده، نام توزیع‌کننده و موقعیت و شهر را از کاربر بگیرد و آن را در پایگاه ثبت کند (اگر موفق به نصب ویژوال استدیو نشدید فرم را دستی رسم کرده و برنامه را بدون استفاده از کامپیوتر دستی بنویسید).
- ب: به صورت مختصر شرح دهید که شاخص چه کاری انجام می‌دهد و در کجا SQLServer به صورت خودکار شاخص می‌سازد.
- ج: بهینه‌سازی کوئری را به زبان خودتان در سه تا پنج خط توضیح دهید.

پیاده‌سازی سیستم‌های پایگاه داده

جلسه نهم

تراکنش‌ها و همروندی

سوال

- وقتی محتوای یکی از ویژگی‌های تاپلی را در پایگاه تغییر می‌دهیم ولی هنوز از آن سطر خارج نشده‌ایم، کنار آن ! قرمزی رنگی ظاهر می‌شود که می‌گوید این ویژگی تغییر یافته. در همین حال اگر جدول را جای دیگری باز کنیم و بخواهیم همین قسمت را تغییر دهیم، اجازه آن به ما داده نخواهد شد. چرا؟

تراکنش‌ها

- یک تراکنش، یک واحد کاری است که سیستم مدیریت پایگاه داده‌ها باید آن را انجام دهد.
- تراکنش ممکن است یک کوئری باشد یا چند کوئری پشت سر هم.
- از آنجا که تراکنش یک واحد کاری است یا به تمامی اجرا می‌شود یا اگر بخشی از آن با شکست روبرو شد گویی کل کوئری‌های آن شکست خورده و پایگاه به وضعیت قبل از اجرای تراکنش بازگردانده می‌شود.

تراکنش‌ها

- مثلا یک دستور به روزرسانی که قرار است وزن تمام قطعات را دو برابر کند اگر در مورد یک تاپل (احتمالا به دلیل سرریز) شکست بخورد در مورد بقیه تاپل‌ها نیز تغییری صورت نخواهد داد.
- مثال دیگر، تراکنش با چند کوئری: عملیات انتقال پول از یک حساب بانکی به حساب. یک دستور پول را از حسابی کم می‌کند و دستور دیگر پول را به حساب دیگر می‌افزاید. اگر یکی از این دستورها ناموفق باشد (از جمله به دلیل نواقص سخت افزار مثل قطع برق یا خراب شدن هارد دیسک) نباید اثر اجرای دستور دیگر در داده‌های سیستم بماند.

ویژگی‌های ACID

تراکنش باید این ویژگی‌ها را داشته باشد:

- Atomic اتمی هستند. یک بسته دستوری غیر قابل تجزیه که موقع اجرا، همه یا هیچ است.
- Consistency پایگاه داده پس از تکمیل تراکنش باید در حالت سازگار باشد، یعنی همه قیود جامعیت رعایت شده باشد (فعلا به این بحث کاری نداریم که در میانه تراکنش لزومی دارد که پایگاه حتما سازگار باشد یا نه).
- Isolation تراکنش‌ها در انزوا و تنهایی کار خود را می‌کنند و در کار هم دخالت نمی‌کنند. نتیجه تغییرات تراکنش فقط پس از تکمیل توسط بقیه (از جمله تراکنش‌هایی که احتمالا به صورت همروند در حال اجرایند) قابل دیدن است.
- Durability نتایج کار تراکنش پس از اتمام، دوام دارد و در پایگاه ماندنی است حتی اگر بعد از آن سیستم خراب شود.

بررسی قیدها

- می‌دانیم قیدهای جامعیت(چه عمومی و چه اختصاصی) باید بعد از به روزرسانی‌ها بررسی شوند و اگر قیود زیرپا گذاشته شده باشند، به روزرسانی واگردانی شود.
- برخی قیدها در سطح یک تاپل هستند (قید تاپل) مانند چک کردن دامنه و نوع. برخی در سطح یک مرابطه‌اند (قید مرابطه) مثل کلید اصلی و برخی چند مرابطه را درگیر خود می‌کنند (قید چندمرابطه‌ای) مثل کلید خارجی.

بررسی قیدهای تاپل و مرابطه فوری است یعنی پس از اجرای تک دستور چک می‌شوند.

بررسی قیدهای چندمرابطه‌ای

- در این مورد اختلاف نظر وجود دارد. برخی معتقدند که تراکنش واحد کاری پایگاه است و بررسی قیود باید تا آخر اجرای تراکنش به تعویق بیافتد و سپس اگر شرایط جامعیت برقرار نبود کل تراکنش واگردان شود. برخی دیگر معتقدند هر دستور یک واحد کاری جداست و قیدهای چندمرابطه‌ای (مانند قید تاپل و قید مرابطه) باید بلافاصله بعد از هر دستور چک شوند.
- موافقان تعویق کنترل قیدها (تا پایان تراکنش) می‌گویند ویژگی *انزوا* موجب می‌شود دیگران حالات درونی تراکنش را نبینند. مخالفان (که دیت هم جزء آنهاست) هم دلایلی دارند. مهمترین دلیل آنها این است که هر چند از بیرون کسی نمی‌بیند ولی هنوز بی‌ثباتی و ناسازگاری در خود تراکنش می‌تواند مشاهده شود و همین امر ممکن است موجب ایجاد پاسخ‌های غلط شود.

