

فصل ششم

آشنایی با مدل شئ - رابطه‌ای و SQL3

هدف کلی

نقد مدل رابطه‌ای و شناخت مدل‌های شئ‌گرا و شئ - رابطه‌ای و SQL3

هدف‌های یادگیری

۱. مدل رابطه‌ای نقد می‌شود و مدل‌های بعدی معرفی می‌شوند. مهم‌ترین این مدل‌ها، شئ‌گرا و شئ رابطه‌ای هستند.
۲. مبانی شئ‌گرایی که از دیرباز در برنامه‌سازی مطرح بوده است به طور مستقیم در پایگاه‌داده‌ها کاربرد دارد. پایگاه‌داده شئ‌گرا به عنوان یک رقیب سرسخت مطرح می‌شود.
۳. مفاهیم شئ‌گرا با مفاهیم رابطه‌ای ترکیب و مدل شئ - رابطه‌ای که خود مدل مستقلی است، معرفی می‌شود. به این ترتیب هم بسیاری از کاستی‌های مدل رابطه‌ای پوشانده می‌شود، هم میراث ارزشمند مدل رابطه‌ای حفظ می‌شود.
۴. مفاهیم مدل شئ رابطه‌ای در قالب زبان بیانی - روالی SQL3 ارائه می‌شود تا خوانندگان گرامی به توانایی‌های فوق العاده آن پی ببرند و بتوانند در عمل از آن استفاده کنند.

مقدمه

مدل رابطه‌ای اصولاً در جهت پاسخ‌گویی به نیازهای تجاری و بازرگانی در دهه‌های ۱۹۶۰ و ۱۹۷۰ پا به عرصه وجود گذاشته است. در آن دوران، کامپیوتر در همه

عرصه‌های زندگی بشر دخالت نداشت. مهم‌ترین کاربرد کامپیوتر را، امور تجاری، نظامی و زمینه‌های مشابه تشکیل می‌داد. نیازهای عام این کاربردها بیشتر بر مبنای پردازش رکورد دور می‌زد و مدل‌های پایگاه‌داده و از آن جمله پایگاه‌داده رابطه‌ای، بر این مبنای وجود آمدند. پردازش رکورد، ویژگی‌ها و محدودیت‌های خاص خود را دارد. مثلاً کارکردن با نسخه‌های مختلف یک پرونده در یک لحظه، معمول نیست و کارتیمی و همیاری و استفاده از نتایج کار دیگران در کوران کار تقریباً غیرممکن است. در کاربردهای پیشرفته پایگاه‌داده‌ها از قبیل کاربردهای چندرسانه‌ای^۱، انواع جدید داده مانند تصویر، صوت، فیلم و غیره مورد استفاده قرار می‌گیرند. تبدیل این نوع داده‌ها به رکورد بسیار مشکل و بعضاً غیرممکن است. طول یک قطعه از این نوع داده‌ها، مانند یک فیلم، بعضاً آنقدر بزرگ است که در قالب رکورد و محدودیت‌های آن نمی‌گنجد. از سوی دیگر شکستن یک داده پیوسته و ریختن آن در قالب رکوردهایی که جداگانه ذخیره می‌شوند (و بازیابی آنها) مشکل‌آفرین است. مثلاً در گوش‌دادن به یک سرود نمی‌توان متظر تأخیرهای این‌گونه‌ای در نقل و انتقال آن از حافظه کامپیوتر ماند.

۶-۱ کاستی‌ها و امتیازهای مدل رابطه‌ای

اولین ایرادی که در این ارتباط بر مدل رابطه‌ای وارد است را می‌توان «ایستابودن شما»^۲ دانست. در تعریف رابطه، فقط می‌توان از تعداد محدودی نوع داده که از پیش تعریف شده‌اند، استفاده کرد (در SQL2 نوآوری‌هایی در این ارتباط صورت گرفته ولی محدود است). این درحالی است که انواع جدید داده‌ها مرتبأ مطرح می‌شوند. درنتیجه نه تنها مدل رابطه‌ای، بلکه هیچ مدل دیگری که قادر به تعریف انواع داده‌ها به صورت پویا نباشد، جوابگو نخواهد بود. به عبارت دیگر، کاربر ماهر باید بتواند انواع جدید داده‌ها را طبق نیاز خود تعریف و از آن‌ها استفاده کند.

پایگاه‌داده رابطه‌ای که برای کاربردهای تجاری پا به عرصه وجود گذاشته است، با پرس‌وجوهای و تراکنش‌های ساده و کوتاه (ثانیه‌ای و دقیقه‌ای) سروکار دارد. کاربردهای جدید پایگاه‌داده، تراکنش‌های طولانی را شامل می‌شود که گاهی روزها و هفته‌ها طول

می‌کشند. علاوه‌بر این، قالب تعریف‌شده تراکنش در این مدل (ACID transactions) محدودیت‌هایی را اعمال می‌کند که همیاری و کار تیمی را غیرممکن می‌سازد. کاربردهای جدید نمی‌توانند بدون چنین توانایی‌هایی موفق باشند. به عنوان نمونه، تیمی که به طراحی یک شیء بزرگ و پیچیده (مانند فضایپیما) می‌پردازد، هم نیاز به استفاده از اطلاعات گسترده‌ای دارد که تنها در قالب پایگاهداده می‌گنجد و هم، نیاز به تبادل نظر مداوم در کوران کار (در حین اجرای برنامه‌ها). این نوع کار، با پایگاهداده رابطه‌ای و سایر مدل‌های کهن بیگانه است.

به دلیل طبیعت غیرروالی و مجموعه‌گرا که زبان‌های پرسش در مدل رابطه‌ای دارند، نمی‌توانند از قدرت کافی برخوردار باشند. به همین دلیل ترکیب زبان‌ها در بانک رابطه‌ای بسیار معمول است. برنامه اصلی به زبان‌های سطح بالا نوشته می‌شود و هر جا نیاز به دستیابی به پایگاهداده باشد، از امکانات SQL استفاده می‌شود. این آمیختگی نوعی عدم هم‌خوانی ذاتی به همراه دارد. زبان‌های سطح بالا با داده‌ها به صورت رکورد به رکورد برخورد می‌کنند، در صورتی که SQL آن‌ها را به شکل مجموعه (رابطه) می‌نگرد. آیا نمی‌توان یک زبان قدرتمند داشت که هم به درد برنامه‌سازی بخورد و هم پرسش؟ پاسخ مثبت است.

