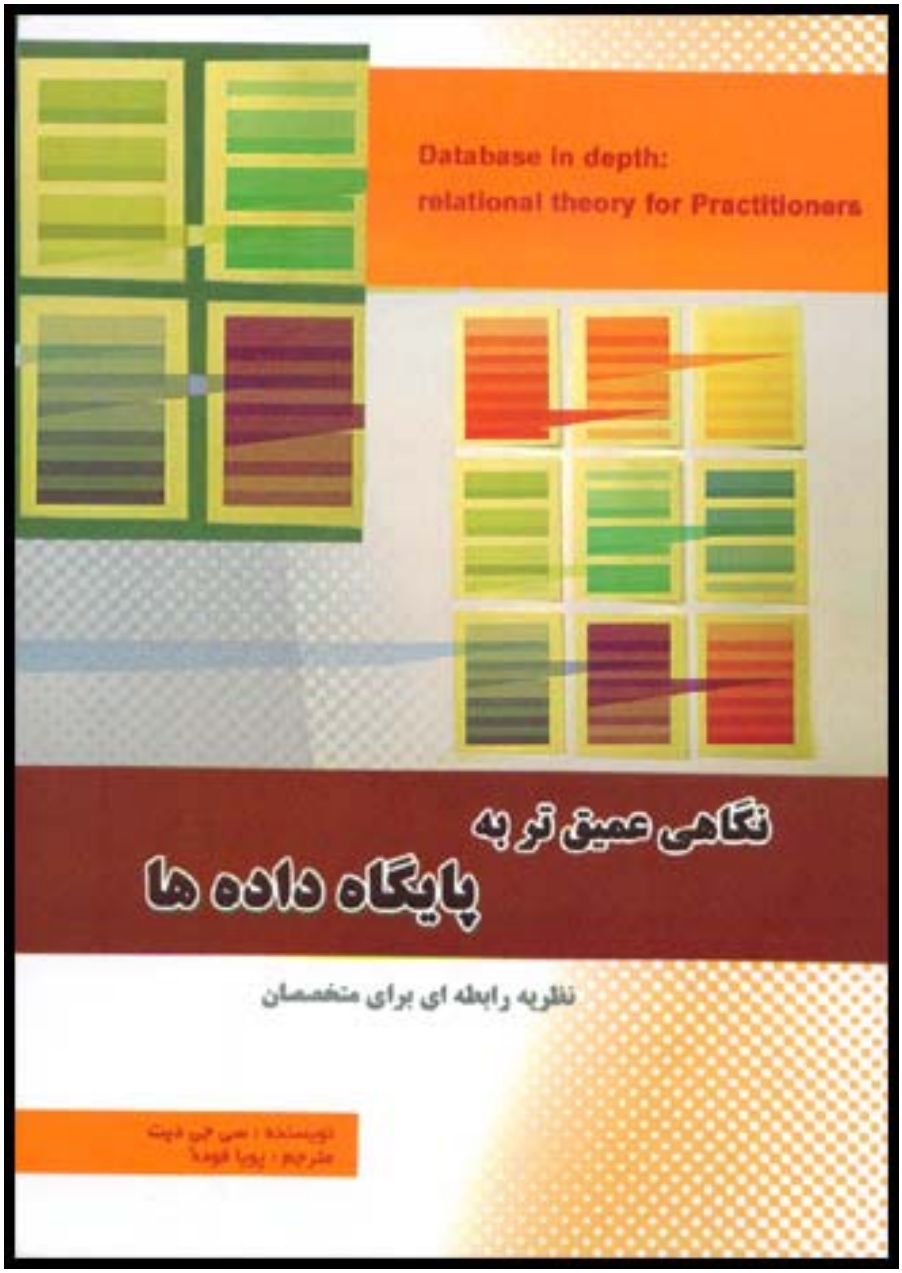

Database in depth:
relational theory for Practitioners
نگاهی عمیق تر به
پایگاه داده ها
نظریه رابطه ای برای متخصصان

