

فصل سوم

شناخت مدل رابطه‌ای و زبان بیانی SQL

هدف کلی

معرفی پایگاهداده رابطه‌ای و زبان بیانی SQL و توانایی‌های آن

هدف‌های یادگیری

۱. خوانندگان گرامی با مدل فرآگیر رابطه‌ای آشنا می‌شوند و سادگی و کارایی آن را می‌شناسند.
۲. پایگاهداده رابطه‌ای به عنوان یک مدل داده مطرح می‌شود و مفاهیم بنیادی مدل داده مانند جامعیت، امنیت، تراکنش، زبان پرسش در قالب آن با پایه علمی تبیین و عرضه می‌شود.
۳. زبان پرسش SQL که از دیرباز در مدل رابطه‌ای مطرح بوده است به خوانندگان معرفی می‌شود. سپس به طور گسترده، امکانات این زبان با ارائه یک پایگاهداده رابطه‌ای، قدم به قدم معرفی و در عمل نمایش داده می‌شود و پرسش‌های گوناگون بررسی می‌شود.
۴. نسل‌های مختلف زبان SQL به خوانندگان معرفی می‌شود و نسخه غالب آن یعنی SQL2 به صورت گسترده برای پیاده‌سازی یک پایگاهداده واقعی مورد استفاده قرار می‌گیرد. نسخه شئ‌گرای این زبان یعنی SQL3 نیز معرفی می‌شود تا در فصل‌های بعد مورد استفاده قرار گیرد.

مقدمه

آقای کاد^۱ با استفاده از دو مفهوم آشنا و نسبتاً ساده «جدول» و «مجموعه»، مدل رابطه‌ای را به وجود آورد [E.F.Codd. 1970]، یک پایگاه‌داده در مدل رابطه‌ای، مجموعه‌ای از جداول به هم پیوسته است. عملیات روی جداول مانند انتخاب سطر و ستون و ترکیب جداول با شکل‌های گوناگون مبنای پرسش در مدل رابطه‌ای هستند. مدل رابطه‌ای به عنوان مدل سنتی پایگاه‌داده و نیز مدل‌های وابسته به آن از چند دهه گذشته تاکنون پیشتاز دنیای نرم‌افزار هستند. توفیق آقای کاد مرهون روان‌شناسی اوست، که تئوری ساده و قابل فهم مجموعه‌ها را با مفهوم همگانی جدول درهم آمیخته و مدل قدرتمندی برای بیان مفاهیم انتزاعی و نیز پیاده‌سازی پایگاه‌داده پدید آورده است. در سال‌های متوالی، پس از همگانی‌شدن مدل رابطه‌ای، مقالات و کتاب‌های بی‌شماری درباره آن به رشته تحریر درآمده و پژوهش‌های پر حجم و ارزشمندی صورت گرفته و زوایای مختلف پایگاه‌داده در قالب این مدل روشن شده است. تأثیر این مدل تا حدی است که بسیاری از دست‌اندرکاران پایگاه‌داده، این پدیده و مفاهیم آن را در قالب مدل رابطه‌ای می‌شناسند، یعنی در ذهن خویش پایگاه‌داده را مجموعه‌ای از جداول تصور می‌کنند. طبیعی است که این مسیر نمی‌تواند از پیچ‌وخم عاری باشد. آقای کاد در کتاب خود می‌گوید: [E. F. Codd 1990]، ده سال طول کشید تا مجتمع علمی و تجاری کشورم را قانع کنم که مدل رابطه‌ای راه‌گشای پایگاه‌داده است.

۱-۳ مدل داده رابطه‌ای

مدل داده، یک محیط انتزاعی است که سه بخش ساختار داده‌ها، امکانات پردازش داده‌ها و امکانات تأمین جامعیت داده‌ها را یکجا فراهم می‌آورد. در مدل داده رابطه‌ای:

۱. تنها و تنها ساختار داده در مدل رابطه‌ای، همان رابطه است که به صورت جدول دو بعدی پیاده‌سازی می‌شود.

به جز مفاهیم ساده مانند نوع داده و تاپل و همچنین مقدار تهی، ساختار دیگری در مدل رابطه‌ای پذیرفته نیست؛ به عبارت دیگر همیشه و همواره داده‌ها به صورت رابطه نمایش داده می‌شوند، هر چند رابطه خالی یا هیچ مقدار باشد.

۲. امکانات پردازش داده‌ها در مدل رابطه‌ای از دو مفهوم رابطه (که به صورت جدول پیاده‌سازی می‌شود) و مجموعه وام گرفته شده است؛ مثلاً انتخاب ستون یا سطرهایی از یک جدول (معادل تاپل‌هایی از یک رابطه) و اشتراک دو رابطه و پیوند^۱ دو رابطه، یا ایجاد یک رابطه جدید، نمونه‌های ساده‌ای از امکانات پردازش داده‌ها در مدل رابطه‌ای هستند.

۳. تأمین جامعیت داده‌ها که عمده‌ترین هدف هر مدل داده‌ای است، به تعریف قواعد جامعیت^۲ می‌انجامد، مانند:

- هر رابطه باید کلید داشته باشد و نمی‌تواند هیچ مقدار باشد.
- مقدار کلید خارجی باید در جدول مرجع وجود داشته باشد.
- نام مؤلف می‌تواند تکراری باشد.

ما در ارائه مدل رابطه‌ای از منابع مختلفی چون [J. D. Ullman, J. Widom, 1997] و [H. F. Korth and A. Silberchatz, 2006] و [P. Rob and C. Coronel, 1997] سود جسته‌ایم.

مفهوم‌های اصلی در منابع مختلف به صورت مشابه تعریف شده‌اند. مهم‌ترین آن‌ها عبارت‌اند از:

- دامنه، مجموعه تمام مقادیر ممکن یک صفت است.
مثال: دامنه عدد صحیح یعنی مجموعه $\{L, 2, -1, 0, 1, 2, L\}$. صفتی که دامنه‌اش عدد صحیح است، می‌تواند هریک از این مقادیر را تا حد گنجایش کامپیوتر مربوطه در خود جای دهد.
- ضرب دکارتی^۳ دو دامنه D_1 و D_2 که به صورت $D_1 \times D_2$ نمایش داده می‌شود، تمام ترکیب‌های ممکن این دو دامنه را دربر می‌گیرد.
مثال: اگر داشته باشیم $D_1: Integer$ و $D_2: String$ ، آنگاه مجموعه تمام مقادیر ممکن زوج‌های مرتب (D_1, D_2) ، ضرب دکارتی است و هر زیرمجموعه از (D_1, D_2) ، یک رابطه است.

1. Join

2. Integrity Constraint

3. Cartesian Product

- ساده‌ترین راه نمایش و پیاده‌سازی رابطه، جدول دو بعدی است، مانند جدول زیر که دارای دو ستون برای نام و شماره شخص است:

D1:string	D2:Integer
مریم	۱۵
زهرا	۱۰
...	...

- زوج مرتب یا تاپل ارتباط مجموعه‌ای از مقادیر در یک رابطه است.
هر سطر یک جدول، معادل یک تاپل است و هر ستون یک جدول معرف یک صفت است. در مدل رابطه‌ای صفت‌ها از دامنه‌های ساده (تجزیه‌نپذیر) تعریف می‌شوند و دامنه‌های تودرتو مجاز نیستند. مثلاً اگر صفتی از نوع تاریخ تعریف شود دیگر اجزای آن (روز - ماه - سال) قابل دستیابی نیستند.

درباره تعریف رابطه باید به نکات زیر توجه کرد:

- نوع داده، یک مجموعه باز است که می‌تواند عضوهایی داشته باشد و عضوهای دیگری هم بپذیرد، مانند نوع داده کاربر - تعریف!

- هر رابطه خود یک نوع داده است که دو بخش دارد:

- شما^۲، یک متغیر رابطه‌ای است و شبیه متغیرها در زبان‌های برنامه‌سازی است که مستقل از مقدار آنها است. تعداد صفت‌های شمای هر رابطه را، درجه رابطه می‌نامند.

- بدنه، مجموعه تاپل‌های یک رابطه است. تعداد تاپل‌های هر رابطه را کاردينالیتی می‌نامند.

عمده‌ترین تفاوت‌های رابطه و جدول به صورت زیر است: (توجه کنیم که رابطه یک مجموعه است)

- رابطه نمی‌تواند تاپل تکراری داشته باشد، ولی جدول می‌تواند سطر تکراری داشته باشد.

۲. صفت‌های یک رابطه نظم مشخصی ندارد، ولی ستون‌های یک جدول می‌توانند نظم مشخصی داشته باشند.

۳. در رابطه، پیاده‌سازی مطرح نیست، ولی در جدول مفاهیم پیاده‌سازی مانند رکورد، ترتیب و محل صفت‌ها و... مطرح هستند.

۴. رابطه می‌تواند از درجه صفر باشد^۱، ولی نمی‌توان آن را به صورت جدول پیاده‌سازی کرد.

۵. رابطه می‌تواند n بعدی باشد، ولی جدول دو بعدی است.
با اندکی مسامحه، مفاهیم انتزاعی مانند رابطه، تاپل و صفت را با معادلهای پیاده‌سازی آن‌ها یعنی جدول، سطر و ستون، جایه‌جا به کار می‌بریم.
در فصل نرمال‌سازی به سایر خواص رابطه خواهیم پرداخت.

۲-۳ قواعد جامعیت

جامعیت در پایگاه‌داده‌ها یعنی هر تراکنش باید داده‌ها را به صورتی جامع بگیرد و به صورتی جامع تحويل دهد. داده‌ها در صورتی جامع هستند که صحیح، سازگار، دقیق و معتبر باشند.

برخلاف بعضی منابع، یک پایگاه‌داده همواره جامع نیست.
مثال: در پایگاه‌داده بانکداری باید جمع جبری واریزها و برداشت‌ها با موجودی بانک برابر باشد.

این قید جامعیتی، فقط در شروع و پایان تراکنش‌ها صحیح است، اما در حین اجرای تراکنش صحیح نیست، زیرا وقتی یک تراکنش مبلغی را از حسابی برداشته و هنوز به حساب دیگر واریز نکرده، این قید شکسته شده است.

نظام مدیریت پایگاه‌داده مسئول تضمین جامعیت آن است و باید عواملی چون اشتباه کاربران و برنامه‌های کاربردی و خرابی‌های سخت‌افزار و نرم‌افزار و تداخل تراکنش‌ها را کنترل کند.

مسئولیت اشتباه‌های غیرقابل کنترل به عهده کاربر است!
مثال: کاربری در انتقال پول، شماره کارت فردی را به غلط وارد و انتقال را تأیید

می‌کند. بانک مسئولیتی در قبال این اشتباه ندارد، هر چند پول کاربر را به حساب نادرستی منتقل کرده است.

بر مبنای معنای داده‌ها و نیز خواص پایگاهداده، تعدادی قید جامعیتی تعریف می‌شود. نظام مدیریت پایگاهداده‌ها در ابتدا و انتهای هر تراکنش، تمامی قیدهای مربوط به آن را بررسی می‌کند. تراکنش در صورتی ثبت شود که تمام قیدهای جامعیت را رعایت کرده باشد. در غیر این صورت ساقط می‌شود.

قیدهای جامعیت بر دو نوع هستند:

۱. قید عمومی از خواص پایگاهداده رابطه‌ای است و همواره باید رعایت شود. به این نوع قید، کلان قید یا کلان قاعده^۱ نیز گفته می‌شود. مثلاً قید «هیچ بخشی از کلید کاندید نمی‌تواند هیچ مقدار باشد»، یک کلان قید است.

در فصل نرمال‌سازی به این نوع قید جامعیت خواهیم پرداخت.

۲. قید کاربر-تعریف که در هر پایگاهداده ممکن است توسط کاربران مجاز تعریف شود. به عنوان مثال در پایگاهداده بانکداری قید «موجودی هر حساب حداقل n است» توسط بانک تعریف می‌شود. مبلغ n می‌تواند در بانک‌های مختلف متفاوت باشد ولی هیچگاه منفی نیست. این قواعد از معنای داده‌ها استخراج می‌شود. اینکه «نمره عددی بین صفر و بیست است» در نظام آموزشی کانونی ایران صحیح است. در بسیاری از کشورها «نمره عددی بین صفر و صد است». قیدهای جامعیتی کاربر-تعریف، در مرحله طراحی و نیز در مرحله نگهداری پایگاهداده تعریف می‌شود. در همین فصل به این مهم خواهیم پرداخت. در زبان SQL در هر زمان می‌توان این‌گونه قید جامعیت را تعریف کرد یا قیدهای موجود را تغییر داد. مسئولیت قیدهای کاربر-تعریف با مدیر پایگاهداده‌ها است. بسیاری از این قیدها هیچگاه قابل تغییر نیستند. مثلاً هیچ مدیر پایگاهداده‌ای نمی‌تواند موجودی یک حساب را، عددی منفی تعریف کند یا نمره زیر ده را، افتاده تلقی نکند. این‌گونه قیدها را، بیانی یا اعلانی^۲ می‌نامند.