در پایگاهداده رابطه‌ای، داده‌ها و پردازش‌ها از هم جدا هستند و تأکید روی داده‌ها است. این دیدگاه نه با دنیای خارج هماهنگ است نه با نگرش جدید داده‌پردازی. در طبیعت، «داده‌ها» و «رفتارها» با هم هستند. مثلاً انسان عبارت است از اجزاء (داده‌ها) مانند دست، چشم، گوش و دل و رفتارها مانند برداشتن، دیدن، شنیدن و دوست‌داشتن. چشمی که نبیند، چشم نیست و گوشی که نشنود، گوش به حساب نمی‌آید. در مدل رابطه‌ای، مشخصات چشم به صورتی کاملاً مجزا از مفهوم دیدن ذخیره می‌شود. کاربر حتی می‌تواند از چشم تقاضای شنیدن کند! نگرش جدید، این نوع سازماندهی را مردود می‌داند و «چشم و دیدن و نگاه‌کردن» را به صورت یک پدیده می‌نگرد. عدم هم‌خوانی با جهان خارج، نمودهای دیگری نیز در مدل رابطه‌ای دارد. مثلاً همه چیز را به صورت جدول نگریستن، ساده‌انگاری است. در تبدیل پدیده‌های گوناگون به یک ساختار ساده مانند جدول، بدون شک ابعاد و مشخصات آن‌ها مسخ می‌شود.

در مدل رابطه‌ای، کلید رکورد مجموعه‌ای از صفات آن رکورد است. این روش مشکلاتی به همراه دارد. اول آنکه با تغییر کلید، خود رکورد نیز عوض می‌شود. در جهان خارج اگر نام و نام خانوادگی شخصی عوض شود، او همان شخص سابق باقی می‌ماند ولی در مدل رابطه‌ای اگر این صفت کلید رکورد باشد، به شخص دیگری تبدیل می‌شود. کلید رکورد در تئوری، یعنی وسیله‌ای ثابت و همیشگی برای شناسایی آن. این شناسه نباید با تغییر بخشی از اطلاعات مربوط به رکورد که ممکن است گاهی با خطا وارد کامپیوتر شده باشد، عوض شود. این مسئله جنبه‌های امنیتی پیچیده‌ای دارد. شخص با استفاده از اسم جعلی، هویت دیگری پیدا می‌کند و پیگیری سوابق او با نام قبلی زمان‌گیر و اشتباه برانگیز است.

در دنیای جدید داده‌پردازی، نسخه‌های متعددی از یک پدیده وجود دارد. مثلاً نسخه‌های متفاوت یک برنامه که در شرایط گوناگون اجرا می‌شود، یا نسخه‌های مختلف طراحی یک خودرو که هنوز نهایی نشده‌است، همگی یک چیز هستند. روش تعریف کلید در بانک رابطه‌ای به سادگی نمی‌تواند این یکسانی را ایجاد کند، به خصوص که تعداد این نسخه‌ها از پیش معلوم نیست. مشکل دیگری که در روش تعریف کلید در بانک رابطه‌ای پیش می‌آید، رجوع به رکوردها توسط رکوردهای دیگر است. ارتباط بین رابطه‌های مختلف در این مدل، توسط کلید خارجی برقرار می‌شود. کترل این ارتباط در یک پایگاهداده پر حجم بسیار زمان‌گیر است. مثلاً در پایگاهداده آموزش و پرورش، وقتی دانش‌آموزی فارغ‌التحصیل و از گردونه سیستم حذف می‌شود، باید ضمن حذف رکورد وی به همه جای سیستم سرکشید و هرگونه ارجاع به رکورد او را حذف کرد. در سیستم‌های خیلی بزرگ این عمل بسیار دشوار و بعضاً غیرممکن است و باعث عدم جامعیت¹ در پایگاهداده می‌شود.

پایگاهداده رابطه‌ای، نقاط ضعف دیگری نیز دارد که برای جلوگیری از زیاده‌گویی به همین مقدار بستنده می‌شود. از جمله این نقاط ضعف که خود آقای کاد نیز به بعضی از آن‌ها پی‌برد و مدل خود را گسترش داد، می‌توان ناتوانی نسبی در بیان معنی، پشتیبانی نکردن از پرسش بازگشتهای تو در تو²، ابهام در استفاده از داده‌ها (مثلاً مقایسه مبهمی برای نام شخص و آدرس او امکان‌پذیر است)، ناتوانی در

1. Dangling References
2. Nested Relations

پاسخ‌گویی به پرسش‌هایی که اطلاعات آنها در سیستم موجود است، مسائل حل نشده و ظاهراً غیرقابل حل مانند بهنگام‌سازی از طریق دید و مشکلات در بیان ارتباط بین چند جدول را نام برد. بیشتر این موارد از نیازهای داده‌پردازی زمان حاضر ناشی می‌شود و بر مدل رابطه‌ای، که متعلق به دهه‌های گذشته است، نباید خرده گرفت. باید توجه داشت که بعضی از این مسائل هنوز هم پاسخ مشخصی ندارند.

مثال: به عنوان نمونه، نگاهی می‌اندازیم به جداول نشر که در سراسر این کتاب مورد استفاده قرار گرفته است و آنها را نقد می‌کنیم. در طراحی این جدول دقت کافی صورت گرفته و اشکال‌های موجود به دلیل کاستی‌های مدل رابطه‌ای است.

۱. ستون‌های جدول کارکنان در جداول مؤلف هم تکرار شده‌است. از سوی دیگر جدول کارکنان می‌تواند به چند نوع که هر کدام جدول جداگانه‌ای هستند، تقسیم شود (مانند تایپیست، فروشنده، ...). پاسخ این نیاز در مفاهیمی چون وراثت (ارث‌بری) نهفته است که در مدل رابطه‌ای ممنوع است (1NF).

۲. ستون‌هایی از این جداول دارای بخش‌های جداگانه‌ای هستند، مانند آدرس که خود بخش‌هایی چون استان، شهر، خیابان، پلاک، واحد دارد. در مدل رابطه‌ای یا باید همگی با هم در یک نام (مانند address) ذخیره شوند و یا در نام‌های جداگانه مستقلی بیایند. هر دو راه حل، بد است و به کاستی‌های مدل رابطه‌ای باز می‌گردد. جداول تودرتو یک راه حل برای این مشکل است که در ادامه ارائه خواهد شد.