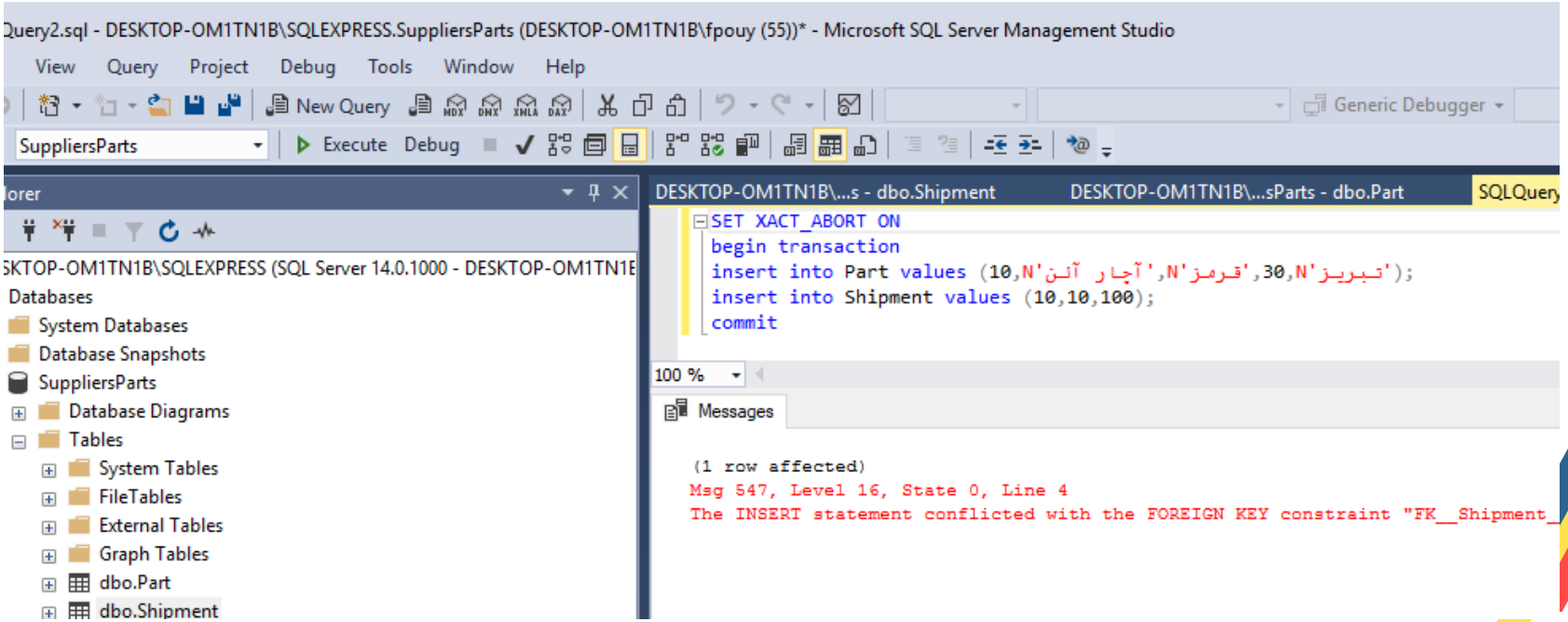
درستی و سازگاری

- طراحی تراکنش باید طوری باشد که نه فقط سازگاری consistency حفظ شود بلکه درستی correctness هم (تا جایی که ممکن است) حفظ شود. در مورد انتقال پول از حساب به حساب نمی‌توان قید جامعیت داشت (چون ممکن است کسی پول نقد به حساب بگذارد و حساب به حساب نکند)، با این حال باید در زمان حساب به حساب مجموع پول حساب‌های درگیر تغییر نکند.
- طراحی اصولی حکم می‌کند که تراکنش‌ها را به صورت تودرتو ننویسیم (با همروندی فرق می‌کند) چون با برگردانده شدن تراکنش بزرگتر، تراکنش کوچکتر با وجود آن که تکمیل شده باید برگردانده شود.

ایجاد تراکنش در SQLServer

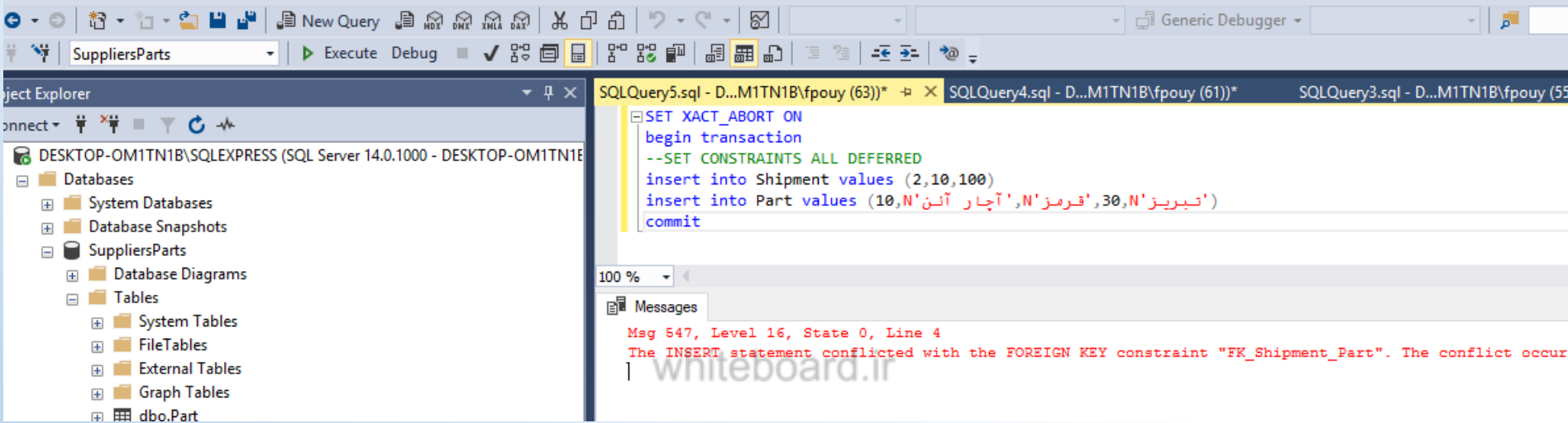
- هر تراکنش با begin transaction شروع شده و با commit تکمیل یا با rollback واگردانی می‌شود.
- اگر بخواهیم مانند یک تراکنش واقعی در صورت شکست یک دستور، کل تراکنش واگردان شود، باید xact_abort را روشن on کنیم (سطر اول).

- تراکنش زیر اول یک قطعه (آچار آلن) به رابطه قطعات اضافه می‌کند و بعد می‌خواهد یک سفارش برای آن ثبت کند اما چون توزیع‌کننده‌ای به شماره 10 وجود ندارد شکست می‌خورد. در نتیجه پس از اجرای آن اثری از آچار آلن هم در پایگاه نمی‌یابید.



SQL Server و تعویق کنترل جامعیت

- SQLServer بعد از ورژن 2000 از تعویق کنترل جامعیت پشتیبانی نمی‌کند و قیود را حتما فوری بررسی می‌کند.
- تراکنش زیر (اگر تراکنش را واحد کاری پایگاه حساب کنیم) باید کار کند (به جابجا بودن دو دستور توجه کنید)، اما با شکست مواجه می‌شود. در ورژن‌های قدیمی با اجرای قسمت سبز این تراکنش قابل اجرا بوده است.



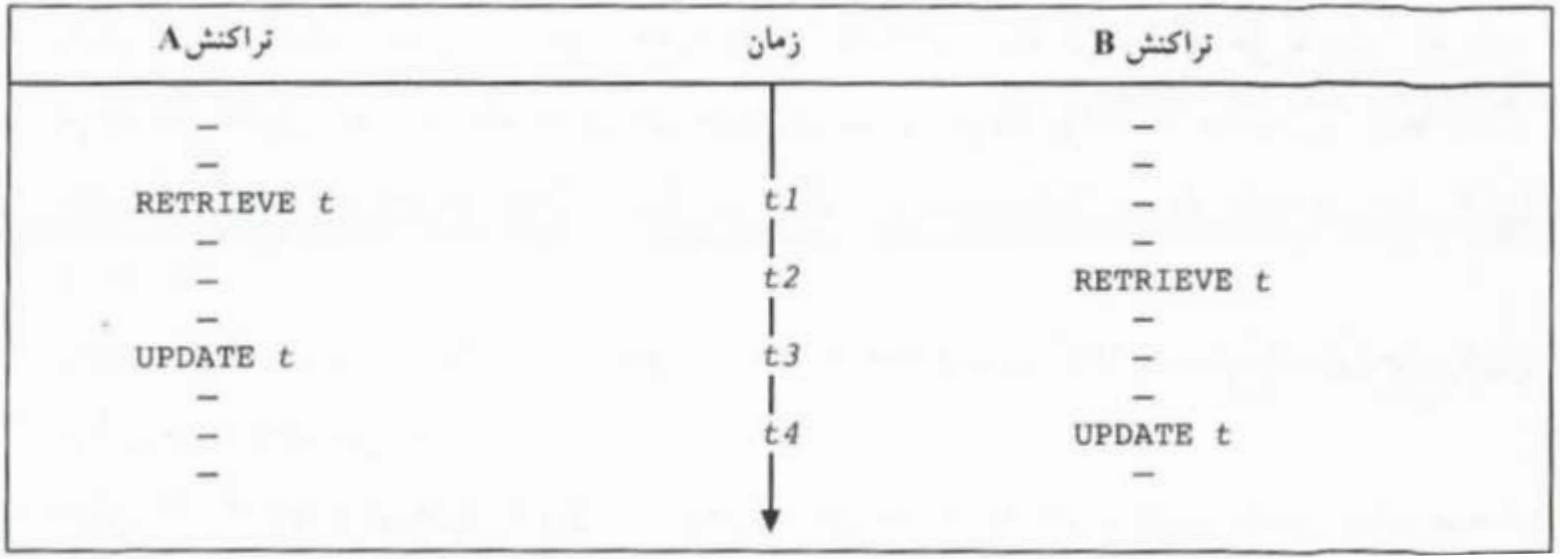
همروندی Concurrency

- ما تراکنش‌ها را تو در تو نمی‌نویسیم، اما این تضمین نمی‌کند که تراکنش‌ها خودشان به صورت همزمان روی پایگاه اجرا نشوند.
- مثلا در محیط شبکه که کاربران و برنامه‌های مختلفی در حال اجرا و کار با پایگاه هستند کاملا محتمل است که در میانه اجرای یک تراکنش، تراکنش دیگری نیز اجرا شود.
- این امر ممکن است مشکلاتی بوجود آورد که با مدیریت صحیح (در واقع پیاده‌سازی انزوا) می‌توان همروندی (اجرای درست و همزمان تراکنش‌ها) را در پایگاه داشت.
- این مشکلات سه گونه‌اند.