یکی دیگر از روش‌های پیاده‌سازی قیود جامعیت، ماش^۳ نام دارد. ماشه مکانیزمی است برای اجرای خودکار یک قید یا قاعده، اعم از جامعیتی یا غیر از آن (مانند قواعد

1. Meta-Constraint

2. Declarative

3. Trigger

هوش مصنوعی). ماشه یک یا چند دستور آماده شلیک است که به محض بروز شرط مورد نظر، اجرا می‌شود. معمولاً شرایط داده در به روزرسانی آن تغییر می‌کند. ماشه با یک نام در سیستم شناخته می‌شود و قبل یا بعد از تغییر داده اجرا می‌شود.

مثال: پس از تغییر کلیدی در جدول کارکنان، تمام ارجاع‌ها به آن را در جداول ناشر و انتشار اصلاح کن.

طبعی است که این گونه موارد باید با یک زبان برنامه‌نویسی مانند SQL پیاده‌سازی شود.

نکته: در این مثال دیده می‌شود که کلید یک عضو جدول را می‌توان تغییر داد. این نکته یکی از نقطه ضعف‌های مدل رابطه‌ای است.

۳-۳ انواع کلید

هر رابطه دارای شناسه‌ای است که کلید نامیده می‌شود. کلید یک رابطه، غیرتکراری است و تمام عضوهای آن را به صورتی منحصر به فرد مشخص می‌کند. در مدل رابطه‌ای، کلید مجموعه‌ای از ویژگی‌ها یا صفت‌ها است. برای مشخص کردن کلید، زیر آن خط می‌کشند. مثلاً کلید رابطه مؤلف می‌تواند شماره ملی آن مؤلف باشد، زیرا هر مؤلف شماره ملی منحصر به فرد خود را دارد. در تعیین کلید رابطه، دو مورد زیر باید رعایت شود:

- کلید تکراری، وارد رابطه (جدول) نشود.
- مقدار تهی برای کلید وارد نشود.

اگر کلید رابطه‌ای تعیین نشود، به طور خود به خود همه صفت‌های آن رابطه، روی هم، کلید آن عضو تکراری داشته باشد.

کلید انواع مختلفی دارد که عبارت‌اند از:

۱. ابرکلید¹ (فراکلید): هر ترکیبی از صفت‌ها که خاصیت کلید داشته باشد یک ابرکلید است. این تنها نوع کلید است که لزوماً کمینه نیست، یعنی زیرمجموعه‌ای از آن هم ممکن است کلید باشد. مثلاً شماره ملی و نام مؤلف با هم ابرکلید هستند در صورتی که شماره ملی به تنها یی، کلید است.

۲. کلید کاندید یا کلید نامزد^۱: هر ترکیبی از صفت‌ها است که کمینه باشد و خاصیت کلید داشته باشد. یک رابطه ممکن است چند کلید کاندید داشته باشد. مثلاً اگر نام مؤلف غیرتکراری (کلید) باشد آنگاه حداقل برای رابطه مؤلف دو کلید کاندید داریم: یکی شماره ملی و دیگری نام مؤلف.

۳. کلید اصلی^۲: یکی از کلیدهای کاندید است که توسط طراح یا مدیر پایگاه داده انتخاب می‌شود. مثلاً در این نوشتار شماره ملی (با ارقامی کوتاه، غیررسمی و خودساخته) به عنوان کلید اصلی مؤلف انتخاب شده است.

یادآوری: کلید اصلی در جدول کتاب، شامل دو ستون *ano* و *bno* است، یعنی این دو ستون با هم دیگر کلید هستند، زیرا یک کتاب می‌تواند چند مؤلف داشته باشد.

۴. کلید فرعی^۳ (کلید ثانوی): یکی دیگر از کلیدهای کاندید است که برای برخی کاربردها انتخاب می‌شود. مثلاً در جدول مؤلف، نام مؤلف به عنوان کلید فرعی انتخاب می‌شود. این کلید فرعی در مواردی که با نام مؤلف کار داریم (از قبیل تهیه لیست‌ها) مورد استفاده قرار می‌گیرد، در صورتی که به تنهایی خاصیت کلیدی داشته باشد (یعنی نام تکراری نداشته باشیم)

۵. کلید خارجی^۴: صفتی است در یک رابطه که در رابطه دیگری کلید است (اصلی یا فرعی) و برای برقراری ارتباط بین دو رابطه استفاده می‌شود (ممکن است همان نباشند ولی ارتباط معنایی برقرار باشد). مثلاً در جدول نشر (Стон *pno* کلید خارجی است و ارتباط این جدول را با جدول کارکنان نشان می‌دهد. در جدول ناشر ستون *pno*، هم بخشی از کلید اصلی است و هم به تنهایی کلید خارجی است. در جدول کتاب نیز ستون *ano* که بخشی از کلید اصلی است، به تنهایی به عنوان کلید خارجی است و ارتباط این جدول را با جدول مؤلف برقرار می‌کند. بنابراین هرگاه بخواهیم سطر جدیدی به جدول کتاب اضافه کنیم، باید شرط اساسی زیر رعایت شود:

ستون *ano* باید مقداری داشته باشد که حتماً در جدول مؤلف وجود دارد.

1. Candidate Key

2. Primary Key

3. Alternative Key, Secondary Key

4. Foreign Key

۴-۳ انواع جامعیت در مدل رابطه‌ای

در مدل رابطه‌ای سه نوع جامعیت مورد تأکید قرار می‌گیرد:

۱. جامعیت دامنه‌ای^۱ یعنی تمام صفات در تمامی رابطه‌ها از نوع دامنه خود باشند. مثلاً ۷۴/۱ به عنوان کلید مؤلف که عدد صحیح مثبتی است پذیرفته نمی‌شود.
 ۲. جامعیت درون‌رابطه‌ای^۲، یعنی هر رابطه به تنها یکی صحیح باشد. مثلاً عضو تکراری نداشته باشد و کلیدهایش دارای مقادیر تهی یا تکراری نباشند.
 ۳. جامعیت ارجاع^۳، یعنی کلید خارجی درست تعریف شده باشد، به عبارت دیگر دو شرط زیر رعایت شده باشد:
 - کلید خارجی یک رابطه حتماً در رابطه دیگر کلید باشد.
 - مقداری که به کلید خارجی داده می‌شود در رابطه دیگر وجود داشته باشد.
- نرم‌افزار نظام مدیریت پایگاه‌داده همواره این خاصیت‌ها را کنترل می‌کند. مثلاً اگر درخواست وارد کردن کتابی را دریافت کند، ابتدا در جدول مؤلف کنترل می‌کند که شخص مربوطه به عنوان مؤلف وجود داشته باشد. در صورت عدم وجود، درخواست وارد کردن آن سطر در جدول کتاب رد می‌شود.

۵-۳ زبان روالی SQL

پیش از آنکه به پیچیدگی‌های مدل رابطه‌ای از دیدگاه نظری بپردازیم، ابتدا می‌کوشیم از نظر کاربردی به آن بنگریم تا خوانندگان ارجمند برداشتی بهتر و دقیق‌تر از آن داشته باشند و بتوانند پایگاه‌داده را در حافظه کامپیوتر ایجاد کنند و عملیات مختلفی روی آن انجام دهند. برای این منظور زبان پرکاربرد SQL را معرفی می‌کنیم و بخشی از پایگاه‌داده نشر کتاب را روی آن پیاده‌سازی می‌کنیم. SQL یک استاندارد است که سه نسل آن تا به حال ارائه شده است (SQL3, SQL2, SQL1). آنچه در جهان واقع وجود دارد پیاده‌سازی‌های متفاوتی از این استانداردها است که معمولاً بدنه اصلی یکسان، اما شاخ‌وبرگ و افزودنی‌های متفاوتی دارند. این زبان در سال ۱۹۸۶ توسط کمیته ANSI استاندارد شد. سپس در سال ۱۹۹۲ نسخه کامل‌تری از این زبان با نام SQL ۲ اعلام شد.

1. Domain Integrity
2. Intra-Relation Integrity
3. Referential Integrity

[ISO، 1992، G.Otten، S.Cannan] [1993]. این نسخه امتیازهای چندی نسبت به نسخه اول داشت. از آن جمله، دو نوع جدید داده BLOB و CLOB برای کار با داده‌هایی چون تصویر و متن‌های طولانی به آن اضافه شد.

در این فصل ابتدا به SQL1 و آنگاه به SQL2 می‌پردازیم و در فصل‌های بعدی به معرفی SQL3 که نسخه شئ-رابطه‌ای آن است خواهیم پرداخت.

زبان SQL بیانی (اعلانی) یا غیرروالی است، یعنی لازم نیست کاربر روال کار را بیان کند؛ مثلاً برای مرتب‌کردن یک جدول درخواست ORDER کافی است و برخلاف زبان‌های برنامه‌سازی سطح بالا، لازم نیست به حلقه‌های تودرتو و جابه‌جا کردن عضوهای جدول بپردازیم.

جنبه‌های پایگاه‌داده رابطه‌ای، اعم از تعریف جداول، تغییر آن‌ها (تغییر تصویر ادراکی)، حذف و اضافه کردن داده‌ها و تغییردادن محتوای جداول، با این زبان امکان‌پذیر است. همچنین تعریف کلیدهای مختلف، تهی‌نمایندن مقدار صفت‌ها، کترل‌های جامعیتی و غیره را می‌توان انجام داد. این زبان در حین اجرای دستورات کاربر، بسیاری از خطاهای او را تشخیص می‌دهد و از بُروز آن‌ها جلوگیری می‌کند. SQL مقبولیت عام یافته است. نه تنها رقبای خود را کنار زده، بلکه جای خالی زبان‌های پرس‌وجوی نسل بعد از خود را نیز پر کرده است تا جایی که امروزه در دنیای متفاوتی همچون مدل شئ-رابطه‌ای، در پرس‌وجو حرف اول را می‌زنند.

مثال‌های این فصل با نرم‌افزار Microsoft SQL Server 08 اجرا شده است. می‌توان از دیگر نرم‌افزارها که قابلیت پشتیبانی SQL3 را نیز دارند استفاده کرد. این گونه موارد موضوع درس جدأگانه‌ای به نام «آزمایشگاه پایگاه‌داده‌ها» است.

۱-۵-۳ جداول

طراحی جداول در مدل رابطه‌ای، عملی بسیار دقیق است. جداول باید به‌طور کامل تصویر ادراکی پایگاه‌داده را تداعی کنند و از قواعد جامعیت پیروی کنند و حتی الامکان نرم‌مال باشند. جداول پایگاه‌داده نشر کتاب که به صورت جداول در شکل ۱-۵ در فصل اول آمده است، در این فصل عیناً مورد استفاده قرار می‌گیرد. در تعریف جدول باید نام، صفت‌ها، دامنه آن‌ها، کلیدهای جدول و کترل‌های لازم روی صفت‌ها مشخص شود.

زبان SQL از دو بخش عمده تشکیل شده است: زبان تعریف داده‌ها^۱ و زبان کارکردن با داده‌ها^۲. این دو بخش به طور جداگانه مورد بررسی قرار می‌گیرند.

۲-۵-۳ انواع داده استاندارد

شکل ۳-۱ دامنه‌های تعریف شده در SQL استاندارد را نمایش می‌دهد.

توضیح:

۱. انواع داده DATE و TIME در همه نسخه‌های SQL پشتیبانی نمی‌شوند.

در بعضی نسخه‌ها نمونه‌های تاریخ و زمان به شکل زیر است:

```
date '2015-01-01'  
time '09:300:05'  
timestamp '2015-01-01 09:300:05'
```

۲. در SQL 2 انواع CLOB (برای متن‌های بلند) و BLOB (برای عکس و...)

پشتیبانی می‌شود.

مثال:

image BLOB (20 MB)
book CLOB (20 KB)

دامنه	شرح مختصر
Smallint	عدد صحیح کوچک‌تر از ۳۲۷۶۸ - ۳۲۷۶۷ تا
Integer	عدد صحیح تا ۱۰ رقم
Numeric (p, q)	عدد حقیقی ۱۹ رقم (قسمت صحیح و اعشاری باهم)
Decimal (n, m)	عدد حقیقی کوچک شامل m رقم اعشار و n-1 رقم صحیح
Float	عدد حقیقی با نقطه شناور تا ۱۹ رقم دقت
Char(n)	کاراکتر با طول ۱ تا ۲۵۶
Date	تاریخ با فرمت روز/ماه/سال
Time	زمان تا هزارم ثانیه
Logical	T و F (درست و نادرست)
Bit(n)	اعداد مبنای ۲

شکل ۳-۱. دامنه‌های شناخته شده در SQL استاندارد

۱. برای استفاده از کاراکترهای غیرمعمول مانند حروف فارسی می‌توان از نوع nvarchar(n) استفاده کرد.
۲. بعضی از دامنه‌های دیگر نیز در SQL استاندارد، تعریف نشده‌اند مانند LOGICAL DATA
۳. در دستورهای SQL بین حروف بزرگ و کوچک تفاوتی نیست (مثلاً DATE با date یکسان است) ولی ما می‌کوشیم دستورهای SQL را با حروف بزرگ و سایر موارد را با حروف کوچک بنویسیم.