نگاهی عمیق‌تر به پایگاه داده‌ها

نظریه رابطه‌ای برای متخصصین

نگارش نخست ۲۰۰۵

نویسنده: سی جی دیت

ترجمه : پویا فوده ۱۳۸۷

شابک: ۳-۴-۹۰۵۶۷-۶۰۰-۹۷۸

قیمت: ۴۸۰۰۰ ریال

ناشر: مهرگان قلم، نقش سیمرغ

پخش:

۱- نقش سیمرغ: خیابان انقلاب خیابان ۱۲ فروردین بن بست حقیقت شماره ۵۲
تلفن ۹-۶۶۴۹۴۵۹۸

۲- مهرگان قلم: خیابان انقلاب خیابان دانشگاه کوچه آشتیانی شماره۲۴ واحد ۳
تلفن ۶۶۹۶۸۹۷۰-۶۶۹۶۰۷۶۳

۳- انتشارات الیاس: خیابان انقلاب نبش ۱۲ فروردین شماره ۱۴۴۴
تلفن ۶۶۴۰۵۰۸۴

تقدیم به یاد و خاطره

ای.اف. کـــاد

کسی که شیفته انجام تجربه بدون پشتوانه تئوری است
مانند ناخدایی است که بدون سکان یا قطب‌نما
به کشتی می‌رود و هیچ‌گاه نمی‌تواند اطمینان
داشته باشد که به کجا می‌رسد.
تجربه همیشه بایستی بر پایه معلومات تئوری بنا شود.
لئوناردو داوینچی

مشکل با مردم این نیست که چیزی نمی‌دانند
بلکه مشکل این است که به اندازه کافی نمی‌دانند.
جاش بیلینگز

یکی چاره آورد از دل بجای
که بد ژرف بین او به تدبیر و رای
فردوسی

فهرست مطالب

دیباچه ترجمه فارسی

فن‌آوری پایگاه داده‌ها یکی از مهمترین و با سابقه‌ترین تکنیک‌هایی است که به صورت واقعی و جدی در ایران مورد استفاده قرار گرفته است و به جرات می‌توان اعلام داشت که بسیاری از دانش‌آموختگان مهندسی کامپیوتر، علوم کامپیوتر و حتی مهندسی صنایع از آن برای ایجاد سیستم‌های کاربردی استفاده می‌نمایند.

اگر چه بسیاری از پایگاه‌های داده تولید شده از کیفیت و کارایی نسبتا خوبی برخوردارند و توانسته‌اند بر بخشی از مشکلات موجود در صنایع و سازمان‌های خدماتی فائق آیند، ولی غالب پایگاه‌های تولید شده به علت نیاز ناگهانی و نبود متخصص با تجربه، از مبانی علمی و اصولی برخوردار نیستند.

این کتاب نوشته سی‌جی دیت و ترجمه آقای مهندس فودهٔ برای افرادی که با مبانی پایگاه داده‌ها آشنایی داشته و حتی چندین سیستم پایگاه داده طراحی و پیاده‌سازی نموده‌اند بسیار مفید خواهد بود. این کتاب به آنها نگاه جدیدی از مبانی پایگاه داده القا می‌نماید و شرایطی را به وجود می‌آورد که علم و تجارب به دست آمده را به چالش کشیده و تصحیح یا تکمیل نمایند.

گرچه این کتاب ترجمه است و مترجم برای حفظ امانت قسمتی از مطالب را آنچنان که در زبان فارسی متداول است نگارش نموده است، ولیکن اینجانب با تجربه نزدیک به ۲۰ سال تدریس، مشاوره و اجرای سیستم‌های پایگاه داده‌ها از مطالعه این کتاب اطلاعات خوبی به دست آوردم و لذت بردم.