مشکل گم‌شدن به روزرسانی‌ها

- فرض کنید یک تراکنش قرار است ۵ واحد به مقدار ویژگی اضافه کند. تراکنش دیگر قرار است ۲ واحد به آن ویژگی اضافه کند. بنابراین پس از تکمیل دو تراکنش، برای همه تاپل‌ها ۷ واحد باید به آن ویژگی اضافه شده باشد. از قضا دو تراکنش همزمان اجرا می‌شوند.
- به روزرسانی این‌گونه است که اول مقدار ویژگی خوانده (retrieve) می‌شود و بعد از محاسبه مقدار جدید به روزرسانی (update) انجام می‌گیرد.

مشکل گم‌شدن به روزرسانی‌ها



- همانطور که می‌بینید در مورد تاپل t تداخل دو تراکنش موجب شده که یکی از به‌روزرسانی‌ها گم شود و در واقع فقط یکی از آنها انجام شود.

مشکل وابستگی به مقدار تکمیل نشده

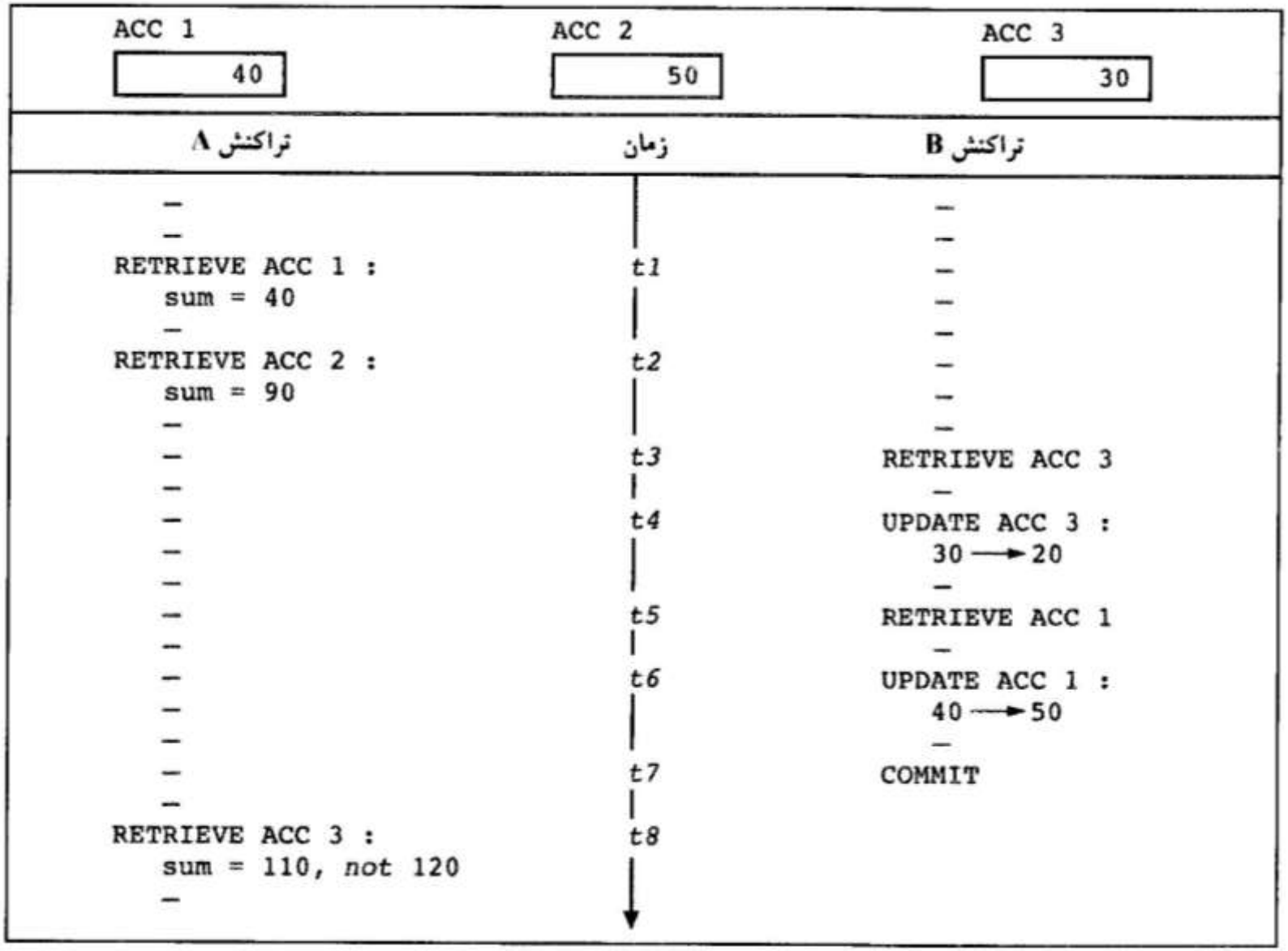
- تراکنش تا وقتی تکمیل commit نشده امکان شکست و واگردانی‌اش وجود دارد.
- ممکن است مقداری که تراکنش آن را به روزرسانی کرده، در میانه تراکنش، توسط تراکنش دیگری خوانده شود و بعد تراکنش واگردان شود. در این صورت مقداری که خوانده شده غیر واقعی است.
- ممکن است یک تراکنش مقداری را به روزرسانی کند، هنوز این تراکنش تکمیل نشده که تراکنش دیگری هم آن را دوباره به روزرسانی می‌کند. حال اگر تراکنش اول واگردان شود، به روز رسانی دوم نیز واگردان و در واقع گم می‌شود.

تراکنش A	زمان	تراکنش B
–		–
–		–
–	t1	UPDATE t
–		–
RETRIEVE t	t2	–
–		–
–	t3	ROLLBACK
–	↓	

تراکنش A	زمان	تراکنش B
–		–
–		–
–	t1	UPDATE t
–		–
UPDATE t	t2	–
–		–
–	t3	ROLLBACK
–	↓	

مشکل تحلیل ناسازگار

- فرض کنید یک تراکنش (که کارش فقط خواندن است) می‌خواهد مجموع موجودی همه حساب‌های بانکی را حساب کند (اینجا فقط سه حساب هست).
- در میان این تراکنش، یک تراکنش دیگر که کارش حساب به حساب کردن وجه است اجرا می‌شود.
- ببینید چگونه تداخل این دو تراکنش موجب می‌شود موجودی حساب‌ها اشتباه محاسبه گردد.



مشکلات تراکنش‌ها بر روی داده مشترک (خلاصه)

- خواندن-خواندن: مشکلی پیش نمی‌آید.
- خواندن-نوشتن: (تراکنش اولی داده را می‌خواند و دومی آن را بروزرسانی می‌کند) ممکن است باعث تحلیل ناسازگار شود.
- نوشتن-خواندن: ممکن است موجب وابستگی به مقدار تثبیت نشده شود.
- نوشتن-نوشتن: ممکن است موجب گم شدن به روز رسانی گردد.

راه حل: قفل‌گذاری

- تکنیکی که می‌تواند این مشکلات همروندی را حل کند قفل‌گذاری نام دارد.
- در این روش یک تراکنش می‌تواند تا زمانی که کارش با داده‌ها تمام نشده، آنها را قفل کند تا تراکنش‌های دیگر نتوانند آنها را تغییر دهند، تا زمان که تراکنش تکمیل شود و داده‌های قفل شده آزاد شوند.
- داده قفل شده می‌تواند یک ویژگی تاپل، کل تاپل، کل رابطه یا حتی کل پایگاه داده باشد.
- قفل گذاری به صورت خودکار توسط DBMS انجام می‌شود و کاربر در آن دخالتی ندارد.