۳-۵-۳ نوع داده کاربر-تعریف

می‌توان دامنه‌های جدیدی با دستور CREATE DOMAIN تعریف کرد. شکل کلی این دستور چنین است (کلمات درون کروشه دلخواه هستند):

```
CREATE DOMAIN name typ
[DEFAULT]
CHECK (...)
[CONSTRAINT]
```

در بعضی نسخه‌ها از CREATE TYPE استفاده می‌شود مانند:

```
CREATE TYPE euro AS NUMERIC(10,2);
```

مثال: هفتۀ ایرانی که از شنبه (Sat) شروع می‌شود.

```
CREATE DOMAIN week CHAR(8)
DEFAULT 'Sat'
CONSTRAINT IRANIAN
CHECK (VALUES IN, 'Sat', 'Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri');
```

دامنه‌های ایجادشده را می‌توان در صورت لزوم تغییر داد. هریک از بخش‌های CONSTRAINT, CHECK, DEFAULT قابل تغییر هستند. برای این‌منظور باید از دستور ALTER DOMAIN استفاده کرد. همچنین می‌توان چنین دامنه‌هایی را نابود کرد. دستور آن ;... است.

۴-۵-۳ تعریف پایگاهداده

برای تعریف پایگاهداده از دستور زیر استفاده می‌شود:

```
CREATE DATABASE db_name AUTHORIZATION
                    dba_name;
```

مثال: پایگاهداده نشر کتاب و اجازه دستیابی به آن

```
CREATE DATABASE publish AUTHORIZATION Haghjoo;
```

۵-۵-۳ تعریف جدول

جداول با دستور CREATE TABLE تعریف می‌شوند. شکل کلی آن‌ها چنین است:

```
CREATE TABLE name
(alter domain [NOT NULL]
.....)
alter ...
[PRIMARY KEY ...]
[UNIQUE ...]
[FOREIGN KEY ... REFERENCES ...
[ON DELETE CASCADE]
[ON UPDATE CASCADE]
FOREIGN KEY...]
[CHECK(...)]);

```

قبل از هر چیز از پایگاهداده نشر کتاب مثال می‌زنیم:

مثال: تعریف جدول book که تصویر ادراکی این جدول به صورت زیر است:

book(bno,title, dop, typ, version, ano)

CREATE TABLE book

```
(bno smallint ,
title nvarchar(30) NOT NULL,
dop date,
typ nvarchar(10),
version smallint,
ano smallint NOT NULL,
PRIMARY KEY (bno,ano),
FOREIGN KEY(ano) REFERENCES author
ON UPDATE CASCADE);
```

در این جدول bno کد کتاب است که به صورت عدد صحیح کوچک تعریف می‌شود (مثل ۲۰۱)، title نام کتاب است و تا ۳۰ کاراکتر متغیر است، dop تاریخ چاپ است از نوع date، typ نوع کتاب است، version نسخه کتاب و ano کلید خارجی مؤلف است که به جدول author ارجاع می‌شود. دو ستون bno و ano کلید اصلی جدول book هستند. این جدول به صورت بالا تعریف می‌شود.

تعریف صفت‌ها روشن است. مثلاً typ داده‌ای مانند «درسی» را در بر می‌گیرد و می‌تواند تهی باشد، زیرا قید NOT NULL ندارد، ولی سایر خصوصیات آن باید مشخص شود در غیر این صورت داده ورودی رد می‌شود و وارد جدول نخواهد شد. در این تعریف قید UNIQUE نداریم چون این جدول کلید فرعی ندارد، زیرا یک کتاب ممکن است چند مؤلف داشته باشد و بنابراین نام کتاب می‌تواند تکراری باشد.

دستور PRIMARY KEY کلید اصلی جدول را مشخص می‌کند. دستوراتی که در ادامه آمده اجباری نیست، یعنی بعضی جداول ممکن است کلید خارجی نداشته باشند.

دستور FOREIGN KEY کلید خارجی را مشخص می‌کند و نیز با قيد REFERENCES تعیین می‌کند که این کلید به کدام جدول ارجاع می‌شود. همچنین قیدهای ON UPDATE CASCADE و ON DELETE CASCADE می‌گویند که اگر زمانی این کلید در جدول اصلی حذف شد یا تغییر کرد، کدامیک در این جدول هم انجام شود. مثلاً اگر نام نویسنده‌ای از جدول نویسنده‌گان حذف شود باید آن را از جدول کتاب حذف کرد، اما اگر نام نویسنده‌ای عوض شود، باید نام جدید او را آورد. مثال: می‌توان جدولی با شمای مشابه یک جدول موجود با قيد LIKE تعريف کرد:

`CREATE TABLE temp_book LIKE book;`

شرطهای مندرج در دستور CHECK را کنترل می‌کند و از ورود داده‌های غلط به جداول جلوگیری می‌کند یا پیام مناسب می‌دهد یا عمل مورد نظر کاربر را انجام می‌دهد.

شکل ۲-۳ تعریف بعضی دیگر از جداول پایگاهداده نشر کتاب را که در پرسش‌های بعدی استفاده می‌شوند، نمایش می‌دهد:

```
CREATE TABLE personnel
(pno smallint ,
name nvarchar(20) NOT NULL,
typ nvarchar(10),
dob date,
sex char(1) NOT NULL,
adres nvarchar(100),
PRIMARY KEY (pno));
CREATE TABLE author
(ano smallint,
name nvarchar(20) NOT NULL,
typ nvarchar(10) ,
dob date,
sex CHAR(1) NOT NULL,
adress nvarchar(100),
degree nvarchar(10),
major nvarchar(10),
```

```

resum nvarchar(100),
PRIMARY KEY(ano);
CREATE TABLE publisher
(pno smallint,
title nvarchar(20) NOT NULL,
typ nvarchar(10),
doe date,
adres nvarchar(100) NOT NULL,
PRIMARY KEY(pno,typ),
FOREIGN KEY(pno) REFERENCES personel);
CREATE TABLE publish
(pubno smallint,
pno smallint NOT NULL,
bno smallint NOT NULL,
dop date NOT NULL,
Uprice dec(10,2),
num integer NOT NULL,
PRIMARY KEY(pubno));

```

شکل ۲-۳. تعریف بعضی از جداول پایگاهداده نشر کتاب

به جای CHECK می‌توان دستور ASSERTION را جداگانه تعریف کرد. خاصیت این دستور این است که اولاً به جدول نمی‌چسبد و همیشه می‌توان آن را تغییر داد و دوماً می‌توان کترل‌هایی روی ترکیب چند جدول داشت. شکل کلی آن چنین است:

`CREATE ASSERTION<name> CHECK (...)`

مثال: کتابی بدون نوع، تعریف نشده باشد.

`CREATE ASSERTION TypeNotNull
 CHECK (NOT EXISTS (SELECT * FROM book
 WHERE typ IS NULL));`

توضیح: ابتدا در تعریف جدول، نوع کتاب می‌توانست تهی باشد و احتمالاً تعدادی داده نیز وارد شده‌است، ولی پس از مدتی این تصمیم عوض می‌شود و دستور بالا به پایگاهداده اضافه می‌شود تا از این به بعد هر کتابی دارای نوع مشخص باشد.

شکل ۲-۳ تعریف بعضی از جداول پایگاهداده نشر را نشان می‌دهد. دستور CREATE DATABASE publish چارچوب پایگاهداده publish ذخیره می‌کند.

۳-۵-۶ تغییر جدول موجود

جداولی که زمانی تعریف شده‌اند، باید روزی تغییر کنند. زبان SQL این عمل را با دستور ALTER TABLE امکان‌پذیر می‌سازد. این دستور در حالت‌های مختلف زیر به کار می‌رود:

۱. تغییر نوع داده ستون

اگر بخواهیم نوع داده ستون یا ستون‌هایی را تغییر دهیم یا دستکاری کنیم، از دستور زیر استفاده می‌کنیم:

```
ALTER TABLE table-name
    ALTER COLUMN <col-name><new typ>;
```

مثال: بزرگ کردن اندازه ستون نام در جدول کارکنان (از ۲۰ کاراکتر کنونی به ۳۰ کاراکتر).

```
ALTER TABLE personnel
    ALTER COLUMN name NVARCHAR(30);
```

اگر جدول دارای داده‌هایی باشد، این داده‌ها خود به خود تبدیل می‌شوند. اگر بخواهیم نوع داده را به طور کلی عوض کنیم، اکثر نسخه‌های SQL جلوگیری می‌کنند مگر آنکه ستون‌های مربوطه فاقد داده باشند. مثلاً اگر بخواهیم تاریخ چاپ کتاب را به کاراکتر تغییر دهیم، ممکن است تبدیل داده‌ها به فرمت جدید دچار مشکل شود و بنابراین، از این عمل جلوگیری می‌شود (دستور زیر):

```
ALTER TABLE book
    ALTER COLUMN dop CHAR(11);
```

یادآوری: این تغییرات را در جداول اعمال نکرده‌ایم یعنی به شکل سابق خود هستند.

توضیح: این دستور در SQL استاندارد و نیز اوراکل 10G به بعد و نسخه‌هایی از SQL server به صورت زیر است:

```
ALTER TABLE table-name
    MODIFY (<col-name><new typ>);
```

۲. افزودن ستون

شکل کلی دستور آن چنین است.

```
ALTER TABLE table-name
    ADD <col-name><typ>;
```

مثال: ستون مرجع resum (رزومه) را با فرمت char(100) به جدول personel اضافه کنید.

```
ALTER TABLE personel
ADD resum Char(100);
```

۳. حذف ستون

چنین امکانی به‌طور مستقیم در SQL پیش‌بینی نشده است، زیرا حذف ستون ممکن است تأثیر جبران‌ناپذیری روی ارتباط جداول با یکدیگر گذارد. با این‌همه می‌توان جدول جدیدی تعریف کرد که فاقد ستون‌های مورد نظر باشد و سپس ستون‌هایی را که باقی می‌مانند از جدول قدیم به جدول جدید کپی کرد. آنگاه جدول قدیم را خالی کرد و سپس از بین برد (با دستور DROP).

۷-۵-۲ بارگذاری پایگاهداده

تا اینجا به شمای پایگاهداده پرداختیم. حالا داده‌ها را وارد، حذف و اصلاح می‌کنیم:

۱. افزودن داده‌ها

افزودن داده‌ها به صورت سطرهای کامل انجام می‌شود. برای بعضی از ستون‌ها که قید NOT NULL ذکر نشده باشد، می‌توان «هیچ‌مقدار» وارد کرد. تعداد داده‌های ورودی باید برابر تعداد ستون‌های جدول باشد و با کاما جداشده باشند و نیز داده‌های ورودی باید به ترتیب با نوع داده ستون‌ها هم‌خوانی داشته باشند. واردکردن داده‌ها با دستور INSERT انجام می‌شود و با دو فرمت مختلف می‌تواند به کار رود:

- واردکردن سطر به سطر که با هر دستور یک سطر به جدول اضافه می‌شود. شکل کلی این دستور چنین است:

```
INSERT INTO table-name
VALUES (list of values)
```

مثال:

```
INSERT INTO book
```

(102, VALUES '3 آموزشی', 'NULL', 'پایگاهداده پیشرفته');

- واردکردن چند سطر که با برداشتن از جدولی دیگر انجام می‌شود. مثلاً فرض کنید جدولی به صورت زیر تعریف شود:

```
CREATE TABLE book_auth
(bno ...
ano ... );
```

حال می‌خواهیم داده‌های این دو ستون را از جدول کتاب برداریم و در این جدول کپی کنیم. این عمل با دستور زیر قابل انجام است:

```
INSERT INTO book_auth
    SELECT bno , ano
```

```
FROM book
WHERE ...;
```

هرچند تا به حال دستور SELECT را معرفی نکرده‌ایم، ولی معنای آن آشکار است. قید WHERE برای شرط به کار می‌رود.

می‌توان چند ستون دلخواه از جدول را با این روش پر کرد. شکل کلی این دستور چنین است:

```
INSERT INTO table-name (coli, colj, ..., coln)
    SELECT ...;
```

یادآوری: در SQL server می‌توان داده‌ها را به صورت دستی نیز وارد کرد یا تغییر داد.

۲. حذف داده‌ها

شکل کلی دستور چنین است:

```
DELETE FROM table-name
    WHERE ...;
```

با این دستور ممکن است هیچ سطری حذف نشود و یا یک یا چند سطر حذف شود.

مثال:

```
DELETE FROM book
    WHERE typ IS NULL;
```

این دستور تمامی اطلاعات مربوط به کتاب‌هایی که برای آنها نوع مشخص نشده‌است را حذف می‌کند.

یک جدول را می‌توان به طور کامل از پایگاه داده حذف کرد. دستور آن چنین است:

```
DROP table-name;
```

بعضی از نسخه‌های SQL از حذف جداولی که تهی نباشد، امتناع می‌ورزند. در این صورت می‌توان ابتدا داده‌های آن را حذف و آن را تهی کرد.

مثال: ستون resum قبل‌اً به جدول پرسنل اضافه شد. به صورت زیر آن را حذف می‌کنیم:

```
CREATE TABLE temp
```

```
(pno smallint,
name nvarchar(20) NOT NULL,
typ nvarchar(10),
dob date,
sex char(1) NOT NULL,
adres nvarchar(100),
PRIMARY KEY (pno));
```

```
/* Copy from personnel */
INSERT INTO temp
SELECT pno, name, typ, dob, sex, adres
FROM personnel;
```

```
/* delete all data from personnel */
DELETE FROM personnel
WHERE pno > 0;
```

```
/* drop personnel */
DROP personnel;
```

```
/* rename temp to personnel */
ALTER TABLE temp
RENAME TO personnel;
```

۳. تغییر داده‌ها

تغییر داده‌ها در جدول با دستور UPDATE انجام می‌شود. شکل کلی آن چنین است:

```
UPDATE table-name
SET attr1 = value1, attr2 = value2, ...
WHERE ...
```

مثال: تبدیل نسخه و نوع کتاب‌های مؤلف با کد ۱ به نسخه ۲ و نوع 'کتاب درسی':
UPDATE book

```
    SET typ = 'کتاب درسی', version = 2
    WHERE ano = 1';
```

طبعی است که ستون‌های دیگر دست نخورده باقی می‌مانند.