۳. به تعریف جدول کتاب دقت کنید. در مدل رابطه‌ای اگر بخواهیم هر کتاب بتواند چند مؤلف داشته باشد T مجبوریم کلید مؤلف (ano) را هم به عنوان بخشی از کلید اصلی کتاب بپذیریم (همین‌کار را هم کرده‌ایم). در این صورت هر کتاب چند سطر از این جدول را اشغال می‌کند. توجه کنید که همه ستون‌های جدول کتاب در این چند سطر تکرار می‌شود. این تکرار بسیاری را «افزونگی داده» نامیده‌اند که در فصل نرم‌افزاری جداول به آن پرداختیم. اگر محدودیت‌های مدل رابطه‌ای نباشد، می‌توان کلید مؤلف را به صورت یک مجموعه تعریف کرد و برای هر کتاب، یک سطر از جدول را اختصاص داد. همین‌طلب در مورد کلید ترکیبی جدول ناشر نیز صحیح است. دقت کنید که یک شخص نمی‌تواند بیش از یک انتشارات از یک نوع داشته باشد! بنابراین در تعریف جداول، همواره نمی‌توان خصوصیت‌های مورد نظر را اعمال کرد. بهوضوح دیده می‌شود که مدل رابطه‌ای مشکلات جدی دارد.

بی‌انصافی است اگر امتیازهای ارزشمند مدل رابطه‌ای را نگوییم. در سال‌های متتمادی، پژوهش‌های ارزشمندی در زمینه‌های مختلف پایگاهداده‌ها در قالب مدل رابطه‌ای انجام شده و مقالات و کتاب‌های بسیار زیاد و معتبری به رشتہ تحریر درآمده است. مسائل پایگاهداده رابطه‌ای نوعاً دارای راه حل‌های روشی هستند و نقاط ضعف و برتری هر راه حل مشخص است. در مرحله پیاده‌سازی، به سادگی می‌توان تصمیم گرفت که در چه شرایطی از کدام راه حل باید سود جست. مدل رابطه‌ای پشتوانه نظری قوی و توانمندی دارد. مفاهیم این مدل با روش ریاضی در سه قالب مختلف که در عین حال معادل یکدیگرند (جبر رابطه‌ای، حساب رابطه‌ای تاپلی و دامنه‌ای) بیان شده و صحت آن‌ها به اثبات رسیده است. در این مدل همه مفاهیم در قالب یک و تنها یک ساختار بیان می‌شود: رابطه. این مفهوم را می‌توان معادل جدول دو بعدی دانست که برای همگان قابل فهم است. پس، اولین عامل موفقیت مدل رابطه‌ای، سادگی آن است. فهم عمومی و یکنواخت مفاهیم این مدل بسیار ارزشمند است. وقتی در این مدل از مفهومی صحبت می‌شود، همگان برداشتی مشابه از آن دارند.

نرم‌افزارهای توانمند و جاافتاده‌ای در مدل رابطه‌ای وجود دارد. به عنوان نمونه می‌توان از نسل‌ها و پیاده‌سازی‌های متنوع زبان پرسش SQL نام برد که بر دنیای پایگاهداده‌ها به صورتی فراگیر سایه افکنده است تا جایی که بسیاری از مفاهیم در پایگاهداده شیگرا نیز با این زبان پیاده‌سازی شده‌اند! زبان جاوا نیز که این روزها بسیار پر طرفدار است، به سادگی به پایگاهداده و SQL متصل می‌شود. جامعیت در مدل رابطه‌ای با استفاده از زبان‌های سطح بالا و بیان محدودیت‌های جامعیتی، قابل پیاده‌سازی است. بهینه‌سازی پرس‌وجو در مدل رابطه‌ای رهیافتی ارزشمند و بسیار نظری است. تکیه بر زبان‌های بیانی در این مدل به سیستم این امکان را می‌دهد که دستورالعمل‌های ضعیف و غیرفنی کاربران را قبل از اجرا بهینه‌سازی کند. علاوه بر تبدیل این دستورها به شکل معادل بهینه، سیستم پایگاهداده رابطه‌ای تمهیدات دیگری نیز در راستای پردازش سریع و ارزان پرس‌وجوها می‌اندیشد و آن‌ها را به نحو شایسته‌ای سامان می‌دهد. وجود قوانین مشخص، برای مباحثی چون «وابستگی تابعی» و «وابستگی چندگانه» و «نرمال‌سازی جداویل» به طراح پایگاهداده کمک می‌کند که محصول خود را مطابق استاندارد بسازد، تغییرات لازم را در طول زمان به صورت قانونمندی اعمال کند و از کارایی آن مطمئن باشد.

۶-۲ شئ‌گرایی در پایگاهداده‌ها

عمده‌ترین مفهومی که پس از رابطه در پایگاهداده، پذیرش همگانی یافت و انقلاب دوم را رقم زد، شئ‌گرایی است. شئ‌گرایی ابتدا به صورت مستقل و سپس با ترکیب تیزبینانه با مفاهیم مدل رابطه‌ای، توانست جای خود را در پایگاهداده‌ها باز کند و در واقع به نوعی صاحب‌خانه تبدیل شود تا جایی که امروزه مدل شیء-رابطه‌ای به موفق‌ترین مدل فرارابطه‌ای تبدیل شده است.

مدل شیء‌گرا، توجه بسیاری از متخصصان پایگاهداده را به خود جلب کرده و کتاب‌ها و مقالات زیادی در این رابطه نوشته شده است. مفاهیم پایگاهداده به صورت‌های متفاوتی در این مدل پیاده‌سازی شده و سیستم‌های مدیریت پایگاهداده موفقی به بازار عرضه شده است. یکی از بزرگ‌ترین دلایل موفقیت پایگاهداده شیء‌گرا، امکان تعریف نوع داده توسط کاربر است. این مدل به کاربر امکان می‌دهد که ضمن استفاده از انواع ساده داده‌ها مانند عدد صحیح، عدد حقیقی، تاریخ، و غیره، خود نیز به تعریف نوع داده بپردازد. مفاهیمی از قبیل کلاس یا رده، سلسله‌مراتب کلاس‌ها^۱ وراثت، چندریختی^۲ و مصرف دوباره^۳ به این مدل قدرت فوق العاده‌ای می‌بخشد.

زبان‌های برنامه‌نویسی شیء‌گرا در زمینه‌های مختلف و از آن جمله، در زمینه پایگاهداده‌ها، کم نیستند. طراحان این زبان‌ها کوشیده‌اند نیازها را در هم ادغام کنند و یک زبان فراگیر ارائه دهند. در حال حاضر سیستم‌های مختلف شیء‌گرا، دستیابی به پایگاهداده را با روش‌های مختلفی سامان می‌دهند. دست‌اندرکاران پایگاهداده شیء‌گرا تأکید دارند که از امکانات خوب موجود در هر زمینه‌ای که باشد، به‌نحو مطلوب استفاده شود.