انواع قفل

- قفل X: قفل انحصاری eXclusive است. اگر این قفل ایجاد شود تراکنش دیگری نه می‌تواند در داده‌ها بنویسد و نه می‌تواند هیچ قفل جدیدی روی آنها بگذارد.
- قفل S: قفل اشتراکی Shared که در صورت وجود آن کسی نمی‌تواند قفل X بزند و در داده‌ها بنویسد اما می‌تواند قفل S جدیدی بر داده‌ها بزند و آنها را بخواند.
- **تراکنشی که قصد خواندن دارد قفل S و اگر قصد نوشتن دارد قفل X بر داده می‌زند.**
- به این صورت قفل‌گذاری حداقل تاثیر منفی را بر کارایی دارد.

نتایج قفل کردن داده‌ها

- با قفل زدن هر سه مشکلی که در مورد همروندی ذکر شد حل می‌شود.
- اما مشکل جدیدی که به وجود می‌آید مخمصه Deadlock (بن بست) است. مخمصه وقتی به وجود می‌آید که دو تراکنش هر کدام به داده‌ای نیاز دارند که تراکنش مقابل آن را قفل کرده است. الگوریتم‌های مقابله با مخمصه در این مورد کارایی دارند.
- در مورد سوال اسلاید اول، هنگام تغییر دستی داده‌های جدول در SQLServer وقتی ویرایش شروع می‌شود (با تغییر مقدار) داده قفل می‌شود و خارج شدن مکان‌نما از آن سطر موجب تکمیل تراکنش و رهاسازی قفل می‌گردد.

فایده آگاهی در مورد همروندی

- همروندی و تراکنش‌ها ربطی به مدل رابطه‌ای ندارد و در هر مدل پایگاه داده این مسائل به صورت یکسان وجود دارد.
- زبان SQL هیچ دستوری در مورد قفل گذاری داده‌ها ندارد. این کارها به صورت کاملا خودکار انجام می‌شود.
- با این حال طراح پایگاه باید این مسائل را به خوبی بشناسد، چون در این صورت پروسه‌هایی که به صورت طولانی مدت داده‌ها را قفل می‌کنند را خواهد شناخت و برای جلوگیری از کند شدن سیستم می‌تواند کوئری‌ها را بصورت بهینه بنویسید یا کارهایی را به زمان خلوت بودن سیستم مثلا نیمه شب موکول کند.

تمرین

- پروسیجری بنویسید که دو پارامتر عددی دریافت کند. یک توزیع کننده به شماره پارامتر اول و نام محسنی با امتیاز ۴۰ و شهر تهران ثبت کند و بعد یک سفارش برای محسنی به شماره قطعه پارامتر دوم و تعداد ۲۰۰ برای آن ثبت کند و در پایان چک کند که اگر در مجموع برای آن قطعه بیش از ۳ سفارش (با احتساب سفارش جاری هست) همه کارها را واگردان rollback کند. بدیهی است اگر هنگام ثبت سفارش یا توزیع کننده سیستم جلوگیری کرد هم همه چیز باید واگردان شود.

پیاده‌سازی سیستم‌های پایگاه داده

جلسه دهم

امنیت پایگاه داده

و پشتیبان‌گیری

مبانی امنیت پایگاه داده‌ها

- امنیت در سیستم‌های کامپیوتری در جایی لازم است که افراد متفرقه نباید داده‌های خاصی را ببینند (دستیابی)، آنها را حذف یا اضافه کنند یا تغییر دهند (دستکاری).
- اصول امنیتی و اصول جامعیت از یک نظر شبیه هستند چون اولی داده‌های را از خرابکاری کاربران غیرمجاز و دومی از خرابکاری کاربران مجاز حفظ می‌کند.
- اصلی‌ترین روش برقراری امنیت کنترل هویت افراد با استفاده از **نام کاربری و کلمه عبور** است.
- هر چند این تنها روش ممکن نیست و روش‌های دیگر مانند کنترل داده‌های بیومتریک (که نیاز به نصب تجهیزات اضافه مانند حسگر اثر انگشت دارد) و یا کنترل محل فرد در شبکه با شماره IP (که انعطاف‌پذیری سیستم را کم می‌کند) نیز ممکن است.

ذخیره سازی کلمه عبور

- کلمه عبور کاربران تحت هیچ شرایطی نباید بصورت ساده در پایگاه داده ذخیره شود. در صورت انجام این کار اولا در صورت هک سیستم کلمه عبور تمامی کاربران بدست هکر می‌افتد، ثانیا مدیران فنی پایگاه داده که احتمالا به همه پایگاه دسترسی دارند می‌توانند به کلمه‌های عبور دسترسی داشته باشند (کارت‌های بانکی امن شمرده میشوند چون هیچ مقامی در بانک‌ها توان پیدا کردن رمز کارت من را ندارد).
- کلمه‌های عبور با الگوریتم‌های درهم‌سازی Hashing تبدیل به یک رشته کد (معمولا کوچکتر) می‌شوند. کلمه کد شده یکتا نیست (دو رمز ممکن است پس از کد شدن تبدیل به یک کد شوند، هرچند احتمال آن بسیار کم است).

ذخیره سازی کلمه عبور

- هر چند که با یک محاسبه ساده امکان تبدیل کلمه عبور به عبارت کد (درهم سازی) شده وجود دارد، اما عکس این مسیر غیرممکن است و با هیچ محاسبه‌ای نمی‌توان از کلمه کد شده به اصل کلمه عبور رسید.
- در سیستم‌های کامپیوتری باید بلافاصله پس از ورود کلمه عبور توسط کاربر آن را درهم سازی نموده و سپس برای ثبت در سیستم یا مقایسه، از کلمه کد شده استفاده نمود. درهم سازی نوعی رمزگذاری با کلید یکطرفه است. یعنی هیچ کلیدی برای گشودن متن رمز شده وجود ندارد.
- معمولا از الگوریتم MD5 استفاده می‌شود. برای استفاده از آن کافیست از تابع آماده که در کتابخانه ابزارهای برنامه‌نویسی امروزی وجود دارد، استفاده شود.

دو روش امنیتی SQLServer

- وقتی DBMS ما SQLServer باشد، در استفاده از دو روش امنیتی مختاریم.
- اولی اینکه شناسایی کاربران بطور کلی به ویندوز واگذار شود. یعنی برای هر کدام از کاربران که قبلا با پسورد خود وارد ویندوز شده‌اند اختیاراتی در نظر بگیریم که مثلا می‌تواند اطلاعات فلان جدول را تغییر دهد و یا فلان جدول را فقط بخواند.
- این روش فقط وقتی ممکن است پایگاه داده روی یک کامپیوتر یا یک شبکه محلی که توسط ویندوز سرور اداره می‌شود اجرا شود.