در دستور UPDATE برای تغییر ستون‌ها می‌توان به جای چند دستور WHERE ... CASE ... WHEN ... ELSE ... یکباره انجام داد.

شکل عمومی این دستور به صورت زیر است:

```
UPDATE TABLE
SET atr = CASE
```

```

WHEN condition 1 THEN value 1
WHEN condition 2 THEN value 2
...
WHEN condition n THEN value n
ELSE...
END;

```

مثال: اعمال تخفیف در قیمت کتاب

```

UPDATE publish
SET Uprice = CASE
WHEN dop <= 1357 THEN 0
WHEN dop >1357 AND dop < 1370 THEN Uprice*0.5
    WHEN dop >= 1370 THEN Uprice * 0.9
    ELSE Uprice * 0.99 /* NULL Price*/
END;

```

در این مثال قیمت کتاب‌هایی که قبل ۱۳۵۷ منتشر شده‌اند، صفر است، بین سال‌های ۱۳۵۷ تا ۱۳۷۰ نصف قیمت و... و در صورتی که تاریخ چاپ کتاب مشخص نباشد، 0.99 برابر خواهد شد.

توضیح: در SQL همانند زبان C می‌توان توضیحاتی در ابتدا یا انتهای سطر مانند $/*$ در مثال بالا اضافه کرد.

توضیح: در SQL server می‌توان داده‌ها را به صورت دستی نیز اضافه کرد.
تغییراتی که در بالا اشاره شد روی جداول موجود صورت می‌گیرد.
برخی جداول نشر کتاب همراه با داده‌های آن‌ها که در پرسش‌های آینده مورد استفاده قرار می‌گیرند، در شکل ۳-۳ آمده‌اند. در این جداول ستون‌های کلید مستقل از یکدیگر هستند، مثلاً مؤلف شماره ۱ و پرسنل شماره ۲ یک نفر است (مصطفی حق‌جو)، زیرا جدول مؤلف معمولاً خارج از حوزه انتشارات خاصی تنظیم می‌شود:

Personel

pno	name	typ	dob	sex	adres
1	احسان بزرگ‌خواه	تایبیست	1336-01-01	m	تهران
2	رضا نظری	ویراستار	1334-03-01	m	کیش
3	مصطفی حق‌جو	صفحه‌آرا	1334-03-01	m	کیش
4	ابوالذر جباری	ویراستار	1373-05-01	m	کیش
5	محمد رضا قربانی	حکاس	1373-04-25	m	کیش
6	شاطمه رهگذر	تایبیست	1345-01-01	f	NULL
7	شهن حقوقی	منشی	1333-01-01	f	تهران
8	نیما ذارع	طریح	1332-01-01	m	تهران
9	مهدی ذارع	صفحه‌آرا	1360-05-23	m	NULL
10	میثم بازیاری	مدیر	1365-05-05	m	کیش
11	رامین شاهد	مدیر	1364-03-25	m	کیش
12	علی قربانی	مدیر	1336-02-12	m	تهران
13	نوروز تبریزی	مدیر	1345-02-25	m	NULL
14	محمدعلی ابراهیمی	مدیر	1360-07-04	m	تهران
15	علی فتنی	مدیر	1372-04-20	m	کیش

Author

	ano	name	typ	dob	sex	adress	degree	major	resum
1	1	مصطفی حق جو	نویسنده	1334-03-01	m	کیش	دکترا	کامپیوتر	inetemet
2	2	مصطفی حق جو	شاعر	1334-03-01	m	کیش	دکترا	کامپیوتر	inetemet
3	3	زین کوب	مترجم	1373-05-01	m	لرستان	دکترا	ادیبات	NULL
4	5	فروغ فرخزاد	شاعر	1345-01-01	f	NULL	NULL	NULL	inetemet
5	6	مصطفی رحماندوست	شاعر	1333-01-01	m	تهران	NULL	ادیبات	inetemet
6	7	جلال آل احمد	نویسنده	1332-01-01	m	تهران	NULL	NULL	NULL
7	10	جعفر تبا	گردآور	1355-05-01	m	تهران	ارشد	کامپیوتر	NULL
8	11	فاطمه نورانی	گردآور	1351-02-02	f	تهران	ارشد	کامپیوتر	NULL
9	12	احمد فراهی	نویسنده	1336-01-01	m	تهران	دکترا	کامپیوتر	inetemet
10	14	سی فنی	گردآور	1372-04-20	m	کیش	کارشناسی علوم	کارشناسی علوم	NULL

book

	bno	title	dop	typ	version	ano
1	1	پایگاه داده ها	1394-01-01	دانشگاهی	2	1
2	1	پایگاه داده ها	1394-01-01	دانشگاهی	2	12
3	2	برنامه سازی پیشرفته	1380-01-01	دانشگاهی	2	12
4	3	لب و سوزن	1379-01-01	شعر	1	2
5	4	مدیر مدرسه	1337-01-01	رمان	NULL	7
6	5	دو قرن سکوت	NULL	اجتماعی	NULL	3
7	6	ساختمان داده ها	1389-08-01	دانشگاهی	6	10
8	7	ساختمان های تجسته	1391-08-01	دانشگاهی	2	11
9	10	پایگاه داده پیشرفته	NULL	دانشگاهی	2	1
10	11	ابزار فنی	1394-02-10	فنی	1	14

publisher

	pno	title	typ	dope	adres
1	3	اطلاعات	روزنامه	1310-01-01	تهران - پارک شهر
2	9	شعر و شرحی	شعر	1392-02-15	کیش
3	10	بنج شبه ها با هنر	شعر	1392-01-20	کیش
4	12	امیرکبیر	عمومی	1370-10-02	تهران
5	13	بیام نور	درسی	1360-01-01	تهران
6	14	انتشارات فنی	فنی	1390-10-10	کیش

publish

	pubno	pno	bno	dop	Uprice	num
1	1	9	1	1394-01-01	90000.00	5000
2	2	13	2	1380-01-01	500000.00	1000
3	3	13	6	1389-08-01	120000.00	5000
4	4	13	7	1391-08-01	150000.00	3000
5	5	14	11	1394-02-10	140000.00	3000

شکل ۳-۳. نمونه‌هایی از جداول نشر کتاب با داده

۸-۵ استخراج اطلاعات

۱. گزینش و پرتو

شكل کلی این دستور چنین است:

```
SELECT col1,...,coln
FROM tab1,...,tabm
WHERE ...
```

مثال: کتاب‌هایی که نسخه آن‌ها مشخص نیست و تاریخ چاپ آن‌ها قبل از ۱۳۹۰ است.

```
SELECT *
FROM book
WHERE dop < '1390' AND version IS NULL;
```

توضیح: تاریخ همیشه در بین دو علامت، می‌آید.

نتیجه:

	bno	title	dop	typ	version	ano
1	4	مدیر مدرسه	1337-01-01	رهان	NULL	7

مثال: نشرهایی که قیمت آن‌ها بالای ۱۰۰۰۰۰ است.

```
SELECT *
FROM publish
WHERE Uprice > 100000;
```

نتیجه:

	pubno	pno	bno	dop	Uprice	num
1	2	13	2	1380-01-01	500000.00	1000
2	3	13	6	1389-08-01	120000.00	5000
3	4	13	7	1391-08-01	150000.00	3000
4	5	14	11	1394-02-10	140000.00	3000

توضیح: ممکن است SQL به‌طور پیش‌فرض تاپل‌های تکراری را حذف کند! برای اطمینان از حذف نشدن تاپل‌های تکراری می‌توان از ALL استفاده کرد.

مثال: تمام اسمی‌حتی‌اسمی تکراری در جدول مؤلف

```
SELECT ALL name
FROM author;
```

بخش SELECT می‌تواند شامل محاسبات نیز باشد.

مثال:

`SELECT bno , Uprice * 1.1`

`FROM publish;`

می‌توان در مقایسهٔ مقادیر، از قیدهای `BETWEEN` و `NOTBETWEEN` استفاده کرد.

مثال: سطرهایی از جدول انتشار که کتاب با قیمت ۱۰۰۰۰۰ تا ۲۰۰۰۰۰ نشر داده‌اند.

`SELECT *`

`FROM publish`

`WHERE Uprice BETWEEN 100000 AND 200000;`

نتیجه:

	pubno	pno	bno	dop	Uprice	num
1	3	13	6	1389-08-01	120000.00	5000
2	4	13	7	1391-08-01	150000.00	3000
3	5	14	11	1394-02-10	140000.00	3000

۲. پیوند دو یا چند جدول

ممکن است اطلاعات موردنیاز یک پرسش در بیش از یک جدول وجود داشته باشد.

مثال: مؤلفهای کتاب‌هایی که به چاپ دوم (یا بیشتر) رسیده‌اند.

این پرسش نیاز به دو جدول دارد: مشخصات مؤلفها در جدول `author` و نسخه کتاب در جدول `book` آمده‌است. پیوند دو یا چند جدول با آوردن نام آن‌ها در کنار یکدیگر در بخش `FROM` انجام می‌شود ولی نتیجه آن ضرب دکارتی یعنی ترکیب کامل سطر و ستون است. نکته بسیار مهم این است که باید این دو جدول را روی کلید مشترک خارجی آن‌ها (`ano`) پیوند داد (پیوند طبیعی)، یعنی شرط تساوی این صفت در دو جدول را بیان کرد تا ارتباط بین مؤلف و کتاب او برقرار شود. توجه کنید که نقطه‌گذاری فقط برای نامهای تکرار لازم است.

`SELECT author.*`

`FROM author,book`

`WHERE author.ano = book.ano`

`AND version >= 2;`

نتیجه:

	ano	name	typ	dob	sex	adress	degree	major	resum
1	1	مصطفی حق جو	لویستده	1334-03-01	m	کیش	دکترا	کامپیوتر	internet
2	12	احمد فراهی	لویستده	1336-01-01	m	تهران	دکترا	کامپیوتر	internet
3	12	احمد فراهی	لویستده	1336-01-01	m	تهران	دکترا	کامپیوتر	internet
4	10	جعفر تها	گردآور	1355-05-01	m	تهران	ارشد	کامپیوتر	NULL
5	11	فاطمه نورانی	گردآور	1351-02-02	f	تهران	ارشد	کامپیوتر	NULL
6	1	مصطفی حق جو	لویستده	1334-03-01	m	کیش	دکترا	کامپیوتر	internet

توضیح: اگر بخواهیم سطر تکراری نداشته باشیم باید از قید DISTINCT استفاده کنیم.

مثال: نام کتاب‌ها و مؤلف‌هایی که به چاپ دوم (یا بیشتر) رسیده‌اند.
پاسخ غلط:

```
SELECT DISTINCT name, title
FROM author, book
WHERE version >= 2;
```

بخشی از نتیجه:

	name	title
1	احمد فراهی	برنامه سازی پیشرفته
2	احمد فراهی	پایگاه داده پیشرفته
3	احمد فراهی	پایگاه داده ها
4	احمد فراهی	ساختمان داده ها
5	احمد فراهی	ساختمان های حساسه
6	جعفر تها	برنامه سازی پیشرفته

توضیح: چون شرط کلید خارجی رعایت نشده‌است، کتاب‌ها مربوط به مؤلفین نخواهند بود! مثلاً احمد فراهی مؤلف همه کتاب‌های بالا و جعفر تنها نیز مؤلف برنامه‌سازی پیشرفته نیست.

مثال: لیست کتاب‌های ناشری با نام «پیام‌نور»
پاسخ غلط: ظاهراً این پرسش به دو جدول book و publisher نیاز دارد، ولی این دو جدول در یکدیگر کلید خارجی ندارند و پاسخ زیر غلط است.

```
SELECT DISTINCT publisher.title, book.*
FROM book, publisher
```

WHERE publisher.title = «پیام نور»;

پاسخ صحیح: این پرسش به جدول publish نیز نیاز دارد، زیرا دو جدول book و publisher در یکدیگر کلید خارجی ندارند. درواقع این دو جدول را publish به یکدیگر ارتباط می‌دهد.

```
SELECT DISTINCT publisher.title, book.*  
FROM book, publish, publisher  
WHERE book.bno=publish.bno  
AND publish.pno=publisher.pno  
AND publisher.title = «پیام نور»;
```

نتیجه:

	title	bno	title	dop	typ	version	ano
1	پیام نور	2	برنامه سازی پیشرفته	1380-01-01	دانشگاهی	2	12
2	پیام نور	6	ساختمان داده ها	1389-08-01	دانشگاهی	6	10
3	پیام نور	7	ساختمان طایی گسته	1391-08-01	دانشگاهی	2	11

پیوند جداول بهشدت سرعت را کاهش و حافظه مورد نیاز را افزایش می‌دهد. در واقع با پیوند دو یا چند جدول، عملاً ترکیب آنها (که ممکن است جدول بزرگی باشد) در حافظه کامپیوتر تشکیل و سپس اطلاعات مورد نیاز از آن استخراج می‌شود. بنابراین باید حتی الامکان از پیوند جداول پرهیز کرد، مگر آنکه امکان‌پذیر نباشد. بهترین جایگزین پیوند جداول، پرسش تودرتو است.