یکی از اصول شیء‌گرایی، بر داده‌های فعال استوار است. شیء‌گرایی، شعار «دنیا را به همان شکل که هست بنگریم» را سرلوحه کار خود ساخته است. در طبیعت، اشیا حداقل دو چهره دارند: اجزاء (داده‌ها) و رفتارها. شیء‌گرایی این دو مفهوم را در قالب کلاس ادغام کرده است.

1. Class Hierarchy
2. Polymorphism
3. Reuse

نوع پیشرفت‌های از اشیاء در دنیای شئ‌گرائی به نام شئ فعال^۱ مطرح شده است که در بعضی از زمینه‌های پایگاهداده کارایی زیادی دارد. شئ فعال، تحت شرایطی به طور خودکار شروع به کار می‌کند و بعضی از متدهای خود را بر حسب مورد فعال می‌سازد. در زمینه‌های مختلفی از قبیل بن‌بست، کنترل هم‌روندی، بازیافت^۲ و کنترل تراکنش‌های طولانی به خوبی می‌توان از این مقوله سود جست. مثلاً شئ فعالی می‌توان تعریف کرد که به محض بُروز هر نوع خرابی^۳ با توجه به نوع آن خرابی، اقدام لازم را آغاز کند و از دست‌رفتن اطلاعات و ضربه‌زدن به جامعیت و امنیت پایگاهداده جلوگیری کند.

در شئ‌گرائی، هر پدیده با هر کیفیت که باشد به محض ایجاد، دارای شناسه^۴ واحدی می‌شود و تا هر زمان که در سیستم وجود دارد، این شناسه را خواهد داشت. شناسه، مستقل از محتوای شئ است، توسط سیستم تعیین می‌شود و مورد استفاده قرار می‌گیرد. با تغییر محتوای شئ به هر شکل و به هر میزان، شناسه آن عوض نمی‌شود. نسخه‌های متعدد یک شئ هر کدام شناسه ویژه خود را دارند، ولی در عین حال به یکدیگر مرتبط هستند.

پایگاهداده شئ‌گرا، یک سیستم باز است یعنی همواره می‌توان داده‌ها، دامنه‌ها و پردازش‌های جدیدی به آن افزود. در صورت نیاز به هرگونه اطلاعات می‌توان «متدهایی» جهت دستیابی به آن‌ها در کلاس مربوطه تعریف و به سیستم اضافه کرد. پایگاهداده شئ‌گرا نقاط ضعفی نیز دارد که از مهم‌ترین آن‌ها می‌توان عدم وجود استانداردهای پذیرفته شده، کار نظری قابل مقایسه با بانک رابطه‌ای، زبان پرسش مستقل و جاافتاده، مشکل بهینه سازی پرس وجو و متفاوت بودن در فهم عمومی پایگاهداده را نام برد. قطعاً این مدل همه پیچیدگی‌های پایگاهداده را حل نکرده و ممکن است در آینده نیز به چنین توفیقی دست نیابد و احتمال دارد مدل بهتری جایگزین آن شود.

1. Active Object

2. Recovery

3. Failure

4. Identifier

۱-۲-۶ مفاهیم بنیادین مدل شیء‌گرا

در این بخش عمده‌ترین مفاهیم شیء‌گرا را به اختصار مرور می‌کنیم:

۱. کلاس

قالب اطلاعاتی که ویژگی‌های مورد نظر را برای مجموعه همسان از اشیای دنیای واقعی بیان می‌کند (مثلاً کلاس دانشجو یا کلاس استاد).

۲. شیء

مجموعه مقادیر داده‌هایی که یک پدیده دنیای واقعی را معرفی می‌کند و نمونه‌ای از کلاس است (مثل هریک از دانشجوها).

هر کلاس یا شیء شامل موارد زیر است:

- مجموعه متغیرها که قالب داده‌های مربوط به آن کلاس را نگهداری می‌کنند.
- مجموعه متدها که هریک، برنامه‌ای است که برای پیاده‌سازی یک رفتار نوشته شده است. پیام به معنی فراخوانی یک متده است یا رویه^۱ است. در هر متده فقط می‌توان به متغیرهای همان شیء به‌طور مستقیم دسترسی داشت. برای دسترسی به داده‌های اشیای دیگر به ارسال پیام نیاز دارد.

مثال: کلاس مؤلف

```
class author
{ /* variables*/
    int id;
    string name;
    string adres;
    ...
/*methods*/
    string get-name();
    int set-address(string new -address); };
```

بدنه متدها معمولاً به‌طور جداگانه پیاده‌سازی می‌شود.

۳. شناسه شیء^۲

برای برقراری تناظر یک‌به‌یک بین هر پدیده در دنیای واقعی با هر شیء ذخیره‌شده در پایگاه داده‌ها به کار می‌رود. این شناسه به مرور زمان و حتی با تغییر برخی یا همه مقادیر صفات آن پدیده، تغییر نخواهد کرد. شناسه می‌تواند یکی از این انتخاب‌ها باشد:

- نام: که توسط کاربر تعريف می‌شود.
- توکار^۱: شناسه موجود در زبان برنامه‌نویسی یا مدل داده‌ای.
- مقداری که توسط سیستم بانک اطلاعات تولید می‌شود^۲، یا یک مقدار منحصر به فرد در عالم خارج از بانک اطلاعات مثل شماره ملی فرد.

۴. وراثت

نکته دیگری که اهمیت دارد، اشتراک داده‌ها و رفتارها است. به عنوان نمونه رفتاری مانند «شنیدن» به بسیاری از جانداران مربوط می‌شود (هر چند هریک متفاوت می‌شنوند). آیا در شبیه‌سازی با غوحس باشد برای هر موجودی «گوش» و «شنیدن» را جداگانه تعريف کرد؟ می‌توان اصول شنیدن را جداگانه تعريف کرد و برای همه به اشتراک گذاشت. مفهوم وراثت برای پاسخ‌گویی به این نیاز پدید آمده است. وراثت یعنی اشتراق کلاس‌ها از یکدیگر و استفاده از کلاس‌ها در تعريف کلاس‌های دیگر. به این ترتیب سلسله‌مراتبی از کلاس‌ها ایجاد می‌شود که قدرت و انعطاف‌پذیری زیادی به برنامه‌ساز می‌بخشد. سلسله‌مراتب کلاس‌ها به صورت درخت است. در این درخت، کلاس‌ها در رده‌های مختلف به صورت آبرکلاس^۳ و زیرکلاس^۴ قرار می‌گیرند. در بالاترین سطح، کلاسی قرار دارد که خصوصیات مشترک همه اشیای مورد نظر را در خود جای می‌دهد. این کلاس از هیچ کلاس دیگری مشتق نمی‌شود (ارث‌بری ندارد). هر زیرکلاس، داده‌ها و رفتارهای کلاس‌های بالاتر را به ارث می‌برد و نیازهای ویژه خود را به آن می‌افزاید. کلاس ممکن است بعضی از ویژگی‌های کلاس بالایی را به ارث نبرد (حذف کند) و یا آنها را تغییر دهد.