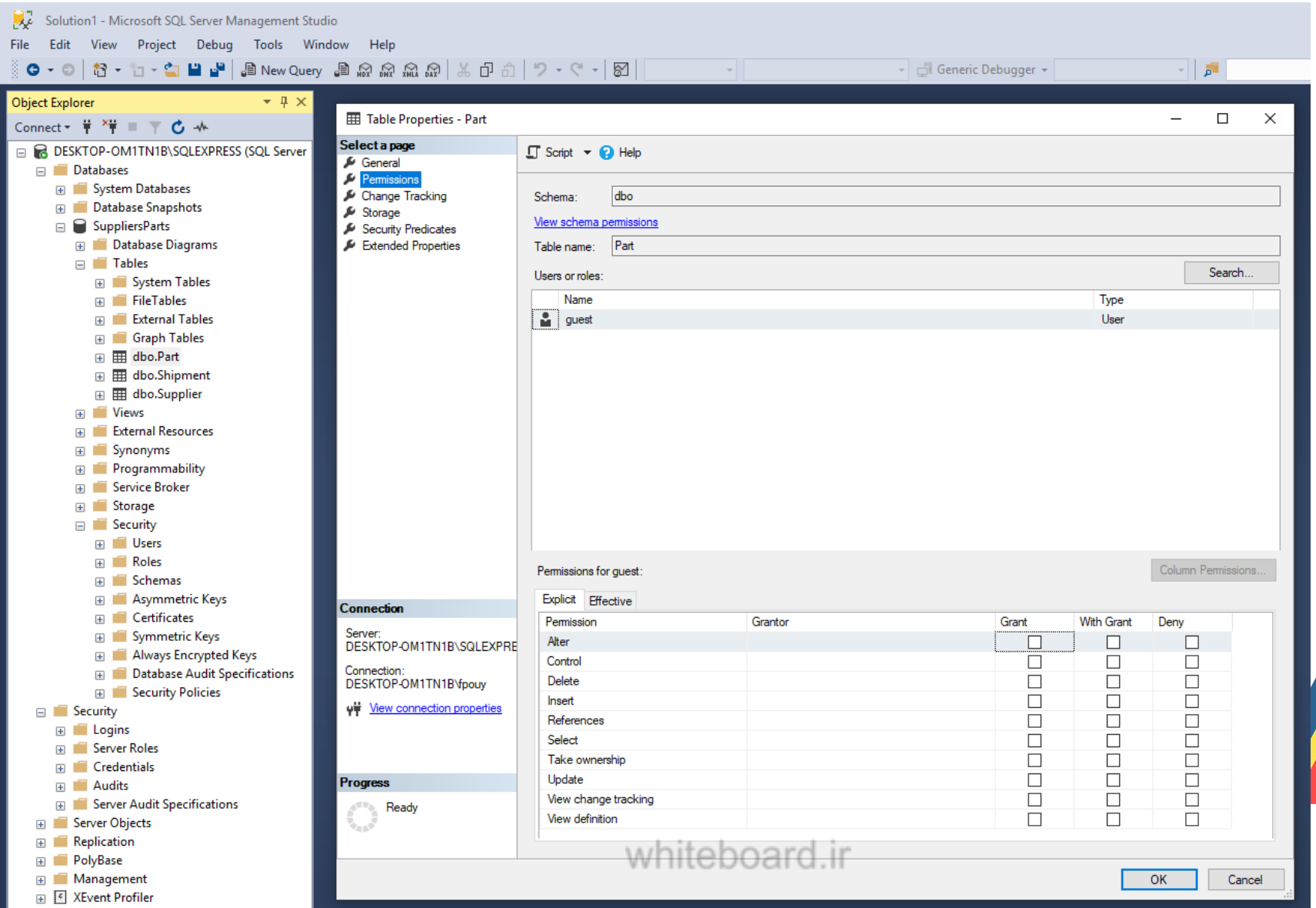
دو روش امنیتی SQLServer

- در روش دوم هر کاربر برای ورود به SQLServer یک نام کاربری و پسورد دارد که با استفاده از آن می‌تواند به پایگاه متصل شود.
- تکیه کردن تنها به این روش و سپس قرار دادن نام کاربری و کلمه عبور ثابت درون اپلیکیشن روش مناسبی نیست چون برخلاف DBMS که غیر قابل نفوذ است، به سادگی می‌توان فایل‌های اجرایی را تبدیل به سورس کد کرد و کلمه‌های عبور را از درون آنها استخراج نمود (مثلا یک روش آنکه دسترسی آن فقط به پروسیجرها و بسیار محدود و حساب شده باشد و کلمه عبور کاربر برای هر کاری به پروسجر ارسال شود).

تعیین دسترسی کاربران

- هنگام ساخت یک کاربر جدید می‌توان تعیین کرد که او چه role دارد و به کدام پایگاه ها دسترسی دارد. این کار با دستورات SQL و یا بصورت دستی از قسمت security و بعد logins انجام می‌شود.
- برای تعیین دسترسی مثلا به یک جدول خاص با بازکردن properties آن جدول و بعد permissions این کار انجام می‌شود.
- بهتر است کاربران معمولی به جدول‌ها هیچ دسترسی نداشته باشند و تمام کارهایشان را از طریق پروسیجرها انجام دهند.