توضیح: خواهیم دید که پیوند جداول با استفاده از قیدهایی مانند NATURAL JOIN بهتر و سریع‌تر انجام می‌شود.

۳. پرسش و پاسخ تودرتو^۱

می‌توان بسیاری از پرسش‌ها را، با پرهیز از پیوند، توسط پرسش‌های تودرتو پاسخ داد.

مثال: مؤلفهای کتاب پایگاهداده‌ها

```
SELECT * FROM author  
WHERE ano IN (SELECT ano FROM book  
WHERE title = «پایگاهداده‌ها»);
```

به چند نکته در رابطه با این مثال دقت فرمایید:

- پرسش تودرتو با پرانتز انجام می‌شود.
- اگر پرسش درونی و بیرونی مستقل باشند، ابتدا پرسش درونی اجرا و نتیجه آن ذخیره می‌شود و در پرسش بیرونی مورد استفاده قرار می‌گیرد. در بالا ابتدا یک جدول تکستونی ano که با شماره‌های مؤلفین پایگاه‌داده‌ها پر شده و ایجاد می‌شود، سپس مشخصات کامل این افراد از جدول author استخراج می‌شود. اتصال دو پرسش (خارج و داخل پرانتز) باید معنی‌دار باشد. در بالا ستون ano با عملگر IN که وجود مقداری را در یک لیست نشان می‌دهد با لیست مقایسه می‌شود.

مثال: انتشاراتی که کتاب‌هایی در سال ۱۳۹۴ انتشار داده است.

```
SELECT * FROM publisher
WHERE pno IN
      (SELECT pno FROM publish
       WHERE bno IN
             (SELECT bno FROM book
              WHERE dop > '1393' AND dop < '1395'));
```

نتیجه:

	pno	title	typ	dop	adres
1	9	شعر و شرحی	شعر	1392-02-15	کیش
2	14	انتشارات فنی	فنی	1390-10-10	کیش

مثال ترکیبی: کتاب‌هایی که مدیر نشر و نویسنده آنها همان هستند.

پاسخ ۱: پیوند

```
SELECT book.* FROM book, author, publish, personel
WHERE book.ano = author.ano
AND book.bno = publish.bno
AND publish.pno = personel.pno
AND author.name = personel.name;
```

نتیجه:

	bno	title	dop	typ	version	ano
1	11	ابزار فنی	1394-02-10	فنی	1	14

پاسخ ۲: پرسش‌های تودرتو

```
SELECT book.* FROM book WHERE ano IN
      (SELECT ano FROM author WHERE name IN
           (SELECT name FROM personel WHERE pno IN
                (SELECT pno FROM publish)));
```

نتیجه:

	bno	title	dop	typ	version	ano
1	11	ابزار فنی	1394-02-10	فنی	1	14

در این مثال مشاهده می‌شود که هر دو روش، نتیجهٔ یکسان می‌دهند ولی روش پیوند بسیار کندر است و نیز فضای زیادی از حافظه را اشغال می‌کند.

توضیح:

- می‌توان تاپل‌های چندبعدی را با هم مقایسه کرد، ولی همهٔ پیاده‌سازی‌های SQL پشتیبانی نمی‌کنند.

مثال:

```
SELECT * FROM book
```

```
WHERE (bno, ano) IN (SELECT (bno, pno) FROM publish);
```

وجود تاپل‌های تکراری در دستور SQL با قید UNIQUE در زیر پرسش مشخص می‌شود.

مثال: کتاب‌هایی که یک مؤلف بیشتر ندارند.

```
SELECT * FROM book
```

```
WHERE UNIQUE (SELECT ano FROM author
```

```
WHERE book.ano = author.ano);
```

این پرسش کتاب‌هایی را باز می‌گرداند که یک مؤلف بیشتر ندارند، ولی اگر مؤلف کتابی معلوم نباشد، در این صورت هم، گزارهٔ UNIQUE مقدار درست را بر می‌گرداند.

مثال دیگر: کتاب‌هایی که بیش از یک مؤلف دارند.

```
SELECT * FROM book
```

```
WHERE NOT UNIQUE (SELECT ano FROM author
```

```
WHERE book.ano = author.ano);
```

توضیح: در پرسش تودرتو هرگاه دستور WHERE درونی به ستونی از جدول دستور FROM بیرونی وابسته باشد، انجام پرسش تودرتو متفاوت خواهد بود، زیرا پرسش درونی را نمی‌توان به‌طور مستقل انجام داد.

مثال: قیمت کتاب و مشخصات ناشرهایی که کتاب مدیر خود را انتشار داده‌اند.

```
SELECT publisher.*, Uprice FROM publish, publisher
```

```
WHERE publish.pno = publisher.pno
```

```
AND bno IN (SELECT bno FROM book
```

```
WHERE publish.pno = book.ano);
```

نتیجه:

pno	title	typ	dop	adres	Uprice
1	14	فنی	الشورات فنی	کیش	140000.00

۴. مرتب‌کردن خروجی

برای مرتب‌کردن خروجی از دستور ORDER BY استفاده می‌کنیم. این دستور در پایان پرسش می‌آید. در واقع سیستم پس از اتمام کار و گرفتن خروجی، آن را مرتب می‌کند. این ترتیب به‌طور پیش‌فرض صعودی خواهد بود، ولی می‌توان از قید ASC نیز استفاده کرد. می‌توان با افزودن قید DESC آن را نزولی کرد.

مثال: نام و شماره کتاب‌های که از سال ۱۳۸۰ به بعد چاپ شده‌اند، براساس نام کتاب به صورت نزولی.

```
SELECT DISTINCT bno, title, dop FROM book
WHERE dop > '1380'
ORDER BY title DESC;
```

نتیجه:

bno	title	dop
1	ساختمان های تجسته	1391-08-01
2	ساختمان داده ها	1389-08-01
3	پایگاه داده ها	1394-01-01
4	ابزار فنی	1394-02-10

می‌توان خروجی را روی چند کلید مرتب کرد:

مثال: نام و شماره کتاب‌های مختلف به شکل مرتب.

- هر دو صعودی

```
SELECT DISTINCT bno, title
FROM book
ORDER BY bno, title;
```

- نزولی شماره کتاب، صعودی نام کتاب.

```
SELECT DISTINCT bno, title
FROM book
ORDER BY bno DESC, title ASC;
```

۵. عملگرهای مجموعه‌ای

عملگرهای مجموعه، همگی در SQL عیناً پیاده‌سازی شده‌اند. عمل اجتماع با UNION، عمل اشتراک با INTERSECT و عمل تفاضل با دستور EXCEPT

پیاده‌سازی شده است. همتایی داده‌ها باید رعایت شود یعنی در صورتی می‌توان هریک از این عملگرهای را روی دو جدول به کاربرد که تعداد ستون‌ها و نیز دامنه ستون‌ها به ترتیب برابر باشد. چنان‌که قبلاً دیده‌ایم، در SQL عملگر تعلق نیز با نام IN (عدم تعلق با IN NOT) پشتیبانی می‌شود. همچنین عملگر دیگری به نام CONTAINS تعریف شده است که مشابه زیرمجموعه عمل می‌کند (مجموعه دست‌راستی، زیرمجموعه دست‌چپی است).

مثال: مؤلفینی که مترجم یا شاعر هستند.

```
SELECT*
FROM author
WHERE typ IN ('شاعر', 'مترجم');
```

نتیجه:

ano	name	typ	dob	sex	adress	degree	major	resum
1	2	مصطفی حق جو	شاعر	1334-03-01	m	کیش	دکترا	کامپیوتر intemet
2	3	لریز کوب	مترجم	1373-05-01	m	لرستان	دکترا	ادبیات NULL
3	5	غروغ غرخزاد	شاعر	1345-01-01	f	NULL	NULL	intemet
4	6	مصطفی رحماندوست	شاعر	1333-01-01	m	تهران	NULL	ادبیات intemet

مثال: همه مؤلفین به جز نویسندها و شاعرها

```
SELECT *
FROM author
WHERE typ NOT IN ('نویسنده', 'شاعر');
```

مثال: نام و تاریخ تولد مؤلفینی که در لیست کارکنان نیز هستند.

```
SELECT name, dob
FROM author
INTERSECT
SELECT name, dob
FROM personel;
```

نتیجه:

	name	dob
1	علی فنی	1372-04-20
2	مصطفی حق جو	1334-03-01

اگر به جای UNION از INTERSECT استفاده کنیم، آنگاه اسمی همه مؤلفین و کارکنان را می‌دهد و اگر EXCEPT به کار ببریم، اسمی مؤلفین به جز آن‌هایی که با کارکنان هم‌نام هستند، می‌دهد.

توجه: ترتیب فقط در EXCEPT مهم است.

مثال: نام و جنسیت و تاریخ تولد مؤلف‌هایی که کارکنان یکسان ندارند.

```
SELECT name, sex, dob FROM author
EXCEPT
SELECT name, sex, dob FROM personnel;
```

نتیجه:

	name	sex	dob
1	احمد غراطی	m	1336-01-01
2	حصیر تتها	m	1355-05-01
3	جلال آل احمد	m	1332-01-01
4	زرين كوب	m	1373-05-01
5	فاطمه نوراني	f	1351-02-02
6	فروغ فرخزاد	f	1345-01-01
7	مصطفی رحماندوست	m	1333-01-01

باید توجه کرد که معادل بعضی از این پرسش‌ها را به روش‌های دیگر نیز می‌توان نوشت.

```
SELECT author.name
FROM author< personnel
WHERE ano <> pno AND author.name = personnel.name;
```

۶. توابع محاسباتی

توابع محاسباتی عبارت‌اند از .SUM, MAX, AVG, COUNT, MIN در صورت لزوم این توابع را می‌توان با کلمه DISTINCT، وادر کرد تا داده‌های تکراری را در نظر نگیرند (برای MIN, MAX داده‌های تکراری اهمیتی ندارند). مقادیر NULL قبل از اجرای این توابع حذف می‌شود و تأثیری روی آن‌ها ندارد. مثلاً در تابع AVG مقادیر NULL در مخرج کسر (تعداد داده‌ها) اثری ندارد.

مثال: تعداد کتاب‌های دانشگاهی.

```
SELECT COUNT(bno)
FROM book
WHERE typ = «دانشگاهی»;
```

نتیجه:

	(No column name)
1	6

توضیح: این نتیجه درست نیست، زیرا شماره کتاب‌های تکراری (کتاب‌هایی با چند مؤلف) را چندبار شمرده است. می‌توان با قید DISTINCT از آن جلوگیری کرد:

`SELECT COUNT(DISTINCT bno)`

`FROM book`

`WHERE typ = «دانشگاهی»;`

نتیجه:

(No column name)	
1	5

تابع COUNT(*) برای شمارش سطرهای جداول تعريف شده است. در این تابع نمی‌توان از DISTINCT استفاده کرد. اگر جدول تهی باشد، مقدار صفر باز می‌گردد.

مثال: تعداد سطرهای جدول کتاب

`SELECT COUNT(*)`

`FROM book;`

نتیجه:

(No column name)	
1	10

مثال: کتاب‌هایی که هیچ‌گاه کمتر از ۲۰۰۰ نسخه چاپ نشده‌اند.

پاسخ غلط:

`SELECT * FROM book`

`WHERE bno IN (SELECT bno FROM publish`

`WHERE num > 2000);`

این پاسخ غلط است، زیرا ممکن است کتابی یک‌بار بالای ۲۰۰۰ و یک‌بار دیگر کمتر از ۲۰۰۰ نسخه چاپ شده باشد. چنین کتابی در خروجی می‌آید.

پاسخ صحیح:

`SELECT * FROM book`

`WHERE bno NOT IN (SELECT bno FROM publish`

`WHERE num < 2000);`

مثال: کتاب‌هایی که تیراز چاپ شده آن‌ها کمینه بوده است.

پاسخ:

`SELECT * FROM book`

`WHERE bno IN (SELECT bno FROM publish`

`WHERE num = (SELECT MIN(num) FROM publish));`

نتیجه:

bno	title	dop	typ	version	ano
1	2	برنامه سازی پیشرفته	1380-01-01	دانشگاهی	2

توضیح: پاسخی مانند آنچه در زیر آمده، غلط است:

```
SELECT * FROM book
WHERE num=(SELECT MIN(num) FROM publish);
```

دلیل آن این است که داده‌ها در جدول هیچ ترتیبی ندارند و اگر مقدار کمینه در سطر `1` جدول باشد، خروجی‌های قبل از آن غلط خواهد بود. پرسش تودرتو در بهکارگیری توابع محاسباتی نقش اساسی دارد.

مثال: تعداد کل و میانگین کتاب‌های چاپ شده توسط انتشارات پیام‌نور.

پاسخ:

```
SELECT SUM(num) AS PayamSum, AVG(num) AS
PayamAvg
FROM publish
WHERE pno IN (SELECT pno FROM publisher
WHERE title = 'پیام‌نور');
```

نتیجه:

	PayamSum	PayamAvg
1	9000	3000

توضیح: خروجی‌ها را می‌توان با قید AS نامگذاری کرد.