۵. بسته‌بندی^۵

کنار هم قراردادن داده‌ها و رفتارها در یک بسته (کپسول) را بسته‌بندی گویند. این نگرش باعث استقلال پدیده‌ها می‌شود یعنی هرچه به یک پدیده در کاربرد مورد نظر مربوط می‌شود، همراه او است. بسته‌بندی خاصیت بسیار ارزشمند دیگری دارد که به «پنهان‌سازی اطلاعات» موسوم است. کلاس‌ها و اشیاء دو چهره دارند:

1. Built-in
2. System Generated
3. Superclass
4. Subclass
5. Encapsulation

- چهره درونی^۱ که شامل تمام داده‌ها و برخی از رفتارهای کلاس یا شیء است. دسترسی به این بخش فقط و فقط توسط خود کلاس یا شیء امکان‌پذیر است. اشیاء از ساختار درونی یکدیگر بی‌خبر هستند و به جزئیات هم کاری ندارند.
- چهره رابط^۲ که شامل تعدادی از رفتارها است، تنها وسیله ارتباطی اشیاء، چهره رابط آن‌ها است. اشیای درون یک سیستم با فراخوانی چهره رابط یکدیگر (که همان پیامرسانی است) کارها را انجام می‌دهند.

یک قانون مهم در بسته‌بندی این است: «چهره رابط باید کمینه باشد». به عبارت دیگر باید تا سرحد امکان در پنهان‌سازی اطلاعات کوشید. این موضوع نیز از مهم‌ترین دلایل موفقیت مدل شیء‌گرا است، زیرا راه حلی اساسی برای حل بزرگ‌ترین مشکل سیستم‌های نرم‌افزاری یعنی نگهداری سیستم^۳ است. هر سیستم روزی تولید می‌شود و از آن به بعد با تغییر شرایط باید مرتباً به‌روز در می‌آید. تغییر بخشی از نرم‌افزار، باعث سرایت مشکلات آن به بخش‌های دیگر می‌شود و به صورت گره کوری در می‌آید. در دنیای شیء‌گرایی، این سرایت به مقدار کمینه می‌رسد، زیرا اشیاء با چهره درونی هم کاری ندارند، خواه تغییر کند یا نکند. نکته بسیار مهمی که باید به آن توجه کرد، طراحی دقیق چهره رابط است. این چهره باید حالتی پایا داشته باشد و با آینده‌نگری، به‌سادگی نیاز به تغییر نداشته باشد.

۶. چند ریختی

در زبان‌های برنامه‌سازی معمولی، استفاده از اسمی تکراری به صورت محدودی امکان‌پذیر است. در دنیای شیء‌گرایی، اسم تکراری به فراوانی رخ می‌دهد، زیرا شیء‌گرایی یعنی شبیه‌سازی جهان خارج که در آن اسمی تکرار می‌شوند. مثلاً کلمه «خوابیدن» برای بسیاری از پدیده‌ها از جمله حیوانات گوناگون استفاده می‌شود و هر کدام به شیوه خود می‌خوابند. شبیه‌سازی این مفهوم را چند ریختی می‌نامند. چند ریختی به اشیای مختلف این امکان را می‌دهد که به یک پیام واحد پاسخ‌های متفاوتی بدهنند. رفتارهایی که شامل چند ریختی می‌شوند در چهره رابط مشابه ولی در پیاده‌سازی متفاوت هستند. آن‌ها نام یکسان و احتمالاً پارامترهای متفاوت دارند.

1. Internal

2. Interface

3. System Maintenance

چندريختی باعث کوتاهشدن و سادهشدن برنامه‌ها می‌شود. به عنوان نمونه در يك حلقة تکرار، می‌توان يك پیام را به اشيای مختلف فرستاد و آن‌ها خود می‌دانند چگونه عمل کنند و پاسخ دهند. مثلاً اگر بخواهیم مسابقه دو، بين حیوانات مختلف را شبیه‌سازی کنیم، کافی است هم‌زمان پیام «بدو» و سپس پیام «بایست» را به همه آن‌ها ارسال کنیم. اگر چندريختی وجود نداشت باید برای هر کدام پیغامی متفاوت (با نام‌های مختلف) بفرستیم.

۲-۲ مدل شئ - رابطه‌ای^۱

مدل شئ‌گرا با وجود توانایی‌های مفیدش، نتوانست به تنهایی جانشین مناسب و بلامنازعی برای مدل محبوب رابطه‌ای باشد. پژوهش‌گران و متخصصان برآن شدند تا برای رسیدن به مدلی بهتر، تلفیقی از دو مدل موفق رابطه‌ای و شئ‌گرا به نام مدل شئ - رابطه‌ای ارائه کنند، یعنی از مدل رابطه‌ای در ذخیره و بازیابی داده‌ها^۲ و از هر دو مدل برای مدل‌سازی داده‌ها استفاده کنند. ترکیب مفاهیم شئ‌گرا و رابطه‌ای به دو صورت زیر ممکن است:

۱. قراردادن جداول و اشیاء در کنار هم. به عبارت دیگر، استفاده از جداول برای بخش‌های ساده و سنتی پایگاه‌داده اطلاعات و همراه با آن استفاده از اشیاء برای بخش‌های پیچیده. این تصور، همان مدل شئ‌گرا است زیرا جدول نیز نوعی شئ است.

۲. وفادار ماندن به قالب جدول و ارتقاء جداول با استفاده از مفاهیم شئ‌گرایی در آن‌ها. این نگرش، که مدل شئ - رابطه‌ای نام دارد، ذهنیت مدل رابطه‌ای را حفظ می‌کند یعنی پایگاه‌داده باز هم مجموعه‌ای از جداول است. لازم به یادآوری است که مدل شئ - رابطه‌ای، بعضی از اصول مدل رابطه‌ای مانند INF را زیر پا می‌گذارد.

در این بخش عمده‌ترین مفاهیم شئ‌گرا که به جداول افزوده می‌شوند تشریح می‌شود:

1. Object Relational Model (ORM)
2. Data Access

۱. رابطه‌های تودرتو

در مدل رابطه‌ای همه صفات باید دامنه‌های تجزیه‌ناپذیر داشته باشند (سطح نرمال ۱NF). پس مقدار یک صفت، نمی‌تواند یک رابطه یا مجموعه یا آرایه باشد. به عبارت دیگر در مدل رابطه‌ای روابط مرکب پشتیبانی نمی‌شود.