تعیین دسترسی کاربران

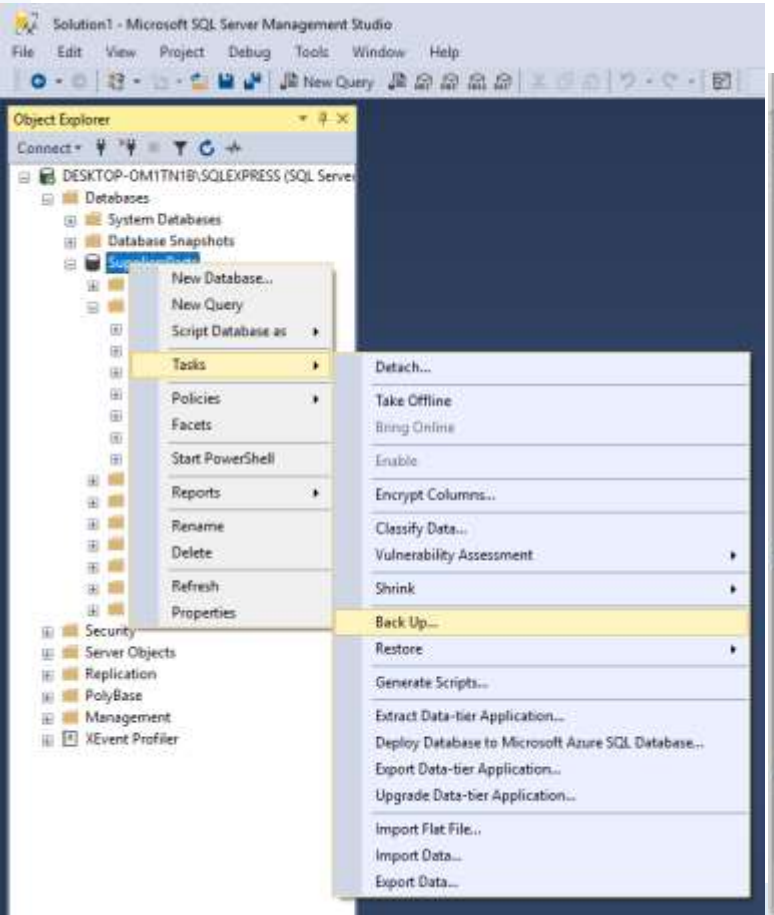


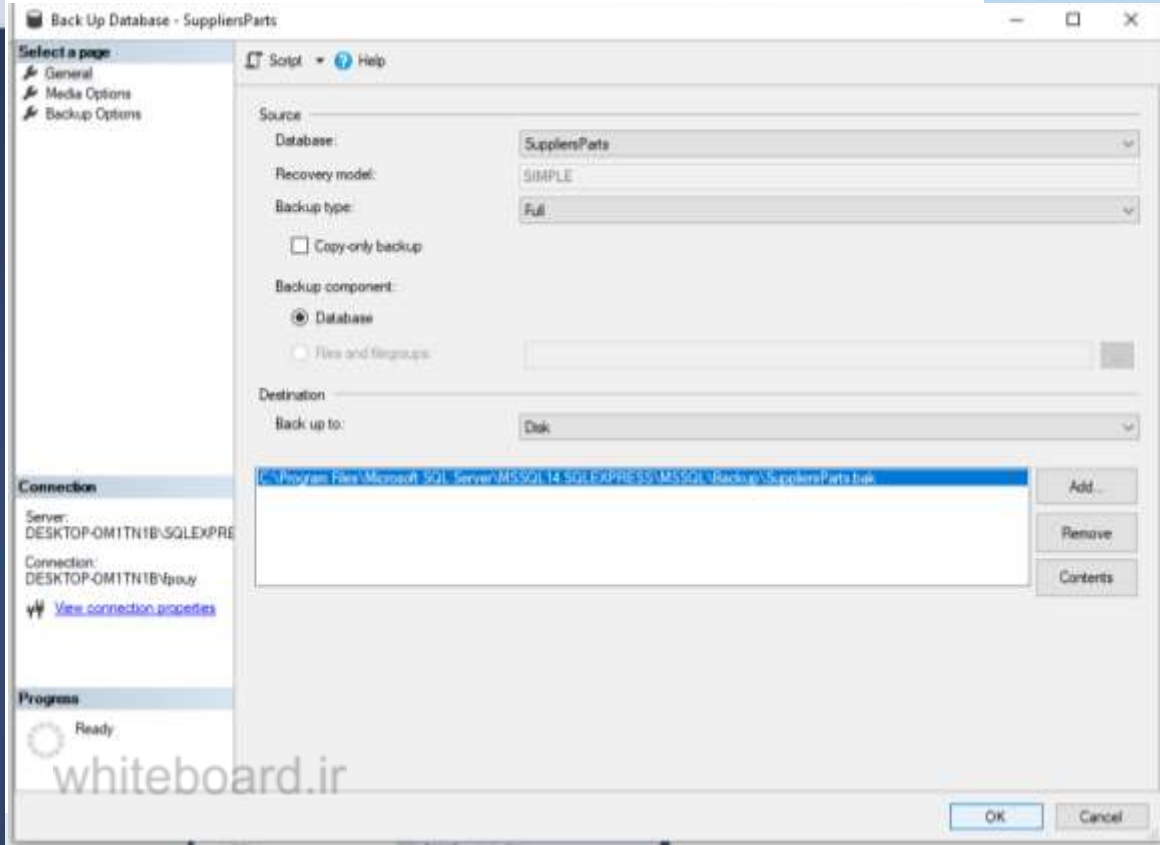
پشتیبان‌گیری و امنیت وجودی

- تبدیل بایگانی کاغذی به پایگاه داده کامپیوتری به خودی خود امنیت وجودی داده‌ها را تضمین نمی‌کند. آتش‌سوزی، سرقت و بلایای طبیعی هر دو (کاغذ و کامپیوتر) را تهدید می‌کند و علاوه بر آن سیستم‌های کامپیوتری همیشه در معرض خرابی هستند.
- اما با یک برنامه پشتیبان‌گیری منظم می‌توان پایگاه داده را از نابودی محافظت کرد.

پشتیبان‌گیری دستی

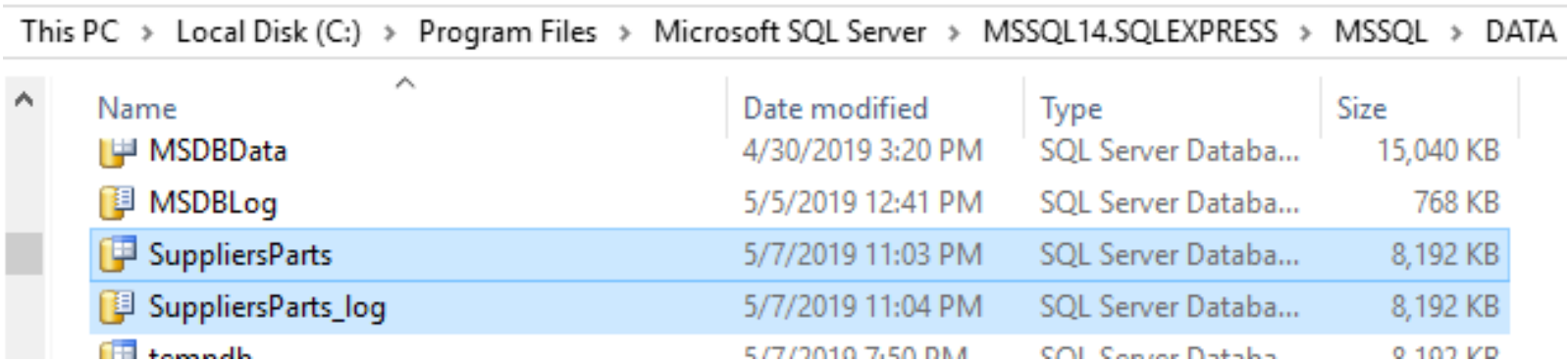
- با کلیک راست روی نام دیتابیس و ... صفحه ساخت بک‌آپ باز می‌شود که آدرس فایل را می‌توان نوشت...





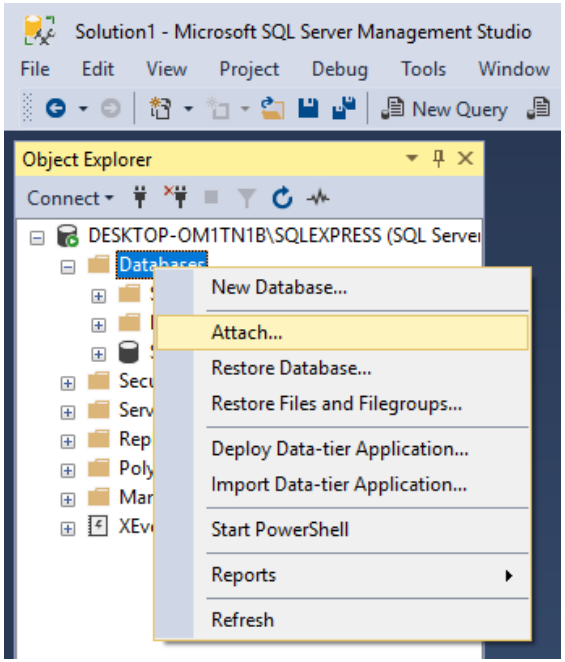
کپی کردن فایل‌های پایگاه

- هر پایگاه داده دارای دو فایل است که روی هارد دیسک ذخیره می‌شوند. با کپی کردن این فایل‌ها می‌توان پایگاه را پشتیبان‌گیری کرد و یا عینا به کامپیوتر دیگری منتقل نمود.



کپی کردن فایل‌های پایگاه

- البته باید قبل از انتقال این فایل‌ها را از DBMS جدا Detach کرد. برای این کار در همان جا (دو اسلاید قبل) ابتدا پایگاه را offline و سپس detach می‌کنیم و حال می‌توانیم فایل ها را هر کجا می‌خواهیم کپی کنیم.



- برای متصل کردن هم روی databases کلیک راست

و ....

نگهداری فایل‌های پشتیبان

- بدیهی است که نگهداری فایل‌های بک‌آپ روی همان کامپیوتر سرور بی‌فایده است. بلکه باید آنها را در مکانی دیگر نگهداری کرد. در گذشته با کپی کردن فایل‌ها روی دیسکت و CD و هارد قابل حمل و انتقال آن به گاوصندوق یا مکان دیگر آنها را محافظت می‌کردند.
- پایگاه‌های داده عظیم و با ارزش، همزمان در چند شهر یا کشور مختلف نگهداری می‌شود.
- امروزه سیستم‌های عمومی ذخیره‌سازی ابری مانند گوگل درایو و dropbox می‌توانند از اطلاعات و فایل‌های ارزشمند به راحتی محافظت نمایند.

آیینه‌سازی پایگاه

- در مورد برخی کاربردها، تنها پشتیبان‌گیری کفایت می‌کند. در این صورت هنگام خرابی سیستم دقایق یا ساعت‌هایی طول خواهد کشید که سیستم تعمیر و دوباره راه‌اندازی شود و احتمال دارد که داده‌های اندکی هم که پس از پشتیبان‌گیری وارد سیستم شده‌اند از بین برود.
- اما در برخی دیگر کاربردها خرابی سیستم حتی برای یک ثانیه در ماه قابل تحمل نیست و نیز نابودی هیچ داده‌ای نباید اتفاق بیافتد.