۷. گروه‌بندی

گاهی می‌خواهیم داده‌ها را گروه‌بندی یا دسته‌بندی کنیم، یعنی می‌خواهیم خروجی را به تفکیک داشته باشیم. به مثال‌های زیر دقت کنید:

۱. تعداد کارکنان به تفکیک زن و مرد

۲. تعداد مؤلفین به تفکیک زن و مرد و مدرک تحصیلی

۳. گرانترین و ارزان‌ترین کتاب‌های هر ناشر

در هریک از این مثال‌ها لازم است شمارش در هر گروه به‌طور جداگانه انجام شود (مثلاً تعداد کارکنان زن جداگانه و تعداد کارکنان مرد نیز جداگانه شمارش شود).

گروه‌بندی در SQL با دستور GROUP BY انجام می‌شود. نکات زیر را باید رعایت کرد:

- تمام ستون‌هایی که گروه‌بندی روی آنها انجام می‌شود، حتماً باید در خروجی (دستور SELECT) آورده شوند.
- اگر به غیر از صفات گروه‌بندی شده، صفات دیگری در خروجی باشند، باید توسط یکی از توابع محاسباتی همراهی شوند.
- بخش ORDER BY جزء آخرین بخش‌های یک دستور است. فقط می‌تواند بعد از آن بیاید.

مثال: تعداد کارکنان به تفکیک زن و مرد:

```
SELECT sex, count(pno) AS NumOfPersonnel
FROM personnel
GROUP BY sex;
```

نتیجه:

	sex	NumOfPersonnel
1	f	2
2	m	13

مثال: تعداد مؤلف‌ها به تفکیک جنسیت و آدرس

```
SELECT sex, address, count(ano) AS NumOfWriters
FROM author
GROUP BY sex, address;
```

نتیجه:

	sex	address	NumOfWriters
1	f	NULL	1
2	f	تهران	1
3	m	تهران	4
4	m	برستان	1
5	m	کیش	3

در خروجی این دستور ابتدا داده‌ها به تفکیک زن و مرد و سپس در هر گروه زن و مرد به تفکیک آدرس آمده است.

مثال: قیمت ارزان‌ترین کتاب هر انتشارات.

پاسخ:

```
SELECT pno, MIN(Uprice) AS MinPrice
```

```
FROM publish
GROUP BY pno;
```

نتیجه:

	pno	MinPrice
1	9	90000.00
2	13	120000.00
3	14	140000.00

شرطِ درون گروهی با قید HAVING
 مثال: تعداد کارکنان به تفکیک زن و مرد و نوع در صورتی که بیش از پنج نفر باشند.

توضیح: شرط بیش از پنج نفر را نمی‌توان در دستور WHERE آورد، زیرا باید بعد از شمارش، در درون هر گروه اعمال شود. این کار در SQL با قید HAVING انجام می‌شود که بعد از GROUP BY می‌آید و شبیه WHERE عمل می‌کند.

```
SELECT sex, typ, count(DISTINCT pno) AS NumOfPersonnel
FROM personel
GROUP BY sex, typ
HAVING COUNT(DISTINCT pno)> 5;
```

نتیجه:

	sex	typ	NumOfPersonnel
1	m	مدیر	6

مثال: کلید مدیران نشر که بعد از سال ۱۳۹۰ بیش از یک نوع انتشارات تأسیس کرده‌اند.

```
SELECT publisher.pno, COUNT(publisher.typ) AS NumOfTyp
FROM publisher
WHERE publisher.doe> '1390'
GROUP BY publisher.pno
HAVING COUNT(publisher.typ)> 1;
```

نتیجه: خروجی این پرسشن یک جدول تهی است، به صورت زیر:

pno	NumOfTyp

۸. خروجی به صورت زیر دستور در SQL می‌توان با استفاده از زیر دستور، بخشی از خروجی یک دستور را تولید کرد.

شرط صحت این است که آن زیر دستور فقط یک مقدار را برگرداند. در غیراین صورت خطای زمان اجرا^۱ اتفاق خواهد افتاد.

مثال: نام هر ناشر و تنوع کتاب‌هایی که منتشر کرده است.

```
SELECT p.title, (SELECT COUNT(book.bno)
                  FROM book, publish
                 WHERE p.pno = publish.pno
                   AND publish.bno = book.bno
                ) AS NumOfBook
  FROM publisher AS p;
```

نتیجه:

	title	NumOf Books
1	اطلاعات	0
2	شعر و شعری	2
3	ینچ شنبه ها با هنر	0
4	امیرکبیر	0
5	سیام نور	3
6	انتشارات فشی	1

۹. قید LIKE

برای کار با رشته‌ها^۲ در SQL از عملگر LIKE استفاده می‌شود. با این عملگر ارزشمند می‌توان رشته‌ها را مقایسه و بررسی کرد.

دو کاراکتر ویژه (wildcard) برای این‌منظور در نظر گرفته شده است:

- کاراکتر «_» (underscore) یعنی «به جای یک کاراکتر»

- کاراکتر «٪» یعنی «به جای هر تعداد کاراکتر»

مثال: مؤلف‌هایی که نام آن‌ها «مصطفی» است.

```
SELECT *
  FROM author
 WHERE name LIKE '%مصطفی%';
```

نتیجه:

	ano	name	typ	dob	sex	adress	degree	major	resum
1	1	مصطفی حق جو	نویسنده	1334-03-01	m	کیش	دکترا	کامپیوتر	inetmet
2	2	مصطفی حق جو	شاعر	1334-03-01	m	کیش	دکترا	کامپیوتر	inetmet
3	6	مصطفی رحمندوست	شاعر	1333-01-01	m	تهران	NULL	ادبیات	inetmet

مثال: نشرهایی که در نام یا نوع آن‌ها کلمه شعر آمده است.

```
SELECT * FROM publisher
WHERE title LIKE '%شعر%' OR typ LIKE '%شعر%';
```

نتیجه:

	pno	title	typ	doe	adres
1	9	شعر و شعری	شعر	1392-02-15	کیش
2	10	پنج شنبه ها با هنر	شعر	1392-01-20	کیش

مثال: «نویسنده»‌هایی که در نام آنها دوبار، «عبدالله» آمده‌است و با یک فاصله جدا شده‌است (مانند «عبدالله عبدالله»).

```
SELECT *
FROM author
WHERE name LIKE '% عبدالله %';
```

مثال: مؤلف‌هایی که در رزومه آنها کد 'OO' به هر شکلی آمده‌است (به معنی (Object Oriented

حل: به دو نکته باید توجه کرد:

- ممکن است این کدها به صورت OO (چسبیده) یا O_O یا O-O یا O در داده‌ها آمده باشد.

- ممکن است از حروف کوچک یا بزرگ یا هر دو استفاده شده باشد.
برای حل مشکل حروف بزرگ و کوچک، بسیاری از نسخه‌های SQL قیدهای LOWER و UPPER تعریف کرده‌اند که ابتدا داده را به حروف بزرگ یا کوچک تبدیل و سپس مقایسه می‌کنند.

```
SELECT *
FROM author
WHERE UPPER(resum) LIKE '%OO%'
OR UPPER(resum) LIKE '%O_O%';
```

توضیح:

- بعضی از نسخه‌های SQL بین حروف بزرگ و کوچک در داده‌ها تفاوت قائل نمی‌شوند. مثلاً عبارت 'TEHRAN'='Tehran' برابر true ارزیابی می‌شود. در این نسخه‌ها توابع و عملگرهایی برای رشته‌ها تعریف شده‌است مانند || برای پیوند رشته‌ها و trim(S) برای حذف فاصله‌ها از آخر رشته.

- علاوه بر عملگر LIKE، عملگر NOT LIKE هم داریم.

۳. برای لغو اثر دو کاراکتر ویژه از % _ از \ استفاده می‌شود.

مثال: اسمای کتاب‌هایی که با ۱۰۰% تمام نمی‌شود.

```
SELECT title
FROM book
WHERE title NOT LIKE '%100\%';
```

۱۰. قید NULL

مقدار NULL از قوانین زیر پیروی می‌کند:

۱. فقط با عملگرهای NOT NULL و IS NULL می‌توان با آن کار کرد. بنابراین دستورهایی مانند WHERE name = NULL و عملیات جبری روی NULL غلط است.

۲. ممکن است از کلمه NULL استفاده نکنیم، ولی اگر مقدار داده‌ای که در دستور مقایسه یا جبری می‌آید NULL باشد. در این صورت:

- نتیجه مقایسه UNKNOWN است (نه TRUE و نه FALSE).

- نتیجه عملیات جبری همواره NULL است.

برای شناخت مقدار UNKNOWN، اگر FALSE=0، TRUE=1 UNKNOWN باشد، آنگاه قوانین زیر صادق است: UNKNOWN=0.5

- عملگر AND روی TRUE و FALSE مقدار کمینه را برمی‌گرداند. مثلاً UNKNOWN برابر TRUE AND UNKNOWN خواهد شد.

- عملگر OR مقدار بیشینه را برمی‌گرداند.

نتیجه عملگر NOT برابر مقدار ۱- است. مثلاً NOT UNKNOWN برابر ۰.۵ برابر است. مثلاً UNKNOWN برابر ۰.۵ یعنی UNKNOWN خواهد شد.

۱۱. قید EXISTS

در ادامه عملگرهای NOT NULL و IS NULL به عملگر EXISTS که مشابه NOT EXISTS و NOT NULL است. مثلاً NOT EXISTS که مشابه IS NULL عمل می‌کنند، اشاره کرد. این دو عملگر وجود یا عدم وجود یک سطر کامل را بررسی می‌کنند. مثلاً مؤلف‌هایی که کتابی برای آنها ثبت نشده است.

```
SELECT*
FROM author
WHERE NOT EXISTS (SELECT bno FROM book
WHERE author.ano = book.ano);
```

نتیجه:

	ano	name	typ	dob	sex	adress	degree	major	resum
1	5	فروغ فرجزاد	شاعر	1345-01-01	f	NULL	NULL	NULL	internet
2	6	مصطفی رحمندوست	شاعر	1333-01-01	m	تهران	NULL	ادبیات	internet

۱۲. قیدهای ALL و ANY

عملگر ALL برای مقایسه «همه مقادیر» و عملگر ANY (در بعضی از نسخه‌های SQL با نام SOME) برای «هریک از مقادیر» تعریف شده‌اند. نتیجه هر دو عملگر (درست) یا «نادرست» (T یا F) است. این دو عملگر را می‌توان با توابع و عملگرهای دیگر معادل‌سازی کرد.

مثال: کتابی که تیراژ آن از همه بیشتر است.

```
SELECT DISTINCT book.title, num
FROM book, publish
WHERE book.bno=publish.bno
AND num>= ALL (SELECT num FROM publish);
```

نتیجه:

	title	num
1	پایگاه داده ها	5000
2	ساختمان داده ها	5000

مثال: پرسنلی که نام آن‌ها در جدول مؤلف آمده‌است.

```
SELECT *
FROM personnel
WHERE name=ANY (SELECT name FROM author);
```

نتیجه:

	pno	name	typ	dob	sex	adres
1	2	مصطفی حق جو	صفحه آرا	1334-03-01	m	کیش
2	14	علی فلی	مدیر	1372-04-20	m	کیش

۱۳. اسمی معادل

قبل‌آمدیدیم که در SQL می‌توان برای جداول نام دوم یا کنیه تعریف کرد. حوزه عملکرد آن، همان دستور مربوطه است، یعنی پس از آن دستور، نام دوم وجود ندارد. کاربرد آن زمانی است که بخواهیم از یک جدول در یک دستور بیش از یک بار استفاده کنیم و یا جداول دارای اسمی طولانی و صفات مشترک باشند و بخواهیم از تکرار نام‌های طولانی بپرهیزیم. شکل کلی آن چنین است.

```
SELECT ....
FROM table_name AS alias_name, ...
```

با استفاده از دستور with می‌توان یک رابطهٔ موقت دارای نام ایجاد و سپس بلافارصله در ادامه پرسش از آن استفاده کرد.

مثال: کتاب‌هایی که بعد از سال ۱۳۹۱ منتشر شده و قیمت آن‌ها بیش از بیشینه قیمت کتاب‌های قبل از سال ۱۳۹۰ است.

```
WITH max_Uprice(maxp)
(SELECT MAX(Uprice) FROM publish
WHERE dop <'1390')
/* now use the result */
SELECT * FROM publish AS p
WHERE p.dop >'1391'
AND p.Uprice >max_Uprice.maxp;
```

۱۴. قید «همه»

معادل‌سازی عملگر همه در SQL به چند روش امکان‌پذیر است. ساده‌ترین این روش‌ها استفاده از توابع COUNT و CONTAINS است.

مثال: کلید مدیر نشرهایی که همه کتاب‌های دانشگاهی را منتشر کرده‌اند.

حل ۱: تعداد کتاب‌های دانشگاهی منتشرشده توسط هر ناشر را می‌شماریم و با تعداد کل کتاب‌های دانشگاهی مقایسه می‌کنیم:

```
SELECT pno
FROM book, publish
WHERE book.bno=publish.bno
AND typ = 'دانشگاهی'
GROUP BY pno
HAVING COUNT (publish.bno) =
(SELECT COUNT(bno)
FROM book
WHERE typ = 'دانشگاهی');
```

نتیجه: چنین مدیر نشری در داده‌ها وجود ندارد.