اولین ساختاری که به مدل رابطه‌ای اضافه شده، رابطه تودرتو است. می‌توان دامنه ستون‌های جدول را از نوع جدول یا حتی مجموعه و... تعریف کرد. این پیشرفت قابل ملاحظه است و امکان بیان مفاهیم را به صورتی گسترده‌تر و دقیق‌تر فراهم می‌آورد. مثال: جدول ناشر را به صورت زیر در نظر بگیرید:

Publisher (pno, title, typ, doe, adres)

(آدرس، تاریخ تأسیس، نوع انتشارات، عنوان ناشر، کلید مدیر نشر) ناشر
حداقل دو ایراد بر این طراحی وارد است:

- فقط کلید مدیر نشر آمده است. چرا نباید مشخصات کامل او بیاید؟
- بخش آدرس، یا شامل همه قسمت‌ها به صورت چسبیده و یا پراکنده و به عنوان سه ستون مجزا تعریف می‌شود.

در پایگاهداده شیء - رابطه‌ای، این جدول را می‌توان به صورت زیر با استفاده از جداول تودرتو تعریف کرد:

publisher

<u>pno</u>	address: addr			manager:personel		
	city	street	no	title	typ	doe

مسلم است که جداول در یکدیگر کپی نمی‌شوند. ساختارهایی مانند ref در مدل شیء - رابطه‌ای از کپی‌شدن جداول و ایجاد افزونگی جلوگیری می‌کند. این عملگر در SQL^۳ برای انجام انواع پیوند پیاده‌سازی شده و سرعت آن را به خوبی افزایش داده است.

علاوه‌بر جداول تودرتو، از ساختارهای دیگر مانند SET و ARRAY نیز می‌توان در تعریف جدول استفاده کرد.

جداول را می‌توان به صورت بازگشتی^۱ تعریف کرد. مثلاً دستیار یک مؤلف می‌تواند مؤلف دیگری از نوع author باشد. این نوع تعریف، پاسخ‌گویی به پرس‌وجوهای را آسان می‌کند. مثلاً پرس‌وجویی مانند «مؤلف‌هایی که کمتر از دستیار خود درآمد دارند» به صورت بازگشتی قابل پاسخ‌گویی است.

۲. بسته‌بندی

امکان دیگری که می‌توان از شئ‌گرایی اقتباس کرد و به مدل رابطه‌ای افزود، بسته‌بندی است. بسته‌بندی یعنی قراردادن داده‌ها و رفتارها در یک بسته. به عنوان مثال، ناشر، رفتارهای مشخصی را به همراه دارد مانند دریافت مجوز و بستن قرارداد و غیره. طبیعی است که ساختاری به نام ناشر چنین متدهایی (توابع یا روال) را هم در خود ذخیره کند. این نکته با نام‌هایی چون برنامه‌سازی پیمانه‌ای نیز شناخته شده است. در بخش SQL3 مثال و توضیح بیشتری خواهیم دید.

۳. وراثت (ارث‌بری)

امکان دیگری که از شئ‌گرایی اقتباس و به بانک رابطه‌ای افزوده شده است، مفهوم ارث‌بری است. اگر جدولی حالت خاصی از جدول دیگری باشد، می‌تواند به عنوان فرزند آن شناخته شود و مشخصات آن را به ارث ببرد. این امکان، حجم کد را کاهش می‌دهد و باعث اشتراک داده‌ها و رفتارها بین جداول می‌شود. به عنوان مثال، جدول مؤلف تمام خصوصیات جدول کارکنان را به ارث می‌برد و ویژگی‌های خاص خودش را نیز به آن می‌افزاید. در اینجا مفهوم پیوند پویا^۲ نیز مطرح می‌شود، یعنی مثلاً نوشتن کتاب برای دوره کارشناسی یا دکتری می‌تواند معنی متفاوتی داشته باشد. در بخش SQL3 مثال و توضیح بیشتری خواهیم دید.

۴. سایر موارد

پایگاهداده شئ – رابطه‌ای در زمینه‌های مختلف شئ‌گرایی و غیرشئ‌گرایی، پیشرفتهای چشمگیری داشته است. تبلور این پیشرفتهای SQL^۳ خودنمایی می‌کند که به طور خلاصه بررسی می‌شود.

1. Recursive

2. Dynamic Binding

SQL^۳ معرفی ۳-۶**۱-۳-۶ ویژگی‌های غیرشیء‌گرایی**

الف) انواع جدید داده‌ها

۱. SQL2 و CLOB و BLOB مانند

۲. BOOLEAN که مقادیر True, False و Unknown را دربر می‌گیرد. عباراتی از قبیل Is True و Is Not False و نیز نتایج شرط‌ها در این نوع داده مورد استفاده قرار می‌گیرند.

۳. ENUMERATED همانند زبان‌های برنامه‌سازی چون پاسکال، مجموعه‌ای از داده‌های تعریف‌شده را دربر می‌گیرد که ترتیب آن‌ها نیز مشخص است.

۴. ساختارهای گروهی شامل TUPLE, LIST, ARRAY, SET, ROW, با عضو تکراری) و ترکیب آن‌ها. MULTILIST

ب. گزاره‌های جدید

- گزاره DISTINCT، معادل بودن دو سطر را مشخص می‌کند و TRUE یا FALSE باز می‌گرداند. مثلاً در دستور زیر اگر حداقل یک ستون this با ستون متناظر برابر نباشد یا NULL باشد T مقدار True باز می‌گردد.

this IS DISTINCT FROM that

- گزار SIMILAR که از سیستم عامل UNIX اقتباس شده است، با شکل متفاوتی پشتیبانی می‌شود.

WHERE NAME SIMILAR TO

مثال:

‘ALI _(‘REZA’| ‘ZADE’, ’POUR’)

پ) پیشرفت‌های معنایی، دیدهای قابل به روز درآوردن^۱ به روز درآوردن دید در SQL2 بسیار محدود است. در SQL3 کاربران می‌توانند به روزرسانی‌های بیشتری انجام دهند. این محدودیت‌ها، بستگی زیادی به وابستگی‌های موجود بین داده‌ها دارد که باعث می‌شود به روزرسانی یک جدول روی جداول دیگر تأثیر گذارد.