آیینه‌سازی پایگاه

- در چنین مواردی تکنیک آیینه‌سازی سرورها مورد استفاده قرار می‌گیرد به این صورت که چند سرور (احتمالا با فاصله مکانی دور از هم) پایگاه داده را نگهداری می‌کنند.
- داده‌های این سرورها دقیقا کپی هم هستند. هر تراکنشی که داده‌ای را ثبت می‌کند در صورتی تکمیل می‌شود که اطلاعاتش در تمام سرورها ثبت شده باشد.
- با از کار افتادن یکی از سرورها، آن سرور از دور خارج می‌شود و سیستم با سرورهای باقیمانده به کار ادامه می‌دهد. کارایی برای خواندن هم افزایش می‌یابد.
- SQL Server دارای امکان آیینه‌سازی است (در نگارش‌های جدید از طریق امکانی به نام Always On availability group صورت می‌گیرد).

پیاده‌سازی سیستم‌های پایگاه داده

جلسه یازدهم

پایگاه داده توزیع شده

پایگاه داده توزیع‌شده چیست؟

- یک سیستم پایگاه داده توزیع شده، مجموعه‌ای از سایت‌هاست که به نحوی بصورت شبکه بهم متصل شده و در آن:
  ۱. هر سایت یک پایگاه داده کامل است.
  ۲. سایت‌ها موافقت دارند که طوری با هم کار کنند که ازدید کاربر انگار که همگی یک پایگاه داده هستند.
- ویژگی دوم (که از دید منطقی تمام پایگاه‌ها بر روی یک DBMS دیده می‌شوند)، **شفافیت** نامیده می‌شود.
- پایگاه‌ها معمولا از نظر فیزیکی متفرق (دور از هم) هستند، اما در عمل کافی است از نظر منطقی متفرق باشند، یعنی حتی ممکن است چند سایت روی یک ماشین ایجاد شوند (مثلا زمان تست اولیه سیستم).

مزایا

- حالتی را در نظر بگیرید که بانک‌های کشور (ملی، تجارت، آینده و ...) و یا دانشگاه‌های دولتی هر کدام [سایت] پایگاه داده خود را دارند. اما کل سیستم به شکل یک پایگاه داده توزیع شده درآید.
- در چنین شرایطی هر صاحب پایگاه، سخت‌افزار و نرم‌افزار مربوط به خودش را تهیه می‌کند. او در حالی که داده‌ها را در کامپیوترهای خود نگه می‌دارد قوانین امنیتی را وضع و پیاده‌سازی می‌کند و مدیر پایگاه داده‌ای که منصوب کرده آن را اداره می‌کند.
- با این حال کاربران در شرایطی می‌توانند دیتای سایت‌های مختلف را مانند دیتای خود ببینند و با آن کار کنند که این امر مثلا انتقال پول از حساب به حساب و یا ورود نمرات دانشجوی مهمان را آسان می‌کند.

اصول دوازده‌گانه

- برای رسیدن به یک پایگاه داده توزیع‌شده آرمانی (که دو هدف گفته شده را کاملا محقق کند)، ۱۲ اصل بایستی هنگام برپایی سیستم مورد توجه قرار گیرد.
- برخی از این اصول نیاز به برنامه‌نویسی دارند، برخی باید در امکانات DBMS ها پیش‌بینی شوند (که بعضی الان هم موجودند)، برخی نیز نیاز به امکانات و هماهنگی سخت‌افزاری و نرم‌افزاری دارند.

۱. خودمختاری محلی

- تمامی عملیات یک سایت باید در همان سایت اداره شود. یعنی از کار افتادن سایت X هیچ تاثیر منفی بر کار سایت Y نگذارد. چرا که هیچ کاری از کارهای سایت Y به گردن سایت X انداخته نشده است.
- مثلا از کار افتادن سیستم دانشگاه کرج هیچ تاثیری بر کارهای (از جمله انتخاب واحد) سایر واحدهای دانشگاه نگذارد.
- جاهایی که رسیدن به این هدف صد درصد ممکن نیست، رسیدن به خود مختاری «حتی‌الامکان» است.

۲. عدم اتکا به سایت مرکزی

- تمام سایت‌ها دارای ارزش و اعتبار یکسان هستند (چنین نیست که بخش عمده‌ای از کار سیستم بر عهده یک سایت باشد).
- بدیهی است در صورت عدم رعایت این اصل با از کار افتادن یک سایت (سایت مرکزی)، سیستم فلج خواهد شد. همان طور که مثلا اگر سیستم شتاب از کار بیافتد بسیاری از کارهای بانکی همه بانک‌ها متوقف می‌شود.

۳. عملکرد دائمی

- سیستم هم **قابل اعتماد** است (حتی با وجود خرابی برخی قسمت‌ها هنوز کار می‌کند) هم **دسترس‌پذیری** آن بالاست (در یک بازه زمانی طولانی، درصد زمان خرابی بسیار کم و یا صفر است).
- در سیستم با عملکرد دائمی «خاموش شدن برنامه‌ریزی نشده» که گاهی در بعضی نقاط سیستم پیش می‌آید موجب ضرر چندانی نیست، ضمن آنکه «خاموش شدن برنامه‌ریزی شده» اصلا نباید در سیستم باشد (یعنی برای هیچ گونه ارتقا و نگهداری و تغییر مجبور به خاموشی نباید باشیم).

۴. استقلال مکانی

- کاربر نباید بداند که داده‌ها کجا ذخیره شده‌اند. سیستم طوری رفتار می‌کند انگار که همه داده‌ها روی سایت محلی خود کاربر هستند.
- بدین ترتیب تغییر فیزیکی محل داده‌ها بدون هیچ تغییری در کار سیستم انجام می‌شود.

۵. استقلال نسبت به چند پاره شدن داده‌ها

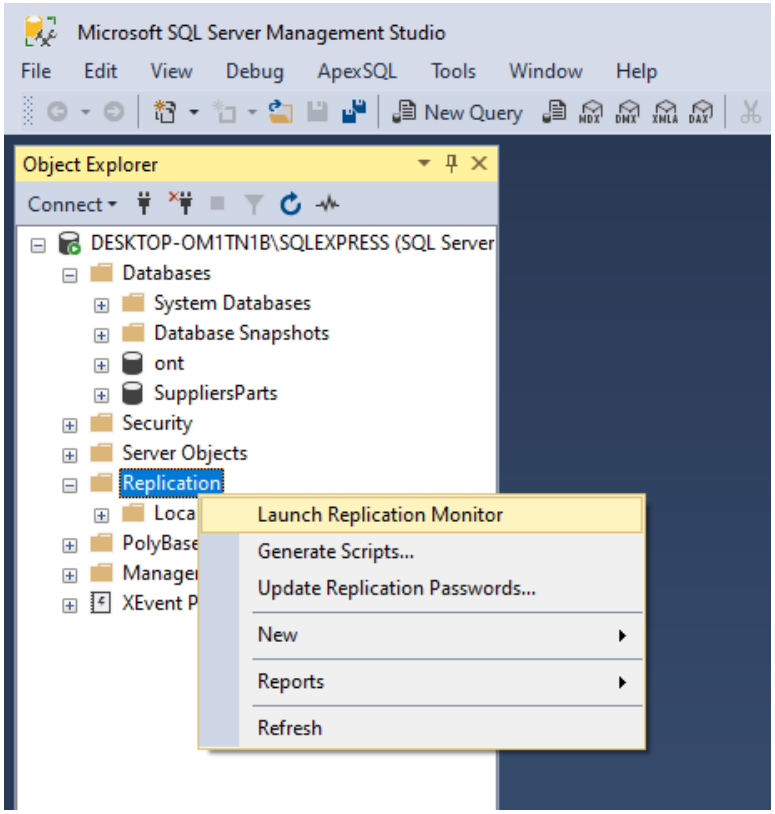
- چندپاره شدن یک جدول می‌تواند افقی یا عمودی باشد. مثلا در بحث تعامد دیدیم جدول توزیع‌کنندگان را می‌شود به دو جدول توزیع‌کنندگان پاریسی و غیر پاریسی تقسیم کنیم (افقی یا سطری). همچنین در بحث نرمال‌سازی دیدیم جدول را می‌شود با پرتو به چند جدول تقسیم کرد (تجزیه بدون کم و کاست، عمودی).
- پس از چندپاره‌کردن می‌توان هر بخش از جدول را در یک سایت ذخیره کرد که این کار از بار شبکه می‌کاهد. مثلا یک جدول دانشجویان داریم اما داده‌های دانشجویان هر واحد در سایت همان واحد ذخیره می‌شود. در این صورت کاربران که اکثرا بر روی شبکه محلی خود واحد به DBMS متصل می‌شوند ارتباطی سریع و مطمئن خواهند داشت.