حل ۲. (با استفاده از CONTAINS): مجموعه کل کتاب‌های درسی باید زیرمجموعه کتاب‌های درسی منتشرشده توسط آن ناشر باشد، زیرا هر ناشر کتاب‌های دیگری هم منتشر می‌کند. قید CONTAINS در همه نسخه‌های SQL پشتیبانی نمی‌شود.

۱۵. انواع پیوند در SQL

عملیات پیوند در SQL نسل اول با قید ویژه‌ای پشتیبانی نمی‌شود. چنان‌که دیدیم باید با استفاده از شرط تساوی (در WHERE) یا زیر دستور، آن را شبیه‌سازی کرد.

در SQL2، این کمبود مرتفع شده و دستوراتی اضافه شده که انواع پیوند‌ها را پشتیبانی می‌کنند. این دستورات به قرار زیر هستند [حق‌جو، ۱۳۹۲]:

- ضرب دکارتی یعنی ترکیب کامل سطر و ستون دو جدول با دستور CROSS JOIN پیاده‌سازی شده است.

مثال:

`SELECT * FROM publisher CROSS JOIN publish;`

نتیجه: سطرهای غلط در خروجی این دستور ظاهر می‌شوند، یعنی سطرهای ناشر و نشرهایی که هیچ ارتباطی به یکدیگر ندارند (مثلاً ناشری به نام اطلاعات، کتاب پایگاه‌داده را چاپ کرده است!). چند سطر از نتیجه در زیر آمده است:

pno	title	typ	doe	adres	pubno	pno	bno	dop	Uprice	num
1	۳ اطلاعات	روزنامه	۱۳۱۰-۰۱-۰۱	تهران - بارک شهر	۱	۹	۱	۱۳۹۴-۰۱-۰۱	۹۰۰۰۰.۰۰	۵۰۰۰
2	۹ شعر و شرحی	شعر	۱۳۹۲-۰۲-۱۵	کیش	۱	۹	۱	۱۳۹۴-۰۱-۰۱	۹۰۰۰۰.۰۰	۵۰۰۰
3	۱۰ بخش شنبه ها با هم	شعر	۱۳۹۲-۰۱-۲۰	کیش	۱	۹	۱	۱۳۹۴-۰۱-۰۱	۹۰۰۰۰.۰۰	۵۰۰۰
4	۱۲ امیرکبیر	علومی	۱۳۷۰-۱۰-۰۲	تهران	۱	۹	۱	۱۳۹۴-۰۱-۰۱	۹۰۰۰۰.۰۰	۵۰۰۰
5	۱۳ بیام نور	درسی	۱۳۶۰-۰۱-۰۱	تهران	۱	۹	۱	۱۳۹۴-۰۱-۰۱	۹۰۰۰۰.۰۰	۵۰۰۰
6	۱۴ الشارات فلسفی	فلسفی	۱۳۹۰-۱۰-۱۰	کیش	۱	۹	۱	۱۳۹۴-۰۱-۰۱	۹۰۰۰۰.۰۰	۵۰۰۰
7	۳ اطلاعات	روزنامه	۱۳۱۰-۰۱-۰۱	تهران - بارک شهر	۲	۱۳	۲	۱۳۸۰-۰۱-۰۱	۵۰۰۰۰۰.۰۰	۱۰۰۰
۸	۹ شعر و شرحی	شعر	۱۳۹۹-۰۲-۱۵	کیش	۹	۱۲	۲	۱۳۹۸-۰۱-۰۱	۵۰۰۰۰۰.۰۰	۱۰۰۰

یادآوری: ترکیب کامل سطر و ستون جدول، با اینکه اطلاعات غلط در آن درج می‌شود اهمیت زیادی دارد، زیرا بعضی از پرسش‌ها را فقط به وسیله آن می‌توان پاسخ داد.

مثال: لیست ناشر و کتاب‌هایی که توسط آن ناشر چاپ نشده است. باید توجه کرد که این پرسش را فقط با استفاده از EXCEPT یا معادل آن می‌توان نوشت.

`SELECT * FROM book CROSS JOIN publish`

`EXCEPT`

`SELECT * FROM book , publish`

`WHERE book.bno=publish.bno;`

- پیوند طبیعی که با شرط تساوی روی کلید خارجی است، با دستور NATURAL JOIN پیاده‌سازی شده است.

مثال: اطلاعات کامل ناشرین و نشرهای آنها

`SELECT * FROM publisher NATURAL JOIN publish;`

در پیوند طبیعی سه یا چند جدول باید عملگرهای پیوند مانند NATURAL JOIN را تکرار کرد.

مثال: نام کتاب و نام انتشارات مربوطه

پاسخ غلط:

```
SELECT book.title, publisher.title
FROM book NATURAL JOIN publish, publisher;
```

پاسخ صحیح:

```
SELECT book.title, publisher.title
FROM book NATURAL JOIN publish
NATURAL JOIN publisher;
```

- پیوند شرطی (theta join) یعنی ترکیب جداول با هر شرط دلخواه توسط دستور JOIN ... ON ... پیاده‌سازی شده است.

مثال:

```
SELECT * FROM book JOIN publish ON
Pno=ano AND Uprice < 30000;
```

أنواع دیگر پیوند نیز به صورت زیر وجود دارد:

- دستور NATURAL JOIN معادل INNER JOIN است.
- نوع جدیدی از پیوند به نام OUTER JOIN پیاده‌سازی شده که شکل تازه‌ای از پیوند طبیعی و شرطی است. می‌دانیم که مثلاً اگر سطری از جدول book با هیچ سطری از جدول publish کلید خارجی مشترک نداشته باشد، آن سطر در bookNATURAL JOIN publish بعضاً ایجاد اشکال می‌کند. مثلاً اگر دید کاربر book را از روی book NATURAL JOIN publish بسازیم، انتظار داریم معادل همان رابطه book باشد، که نیست! برای رفع این مشکل، دستورات زیر در SQL2 موجود است:

- دستور FULL OUTER JOIN که سطرهای هر دو جدول را که با جدول دیگر مقدار همنام مشترک ندارند، نیز در خروجی می‌آورد.

مثال: دستور زیر تمام کتاب‌ها و انتشارات، حتی کتاب‌هایی که توسط انتشارات شناخته شده‌ای چاپ نشده‌اند (مثل کتاب‌های بدون شابک) و انتشاراتی که کتابی چاپ نکرده‌اند (مثل انتشارات تازه تأسیس) را نیز شامل می‌شود:

```
SELECT * FROM book FULL OUTER JOIN publish;
```

- دستور LEFT OUTER JOIN که مشابه دستور فوق را برای سطرهای جدول سمت چپ، که در دیگری مقدار همنام مشترک ندارند، عمل می‌کند.

مثال: دستور زیر کتاب‌هایی که انتشاراتی برای آن‌ها تعریف‌نشده را نیز می‌دهد:

```
SELECT * FROM book LEFT OUTER JOIN publish;
```

- دستور RIGHT OUTER JOIN که کاربرد آن مشخص است.
- دستورهای بالا از پیوند طبیعی استفاده می‌کنند. می‌توان به جای آن از شرط ON استفاده کرد (پیوند شرطی).

مثال:

```
SELECT * FROM book LEFT OUTER JOIN publish ON  
book.bno = publish.bno AND Uprice < 30000;
```

۱۶. دید کاربران و امنیت^۱

هریک از کاربران به بخش‌هایی از داده‌ها کار دارند. بنابراین دستیابی آن‌ها به بخش‌های دیگر، هم کار را مشکل و هم امنیت را تهدید می‌کند. به این دلیل هر کاربر فقط دید خود را دارد، یعنی تنها به آنچه نیاز دارد دسترسی می‌یابد و از مابقی پایگاه‌داده بی‌خبر است. تنها مدیر پایگاه‌داده و برخی از برنامه‌سازان می‌دانند در کل بانک چه می‌گذرد. دید کاربران، جداول مجازی هستند که لزوماً وجود خارجی و فیزیکی ندارند، ولی می‌توان به آن‌ها دسترسی داشت و در موارد محدودی آن‌ها را تغییر داد. این نوع جدول‌ها در این بخش بررسی می‌شوند.

جدول مجازی در SQL با دستور CREATE VIEW تعریف می‌شود. شکل کلی آن چنین است:

```
CREATE VIEW view_name [(attributes)] AS SELECT ...
```

مثال: جدول مجازی BookInfo شامل نام کتاب، نام مؤلف و تاریخ نشر.

```
CREATE VIEW BookInfo AS  
    SELECT book.title, author.name, dop  
    FROM bookNATURAL JOIN author;
```

نتیجه: در جدول BookInfo ذخیره شده است. با پرسش زیر می‌توان آن را نمایش داد:

```
SELECT * FROM BookInfo;
```

می‌توان ستون‌های جدول مجازی را با نام‌های جدیدی نام‌گذاری کرد. در این صورت به جای [attributes]، نام‌های جدید را می‌گذاریم. مثلاً اگر دستور فوق را

به صورت زیر بنویسیم:

CREATE VIEW BookInfo (BookName, AuthorName, Date) AS
SELECT book.title, author.name, dop FROM book NATURAL JOIN author;

نتیجه:

	BookName	AuthorName	Date
1	پایگاه داده ها	مصطفی حق جو	1394-01-01
2	پایگاه داده ها	احمد فراهی	1394-01-01
3	برنامه سازی پیشرفته	احمد فراهی	1380-01-01
4	لب و سوزن	مصطفی حق جو	1379-01-01
5	مدیر مدرسه	جلال آل احمد	1337-01-01
6	دو قرن سکوت	زرین کوب	NULL
7	ساختمان داده ها	جعفر تتها	1389-08-01
8	ساختمان های نگسته	فاطمه تورانی	1391-08-01
9	پایگاه داده پیشرفته	مصطفی حق جو	NULL
10	ابزار فنی	علی فنی	1394-02-10

جداوی مجازی، تصویری از جدواول حقیقی هستند، یعنی توسط سیستم به جدواول اصلی مرتبط می‌شوند و معمولاً وجود خارجی ندارند. دسترسی به آن‌ها از دید کاربر، مستقیم است، ولی از دید سیستم، غیرمستقیم است، یعنی سیستم هرگونه استخراج اطلاعات را از جدواول اصلی انجام می‌دهد. با این همه، کاربر می‌تواند مستقیماً پرس‌وجوی خود را روی جدول مجازی بنویسد.

جداوی مجازی را می‌توان همانند جدواول اصلی، در پرس‌وجوهای مختلف چون پیوند، پرتو و گزینش به کاربرد. در این ارتباط، تفاوتی بین جدواول اصلی و مجازی نیست. همچنین می‌توان جدواول مجازی را با دستور DROP VIEW نابود کرد.

بعضی از نسخه‌های SQL امکان ذخیره‌سازی دید را فراهم می‌کند (Materialized View). در این صورت دیدهای ذخیره‌شده همراه با تغییر داده‌ها به روز نگه داشته می‌شوند. دستورهایی هم برای درخواست ذخیره‌کردن دید مورد نظر، وجود دارد ولی فرم استانداردی از این دستورها ارائه نشده است.

در SQL 2، هر کاربر شناسه ویژه‌ای دارد. شناسه‌ها را می‌توان در رابطه با دستیابی به اطلاعات محدود کرد یا امتیاز داد. در مجموع، نوع امتیاز برای دستیابی به جدواول در نظر گرفته شده که عبارت‌اند از:

SELECT, INSERT, DELETE, UPDATE, REFERENCES USAGE

مثلاً اگر کاربری بخواهد جدولی را به روز درآورد، باید شناسهٔ او از امتیاز UPDATE روی آن جدول برخوردار باشد. امتیاز REFERENCES برای کنترل و اعمال محدودیت‌های جامعیتی است: کاربر نمی‌تواند محدودیت خاصی را کنترل یا اعمال کند مگر آنکه از امتیاز REFERENCES برخوردار باشد. مثلاً در یک سازمان دولتی، رئیس سازمان باید بتواند روی دستمزد کارمندان خود محدودیت بگذارد، ولی سایرین چنین حقیقی ندارند. امتیاز USAGE برای استفاده از امکاناتی از قبیل VIEW در نظر گرفته شده است. قابل ذکر است که سه امتیاز INSERT، REFERENCES و UPDATE می‌توانند برای بخشی از جدول داده شوند (در این صورت دارای پارامتر خواهند بود).

دادن (اعطاء کردن) امتیاز به کاربر در SQL با دستور GRANT انجام می‌شود.

شکل کلی آن چنین است:

بخشی از پایگاه‌داده <ON> لیست امتیازها <GRANT>
 TO [لیستی از کاربران] WITH GRANT OPTION];
 اگر قید WITH GRANT OPTION ذکر شود، معنای آن این است که کاربر می‌تواند این امتیازها را به کاربران دیگر نیز واگذار کند (با دستور GRANT دیگری).
 مثال:

```
GRANT SELECT, UPDATE ON book, publish, author
TO Haghjoo, Faraahi;
GRANT SELECT ON pnu_db
TO Ghorbani WITH GRANT OPTION;
GRANT UPDATE (degree, major, resum) ON author
TO Jabbari;
```

امتیازها را می‌توان با دستور REVOKE بازپس گرفت (لغو کرد). شکل کلی آن چنین است:

بخشی از پایگاه‌داده <ON> لیست امتیازها <REVOKE>
 FROM [لیست کاربران] ;

این دستور می‌تواند با قیدهای RESTRICT یا CASCADE همراه باشد. در صورت اول، امتیازات واگذارشده به غیر، از طریق کاربر مورد نظر نیز بازپس گرفته می‌شود. همچنین می‌توان فقط حق واگذاری امتیاز به غیر را با دستور REVOKE GRANT OPTION ON

بازپس گرفت، ولی امتیاز همچنان به قوت خود باقی است ولی از این به بعد نمی‌تواند به دیگری واگذار کند.