پرسش‌های بازگشتی

در پرس‌وجوهای بازگشتی، به هر زیرپرسش یک نام، داده می‌شود و سپس از این نام در همان پرسش استفاده می‌شود. شکل کلی آن به صورت زیر است:

```
WITH RECURSIVE
Q1 AS SELECT ....
Q2 AS SELECT ....
SELECT .... FROM Q1, Q2 WHERE ....
```

مکان‌نما^۱ و محل‌نما^۲

می‌توان مکان‌نما را در حالت‌های INSENSITIVE و SENSITIVE قرار داد. در حالت دوم دستور OPEN باعث ایجاد نسخه جدیدی از داده‌ها می‌شود و تغییرات روی نسخه اصلی اثر نمی‌گذارد. با تغییر وضعیت مکان‌نما به SENSITIVE، تغییرات قبلی روی نسخه اصلی اعمال می‌شود.

در مواردی که تصمیم قطعی برای تغییر داده‌ها، بستگی به شرایطی دارد، می‌توان از این امکان استفاده کرد.

نقشه بازرسی^۳

تراکنش‌های طولانی، اگر دچار سقوط شوند، به روال‌های بازگشتی طولانی بر می‌خورند. برای جلوگیری از این وضعیت می‌توان آن‌ها را به چند زیرتراکنش تقسیم کرد و برای هر کدام عمل بازگشت را جداگانه انجام داد. در SQL3 این عمل با تعریف SAVE POINT شبیه‌سازی می‌شود و با دستورهای زیر پشتیبانی می‌شوند:

```
ROLL BACK TO SAVE POINT
RELEASE SAVE POINT
```

موارد زیاد دیگر مانند راه‌انداختن خودکار، انواع جدید پیوند، تابع و رویه و متدهای چندین دستور جدید برای پشتیبانی عبارات شرطی (if-then-else)، ترتیب اجرای دستورها (for, while,...) و غیره اضافه شده‌اند که در این مختصراً نمی‌گنجد. علاقه‌مندان می‌توانند به [حق‌جو، ۱۳۹۳] مراجعه کنند.

1. Cursor
2. Locator
3. Save Point

۶-۳-۲- ویژگی‌های شئ‌گرایی

الف) ارثبری

مفهوم ارثبری در SQL3 با زیرجدول و ابرجدول پشتیبانی می‌شود. مثلاً می‌توان جدول author را به عنوان زیرجدول personel تعریف کرد. در این صورت تمام ستون‌ها و رفتارهای آن و نیز کلید اصلی آن را به ارث می‌برد. ستون‌های به ارث برده شده می‌توانند تغییر نام یابند و می‌توان ستون‌های جدیدی به آن اضافه کرد. سطرهای متناظر دو جدول با کلید اصلی مشترک مشخص می‌شود.

از یک آبرجدول می‌توان چندین زیر جدول مشتق کرد و هر زیرجدول نیز به نوبه خود می‌تواند ابرجدول سطوح پایین‌تر، بدون محدودیت باشد.

ب) نوع داده کاربر - تعریف^۱ و ref^۲

نوع داده کاربر - تعریف ویژگی‌های زیر را دارد:

- می‌توان از همه انواع داده‌ها در آن استفاده کرد.
- جنبه‌های رفتاری آن توسط توابع، رووال‌ها و متدها تأمین می‌شود.

با استفاده از توابع سیستمی get و set، دستیابی به داده‌ها انجام می‌شود و مقدار متغیرها که از هر نوعی می‌توانند باشند. (حتی انواع کاربر - تعریف دیگر) ذخیره و بازیابی می‌شود. مقایسه مقادیر آن‌ها فقط از طریق توابع کاربر - تعریف انجام می‌شود. می‌توان از زیرنوع^۲ و ابرنوع^۳ در آن‌ها استفاده کرد و ارثبری چندگانه به صورت محدود پشتیبانی می‌شود.

مثال:

```

CREATE TYPE author_type
UNDER personel_type
AS (degree, major, reseme)
INSTANTTABLE
REF(ano)
METHOD find_info(ano integer);
.....
CREATE METHOD find_info (ano integer)
BEGIN
IF self.typ = 1 and sex = m THEN ...
END.

```

1. User-Defined Type

2. Subtype

3. Supertype

SQL3 تفاوت‌های عمدۀ ای بین توابع و متدها قائل است. مهم‌ترین تفاوت این است که متدها حتماً باید به یک نوع کاربر-تعریف وابسته باشند، ولی توابع آزادانه مورد استفاده قرار می‌گیرند. همچنین نقطه‌گذاری^۱ مخصوص متدهاست.

SQL3 نوع ویژه‌ای به نام REF دارد که مقدار متغیرهای این نوع، شناسهٔ شئ است. از این نوع می‌توان برای ارجاع استفاده کرد. مثلاً برای مؤلفهایی که در جدول پرسنل وجود دارند، می‌توان در جدول book ستون ano را به صورت زیر تعریف کرد تا از نوع personel باشد:

ano REF (personel(pno))

از این نوع داده می‌توان شبیه لیست پیوندی برای ارجاع به بخش‌های آن استفاده کرد. مثلاً عبارت زیر، مدرک تحصیلی مؤلف را باز می‌گرداند:

```
SELECT author-> degree
FROM book
WHERE .....
```

۴- مقایسه

مدل شئ‌گرا در دوران ما رقیبی سرسرخت برای مدل رابطه‌ای است و توجه بیشتر صاحب‌نظران این مقوله، حتی متعصبین سابق مدل رابطه‌ای را به خود جلب کرده‌است. پژوهش‌های گسترده‌ای در این زمینه صورت گرفته و محصولات تجاری ارزشمند با کارایی بالا عرضه شده‌است. در حال حاضر، ترکیب شئ‌گرایی و رابطه‌ای که مدل شئ - رابطه‌ای را فراهم آورده‌است، بر دنیای پایگاه‌داده حکومت می‌کند. محصولات رابطه‌ای، خود را با زیورهایی که از شئ‌گرایی به عاریت گرفته‌اند، می‌آرایند و محصولات شئ‌گرایی، به پشتیبانی از مفاهیم رابطه‌ای و ارتباط‌داشتن با گذشته می‌بالند. ما در این بخش کوشیدیم شمائی از وضعیت موجود را ارائه دهیم و خوانندگان محترم را با دنیای جدید پایگاه‌داده آشنا کنیم. با توجه به نوین‌بودن مطلب، کوشیدیم جانب اختصار را رعایت کنیم.

اگر بخواهیم مقایسه کوتاهی انجام دهیم، موارد زیر را می‌توانیم یادآور شویم:

- تصویر ادراکی بانک رابطه‌ای، همتای سلسله‌مراتب کلاس‌ها است.
- تعریف رابطه‌ها در بانک رابطه‌ای، همتای تعریف انواع داده‌ها است.