۶. استقلال نسبت به تکثیر داده‌ها

- تکثیر داده‌ها بدین معناست که یک متغیر رابطه‌ای پایه [جدول] بتواند بصورت چندین کپی مجزا در چندین سایت ذخیره شود. مثلا دو سایت مستقل آموزش و صندوق رفاه هر کدام لیستی از دانشجویان (شماره دانشجویی، نام، ...) را داشته باشند.
- تکثیر داده‌ها موجب افزونگی نمی‌شود. چرا که تراکنش تغییر در این جدول در صورتی commit می‌شود که تغییر در همه جدول‌ها به انجام رسیده باشد. بنابراین وجود دیتای متناقض کلا غیر ممکن می‌شود. اگر سایتی خراب شود هم از دور خارج می‌شود و پس از راه‌اندازی باید ترمیم یا ریکاوری شود و سپس وارد سیستم گردد.

- تکثیر از یک طرف موجب کاهش بار شبکه و افزایش کارایی هنگام خواندن می‌شود (مثل چند پاره شدن) و از طرف دیگر دسترس‌پذیری را افزایش می‌دهد (وقتی سایتی خراب می‌شود سیستم هنوز قادر به ادامه کار است).
- سیستم باید طوری رفتار کند که کاربر در سطح منطقی متوجه تکثیر شدن داده‌ها نشود.
- آیینه‌سازی (که در بخش پشتیبان‌گیری مورد بحث قرار گرفت) به تکثیر داده شباهت دارد، اما در تکثیر یک یا چند جدول تکثیر می‌شوند، نه کل پایگاه و متعلقات آن.

- برای تکثیر یا Replication، سیستم SQL Server امکاناتی را در اختیار ما می‌گذارد. برای آزمایش این امکانات بایستی چند سایت (سرور) مجهز به SQL Server که بر روی شبکه به هم متصل شده‌اند (یا دست کم چند رجیستر برنامه روی یک کامپیوتر) را در اختیار داشته باشیم.



۷. بهینه‌سازی کوئری‌ها

- از آنجا که در اینجا، پردازش کوئری‌ها علاوه بر زمان خواندن و نوشتن بر روی دیسک و پردازش در پردازنده، احتمالا نیازمند انتقال حجم عظیمی از دیتا بر روی شبکه خواهد بود (مثلا موقع پیوند دو جدول ممکن است بر روی دو سایت دور از هم قرار داشته باشند)، بهینه سازی کوئری‌ها با توجه به این مساله باید انجام شود و دیگر فقط زمان کار CPU و IO ملاک نیست.

۸. مدیریت تراکنش‌ها

- ترمیم و کنترل همروندی نیازمند روش‌های پیشرفته‌تر است. قفل‌گذاری باید هنگام تکثیر بر روی چند نسخه از داده‌ها انجام شود.

استقلال ۹ سخت‌افزار
۱۰. سیستم‌عامل ۱۱. شبکه ۱۲. DBMS

- سایت‌های مختلف در **حالت ایده‌آل** باید بتوانند بر روی سخت افزارهای گوناگون (PC، Mac، ...)، سیستم‌عامل‌های مختلف (یونیکس، ویندوز،...)، شبکه‌های مختلف (اینترنت، LAN، ....) باشند.
- حتی نباید مجبور به استفاده از یک نوع DBMS برای تمامی سایت‌ها باشیم. در حالت ایده‌آل DBMS های ناهمگن (مثل اوراکل و SQL Server) باید بتوانند با هم تعامل داشته باشند. XML می‌تواند در این پروسه نقش اساسی داشته باشد.

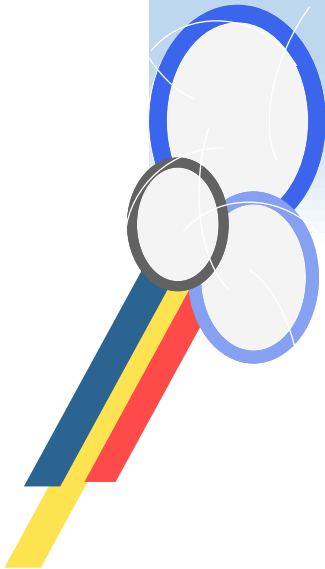
خلاصه

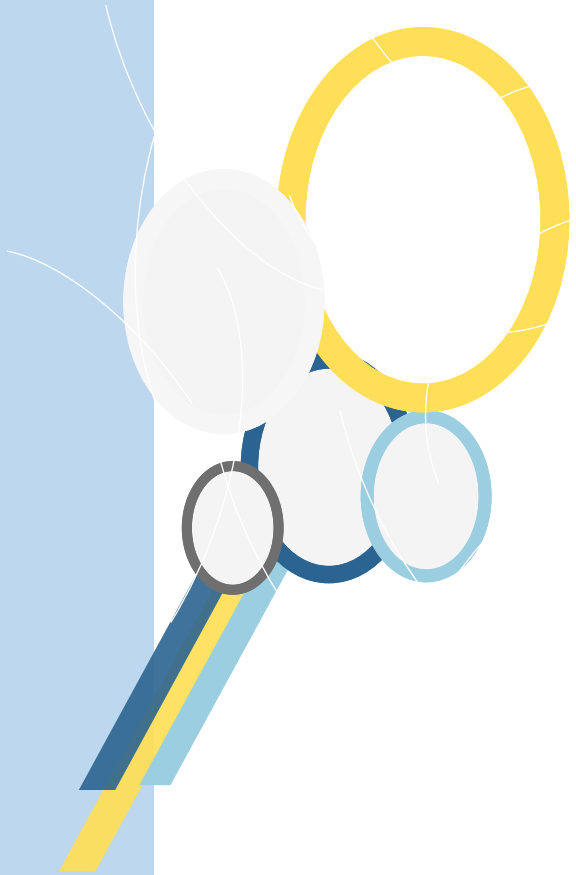
- تکنولوژی کنونی پایگاه داده برای پیاده‌سازی این دوازده اصل در مراحل مختلفی است. مثلا تکثیر اکنون تا حدود زیادی توسط سیستم‌ها قابل انجام است اما استقلال از DBMS هنوز چندان دست‌یافتنی نشده است. برخی مثل عدم اتکا به سایت مرکزی نیز نیازمند کار طراحی و پیاده‌سازی دشواری است.

تمرین

- فرض کنید که سیستم دانشگاه شامل چهار بخش آموزش (گروه و دانشکده)، مالی (شهریه‌ها)، صندوق رفاه (وام‌ها) و فارغ‌التحصیلی است. به نظر شما چه دیتایی باید در سایت هر کدام از این چهار بخش قرار گیرد و چه دیتایی بهتر است تکثیر شود؟ با یک دیتای فرضی جدول‌ها را طراحی کنید و با توجه به اینکه هر بخش یک سرور مستقل برای خود دارد بگویید که هر جدول باید در کدام سرور قرار گیرد. این تمرین نیازی به پیاده‌سازی روی سیستم ندارد و طراحی روی کاغذ کافی است.
- استقلال مکانی به چه معناست؟

Q&A





خسته نباشید