مثال:

```
REVOKE INSERT ON book, publish FROM Haghjoo,
Farahi;
REVOKE GRANT OPTION ON pnu_db FROM Ghorbani;
```

مشکل اصلی در جداول مجازی، بهروز درآوردن آن‌ها است. در صورتی می‌توان جدول مجازی را بهروز درآورد، که اعمال تغییرات مورد نظر روی جداول اصلی بدون اشکال و ابهام باشد. برای روشن شدن مطلب مثالی می‌زنیم.

مثال: کتاب جدیدی با نام «روش تحقیق» به نویسنده‌گی «هومن نظری» و تاریخ چاپ ۱۳۹۴/۲/۵ به جدول مجازی BookInfo اضافه کنید.

حل: این درخواست، قابل انجام نیست زیرا ستون bno که کلید اصلی جدول book است، مشخص نشده‌است (و نمی‌تواند مشخص شود، زیرا ربطی به جدول مجازی BookInfo ندارد). یادآوری می‌شود که برای کلید جدول نمی‌توان مقدار NULL در نظر گرفت.

بیان تمام شرایطی که بهروز درآوردن جدول مجازی را مجاز بداند، سخت است. بعضی از نسخه‌های SQL این عمل را به طور کلی غیرمجاز می‌دانند. در بعضی دیگر، اختیار با سیستم است تا در صورت امکان انجام دهد. در SQL2 قواعد مشخصی برای بهروز درآوردن جداول مجازی وجود دارد که مهم‌ترین آن‌ها عبارت‌اند از:

- جدول مجازی فقط از یک جدول اصلی ساخته شده باشد.
- در ساختن جدول مجازی از SELECT DISTINCT استفاده نشده باشد.
- اگر جدول مجازی روی جدول اصلی r ساخته شده و r در زیردستور نیامده باشد. این گونه جداول مجازی را «قابل تغییر^۱» می‌نامند.

۱۷. تراکنش

در نسخه‌های جدید SQL مفهوم تراکنش ACID به خوبی پشتیبانی می‌شود. چند دستور SQL که با BEGINATOMIC شروع می‌شود و با END پایان می‌یابد، یک تراکنش محسوب می‌شود (در حالت عادی هر دستور به تنها یک تراکنش است).

تراکنش در SQL در یکی از حالت‌های ثبت^۱ یا لغو^۲ پایان می‌یابد. در اولی همه تغییرات روی داده‌ها دائمی و در دومی ملغی می‌شود، یعنی وضعیت پایگاهداده، به‌حالتی که قبل از شروع تراکنش داشته، باز می‌گردد. در مواردی سیستم به‌طور خودکار پس از لغو تراکنش دستور شروع مجدد^۳ را اجرا می‌کند. ارزش مفهوم تراکنش این است که حالت نیمه‌کاره به‌وجود نمی‌آید، یعنی یک کار یا به‌طور کامل انجام می‌گیرد یا به‌طور کامل لغو (ختنی) می‌شود. علاوه‌بر این می‌توان اعمال قید جامعیت را تا پایان تراکنش با قید DIFFERED به تعویق انداخت. مثلاً یک تراکنش اجازه می‌دهد که ابتدا کتابی را در جدول book وارد کنیم و سپس نویسنده آن را در جدول author وارد کنیم و فقط در پایان تراکنش قید جامعیت «هر کتاب باید مؤلفی در جدول مؤلف داشته باشد» برقرار است. بدون استفاده از قید DIFFERED از واردکردن داده جلوگیری می‌شود.

مثال: اگر مؤلف ۱ کتاب ۱۱ را نوشته باشد، آنگاه تراکنش زیر پذیرفته می‌شود، ولی اگر در درون دو دستور BEGIN ATOMIC...END قرار نگیرد، اجازه ورود سطر اول داده نخواهد شد.

```
BEGIN ATOMIC
    INSERT INTO book VALUES (11, ...., 1);
    INSERT INTO author VALUES (1, ....);
END
```

۱۸. تعریف شاخص

پرونده‌های شاخص‌دار^۴، پرونده‌هایی هستند که علاوه‌بر داده‌های اصلی دارای بخشی به نام شاخص هستند که در واقع پرونده را به چندین ناحیه کوچک‌تر تقسیم می‌کند و بازیابی اطلاعات را سرعت می‌بخشد. پرونده‌های شاخص‌دار انواع و اقسام دارند و می‌توانند چند شاخص داشته باشند. در زبان SQL می‌توان برای تمام ستون‌های جدول شاخص تعریف کرد. دستور کلی این است:

```
CREATE [UNIQUE] INDEX idx_name
ON tab_name (attr);
```

-
1. Commit Work
 2. Rollback Work
 3. Restart
 4. Indexed

قید UNIQUE دلخواه است. idx_name نام شاخصی است که روی جدول tab_name برای صفت attr تعریف می‌شود. اگر از قید UNIQUE استفاده شود، صفت مورد نظر نباید مقادیر تکراری داشته باشد و در بهروز درآوردن جدول نمی‌توانیم مقادیر تکراری برای آن صفت وارد کنیم.

مثال: تعریف شاخص برای book

`CREATE INDEX book_index ON book(bno);`

تعریف شاخص روی جداول یک رهآورد بزرگ و ارزشمند و دو زیان کوچک دارد. رهآورد آن این است که اگر تعداد داده‌ها قابل توجه باشد، سرعت عملیات را به طور معجزه‌آسایی بالا می‌برد.

زیان اول آن، استفاده از مقداری حافظه است، که در دوران ما عامل مهمی محسوب نمی‌شود. توجه داشته باشید که در روش‌های جدید (مثل B^+ _tree) پرونده، کپی یا مرتب نمی‌شود، بلکه فقط شاخص‌هایی برای آن ذخیره می‌شود و شاخص‌ها حافظه کمی را اشغال می‌کنند.

زیان دوم آن، مقدار زمانی است که برای بررسی پرونده و شاخص‌گذاری آن صرف می‌شود. این مقدار زمان، ممکن است قابل توجه باشد، ولی پرونده یکبار شاخص‌گذاری می‌شود و برای مدت طولانی از آن استفاده می‌شود. همچنین شاخص‌گذاری را می‌توان در موقعی که سیستم خیلی فعال نیست (مانند شب‌ها و ایام تعطیل) انجام داد. قابل ذکر است که با افزودن رکوردهای جدید به پرونده، شاخص‌های آن نیز خود به خود توسط سیستم به‌هنگام می‌شود.

۱۹. ارتباط با زبان‌های برنامه‌سازی

ارتباط SQL با زبان‌های برنامه‌سازی از طریق نرم‌افزارهایی مانند ODBC و JDBC و LINQ و... انجام می‌گیرد. مطالب دیگری نیز در رابطه با SQL وجود دارد، مانند تعریف توابع و ذخیره‌سازی آن‌ها. بیشتر این مطالب نیاز به ساختارهایی دارد که در SQL3 تعریف شده‌است. بنابراین ادامه بحث درباره SQL را به فصل مربوط به مدل شی - رابطه‌ای می‌سپاریم و ارتباط SQL با زبان‌های برنامه‌سازی از طریق نرم‌افزار LINQ را در آنجا می‌آوریم.

خلاصه فصل سوم

در این فصل به ارائه متداول‌ترین مدل پایگاهداده برای طراحی منطقی، همراه با پیاده‌سازی آن پرداختیم. مدل رابطه‌ای که توسط آقای کاد ارائه شد از مفهوم ساده و آشنای رابطه استفاده می‌کند که به‌سادگی توسط کامپیوتر به صورت جدول تعریف می‌شود و مورد استفاده قرار می‌گیرد. ابتدا کوشیدیم مفاهیم مبنایی مدل رابطه‌ای را بیاموزیم، اصلی‌ترین ابعاد آن را ارائه کنیم و نشان دهیم که چگونه می‌توان یک پایگاهداده را در قالب چند رابطه به‌هم پیوسته طراحی کرد. مفاهیمی مانند جامعیّت و کلید و... در مدل رابطه‌ای تبیین شد. سپس یک زبان کامپیوتری برای پیاده‌سازی پایگاهداده ارائه شد.

این زبان سنتی که سال‌هاست بر جهان پایگاهداده حکومت می‌کند، همچنان پیشتر از است و نسل‌ها و نمونه‌های متفاوتی دارد، SQL خوانده می‌شود. کوشیدیم ابعاد مختلف این زبان را برای تعریف و کاربرد پایگاهداده‌ها، جداول و سطرها و ستون‌های آن، پرسش و جنبه‌های مختلف آن بررسی کنیم. مهم‌ترین دستاوردهای SQL همانا پرسش است. با این وسیله می‌توان در طول زمان به نگهداری و پردازش پایگاهداده‌ها پرداخت و اطلاعات لازم را از آن استخراج کرد. داشتن پایگاهداده امن و سریع و گرفتن و دادن اطلاعات به آن را می‌توان اصلی‌ترین جنبه SQL دانست.

تمرین‌های تشریحی فصل سوم

۱. برای موارد زیر پایگاهداده مورد نیاز را طراحی و برای هر کدام چند پرسش طرح و همراه با پاسخ ارائه کنید:

- یک انبار

توضیحات: ورود و خروج کالا، دریافت‌کنندگان، واردکنندگان، صادرکنندگان، مقصد و مبدأ کالا باید مشخص باشد.

- یک پایگاهداده برای تأمین قراردادهای یک شرکت:

توضیحات: پیمانکاران، افراد و ارگان‌هایی که با شرکت قرارداد می‌بندند و ضوابط قراردادها باید مشخص باشد.

- جداول یک پایگاهداده شرکت تعمیرات برق ساختمان

توضیحات: این شرکت تعدادی پرسنل اداری و تعدادی پرسنل خدماتی دارد و مشتریانی که می‌توانند ارگان‌ها و یا اشخاصی باشند که جهت تعمیرات مراجعه می‌کنند.

- جداول پایگاهداده برای یک نمایشگاه ماشین

توضیحات: خریدوفروش ماشین، فروشنده‌گان و خریداران، پرسنل نمایشگاه، طرف حساب خریدار یا فروشنده باید مشخص باشد.

- جداول پایگاهداده برای یک مؤسسه زبان

۲. SQL را با جبر رابطه‌ای مقایسه کنید. اگر جبر رابطه‌ای عیناً پیاده‌سازی شود، چه نقاط قوت و ضعفی خواهد داشت؟

۳. دستور CREATE DOMAIN چه ارزشی دارد؟

۴. آیا به نظر شما ادغام دستورهای Σ و Π در دستور SELECT صحیح است؟
دستورهای پیوند چطور؟

۵. آیا توابع محاسباتی SQL کافی هستند؟ آیا می‌توانید توابع دیگری پیشنهاد کنید؟

۶. جداول پایگاهداده فروشگاه زنجیره‌ای را طراحی و به پرسش‌های زیر پاسخ دهید:

- مشتریانی که کالای سفارشی تحويل‌نشده دارند.

• صورت‌حساب یک مشتری خاص شامل نام مشتری، شهر، آدرس، کدپستی و تلفن به انضمام لیست آخرین سفارش.

• مشتریانی که از سفارش خود بدھی دارند.

• کارمندان ناحیه 'x' که کمتر از ۳۰۰۰۰ ریال حقوق می‌گیرند.

• قیمت کل کالاهای موجود در انبار شماره ۵

۷. جداول پایگاهداده کتابخانه را طراحی و به پرسش‌های زیر پاسخ دهید:

- اسامی کتاب‌هایی که از انتشارات Prentice Hall می‌باشند.

• کتاب‌هایی که موضوع آنها با عبارت DB آغاز می‌شود (مثل DB, DBOO, DB) به تفکیک موضوع.

- دانشجویانی که قبل از تاریخ ۱۳۷۷/۲/۲۰ کتاب به امانت برده‌اند.

- دانشجویانی که در موضوع OOP کتاب به امانت گرفته‌اند.

- دانشجویانی که حداقل دو کتاب در مورد سیستم عامل به امانت گرفته‌اند.

۸. در پایگاهداده بانکداری به پرسش‌های زیر پاسخ دهید:
یک بانک را با مسئولیت ۱۱ ایجاد کنید.

- حق update را از همه کسانی که از ۲۲ گرفته‌اند، پس بگیرید.
- مدیر شعبه‌هایی که وام گرفته و نپرداخته‌اند.
- مشتریانی که در همه شعبه‌های کرج حساب دارند.
- حساب‌هایی که به بیش از یک مشتری تعلق دارند.
- مشتریانی که مبلغ پرداخت وام آنها هیچگاه زیر ۱۰۰۰۰ نبوده است.
- حساب‌هایی که موجودی آنها از همه حساب‌ها بیشتر است.