- تاپل یا رکورد در بانک رابطه‌ای، همتای شیء است.
- زبان پرسش در مدل شیء‌گرا، وابسته به زبان‌های روالی شیء‌گرا است و در مدل شیء-رابطه‌ای از مدل‌های رابطه‌ای و شیء‌گرا وام گرفته شده است.
- مفاهیم زیر که در مدل شیء‌گرا اهمیت ویژه‌ای دارند و در داده‌پردازی بسیار ارزشمند هستند، در مدل رابطه‌ای همانندی ندارند:
 ۱. شناسه شیء
 ۲. ارتبری
 ۳. تغییر پویای تصویر ادراکی
 ۴. بسته‌بندی
 ۵. پیام‌رسانی

مفاهیم دیگری نیز در پایگاهداده وجود دارد که در مدل رابطه‌ای، یا راه حل مناسبی نیافته‌اند و یا به عنوان مسئله‌ای فرعی و حاشیه‌ای مطرح بوده‌اند. خوانندگان محترم می‌توانند برای آگاهی بیشتر و عمیق‌تر از شیء‌گرایی و پایگاهداده شیء‌گرا و شیء-رابطه‌ای به منابع زیر مراجعه کنند:

[E.], [G. Booch ,1994] ، [R. Cooper 1997]، [Dietrich, Susan D. Urban, 2010]
[W. Kim, 1993 و Marcos, B. Vela, and J.M. Cavero, 2001]

۶-۵ ارتباط پایگاهداده با زبان‌های برنامه‌سازی

زبان بیانی SQL از طریق نرم‌افزارهای معروفی مانند JDBC و ODBC و Hibernate با زبان‌های برنامه‌سازی، ارتباط می‌یابد و کمبودهای آن جبران می‌شود [حق‌جو، ۱۳۹۲]. نرم‌افزار جدید و پرطرفداری برای ارتباط SQL با زبان‌های برنامه‌سازی به نام LINQ به بازار آمده است که قدرت برنامه‌سازی فوق‌العاده‌ای دارد. زبان (LINQ) Language از روی زبان SQL گرفته شده است و برای پاسخ به مسئله‌های پیچیده در دنیای برنامه‌سازی پیشرفتی دارد که مثلث راکس^۱ نامیده می‌شود. این روزها اکثر برنامه‌سازهایی که با زبان‌های شیء‌گرایی مثل سی‌شارپ یا جاوا کار می‌کنند، از دو ابزار مهم دیگر نیز برای ساختن برنامه‌های خود بهره می‌برند: یکی از این ابزارها

^۱ XML است. زبان XML دارای ویژگی‌های جالبی است که آن را به رایج‌ترین ابزار انتقال داده بین انواع مختلفی از برنامه‌های کاربردی تبدیل کرده است.

مشکل برنامه‌نویسان این است که هریک از این سه ابزار از منطق و مدل داده‌ای متفاوتی پیروی می‌کنند. برنامه‌نویس در حین کار روزمره خود مرتبًا نیاز دارد داده‌های خود را از یک مدل به مدل دیگر منتقل کند. مثلاً در یک برنامه مدیریت فروش محصولات، گاهی برنامه‌نویس لازم می‌داند داده‌های مشتریان را از پایگاهداده (ORACLE یا MySQL یا SQL Server) بخواند و سپس به فرمت XML تبدیل کند تا بتواند آن را از مجرای اینترنت (مثلاً از طریق وب‌سرویس‌ها) عبور دهد. در این صورت ناگزیر است همزمان داده‌ها را ابتدا با منطق SQL بخواند، سپس با منطقی شبیه ado.net پردازش کند و سپس به منطق xquery تبدیل کند.

اریک میر، در شرکت مايكروسافت این سه‌گانگی را مثلث راکس نامیده است که سرnam عبارات XML in Objects in business tier ، Relations in data tier و presentation tier است. منابع داده‌ای که LINQ می‌تواند از آن‌ها استفاده کند، می‌تواند یک شی ایجاد شده، یک فایل مستندات XML، یک بانک SQL Server و یا یک منبع دلخواه باشد. LINQ این توانایی را دارد که با تمامی این منابع داده کار کند. علاوه بر استخراج و اجرای پرسش بر روی منابع داده، به وسیله LINQ، امکان تغییر و دستکاری یک منبع داده مانند یک بانک SQL Server را نیز خواهد داشت. LINQ یک منطق است که درون کدهای برنامه به کار گرفته می‌شود و خیلی به منطق و زبان انسان شبیه است. به همین دلیل استفاده از آن باعث ساده شدن برنامه‌نویسی می‌شود.

خلاصه فصل ششم

درس پایگاهداده تا حد زیادی بر محور مدل رابطه‌ای می‌چرخد، اما دنیای واقعی از این مرحله فراتر رفته و در زمان ما به مدل شی – رابطه‌ای رسیده است. پرداختن به مدل رابطه‌ای از آن جهت لازم است که بدون آن نمی‌توان مدل شی – رابطه‌ای را فهمید. بنابراین باید دانشجویان عزیز با این مدل آشنا شوند تا در دروس ومطالعات آینده خود از ارزش‌های آن بی‌نصیب نمانند.

در این فصل ابتدا مدل رابطه‌ای نقد و نقاط ضعف و قوت آن بیان شد. سپس مفاهیم بنیادین مدل شیء - گرا ارائه شده و بر مبنای این دو مدل، مدل شیء- رابطه‌ای که البته مدل جدید و مستقلی است، ارائه شد. انواع دستاوردهای مدل شیء - رابطه‌ای مانند رابطه‌های تودرتو، بسته‌بندی در قالب جدول، ارث‌بری جداول از یکدیگر و موارد مشابه بیان شده‌است. در پایان این فصل به صورتی گذرا زبان SQL^۳ و نرم‌افزار LINQ معرفی و ارائه شده، با این امید که در درس بعدی (آزمایشگاه پایگاه‌داده‌ها) مورد استفاده قرار گیرد.

تمرین‌های تشریحی فصل ششم

۱. آیا نرم‌افزار DBMS برای داده‌های چند رسانه‌ای هم کارایی دارد؟
۲. مدل شیء - رابطه‌ای را نقد کنید. آیا می‌توان «رفتار» را به عنوان بخشی از یک جدول در نظر گرفت؟
۳. اگر مدل التقاطی شیء - رابطه‌ای نبود، چه می‌شد؟ آیا نمی‌توان به مدل شیء گرا اکتفا کرد و جدول را هم به عنوان شیء در نظر گرفت؟
۴. چرا مدل شیء - رابطه‌ای یک مدل مستقل است؟