دکتر محمد علی نعمت بخش
دانشگاه اصفهان – گروه کامپیوتر
خرداد ۱۳۸۷

پیشگفتار مترجم

امروزه و در کشور ما تمام کسانی که به نحوی با دانش نرم‌افزار و یا مدیریت نهادهای اداری و صنعتی سروکار دارند، اهمیت پایگاه داده‌ها را کاملا احساس می‌کنند. به ویژه فعالیت تقریبا تمامی افرادی که به برنامه نویسی مشغول‌اند، در زمینه پایگاه داده‌هاست. دلیل این وضعیت آن است که در شرایط فعلی و عدم وجود قانون کپی رایت برای نرم‌افزارهای خارجی و عدم اجرای دقیق آن برای نرم‌افزارهای ایرانی، تولید برنامه‌های کاربردی عمومی در عمل صرفه اقتصادی ندارد و تمام توان متخصین نرم‌افزار به سمت تولید سیستم‌های نرم‌افزاری اختصاصی و نیمه اختصاصی برای سازمان‌های فعال در صنعت و تجارت متمرکز گردیده است. بدیهی است هسته مرکزی چنین سیستم‌هایی پایگاه داده‌های آنهاست.

سازنده پایگاه‌داده‌ها بایستی به سه چیز مجهز باشد. هنر طراحی، تسلط عملی بر یک یا چند DBMS و دانش تئوری طراحی پایگاه داده‌ها. اولی امری ذاتی و خدادادی است که افراد مختلف کم و بیش از آن بهره‌مندند. دومی با تجربه و تمرین و احتمالا شرکت در دوره‌های آموزشی حاصل می‌شود و اما سومین مورد که به معنای تسلط به نظریه رابطه‌ای و شامل مباحثی علمی و بنیادی است. مطالبی که عمدتا بایستی در دانشگاه آموخته شوند. متاسفانه -یا خوشبختانه!- فقط کشور ما نیست که در این زمینه دارای ضعف است. سی‌جی‌دیت، نویسنده کتاب و از پیش‌کسوتان نظریه رابطه‌ای در مقدمه به این نکته اشاره می‌کند که مباحث نظریه‌ای رابطه توسط اکثریت جامعه بانک اطلاعاتی به خوبی فراگرفته نشده است ولی تنها با تکیه بر مهارت‌های عملی نمی‌توان پایگاه‌های داده با کیفیت و کارایی بالا طراحی کرد.

هدف از ترجمه این کتاب افزایش دانش تئوری مورد نیاز کسانی است که قصد دارند به طور عملی در زمینه پایگاه داده‌ها کار کنند. بنابراین کتاب حاضر برای کسانی که در زمینه نرم‌افزار فارغ‌التحصیل شده و هم اکنون در این زمینه مشغول کار هستند و جهت یا درس پایگاه‌داده‌های دانشجویان کارشناسی پیوسته نرم‌افزار و IT (پس از یادگیری مقدمات و زبان SQL)، کارشناسی ناپیوسته نرم‌افزار و یا به عنوان بخشی از مباحث کارشناسی ارشد نرم‌افزار، مفید است.

در پایان از تمام کسانی که مرا در انجام این کار یاری کردند تشکر می‌کنم خصوصا: همسرم که در تمام مراحل پشتیبان من بود و همچنین ویرایش نسخه نهایی را بر عهده داشت، آقای دکتر محمدعلی نعمت‌بخش که توصیه‌های مفیدی بر نسخه نهایی ارائه نمودند، دوستان عزیزم هادی صبوحی که ویرایش نسخه اولیه را بر عهده داشت و مهدی محمدی که درک بسیاری از مطالب کتاب مدیون تجربه‌های عملی است که حین کار دوشادوش با او به دست آورده‌ام.

پویا فودهٔ

رودهن، گروه کامپیوتر دانشگاه آزاد

اسفند ۱۳۸۶

دیباچه

من زمانی که برای نخستین بار کتابی از کریس دیت نخریدم را به خوبی به خاطر می‌آورم، درست خواندید گفتم «نخریدم». اواخر سال ۱۹۹۱ یا اوایل ۱۹۹۲ بود که در کتابفروشی کوچکی در میشیگان مال به دنبال کتابی در مورد SQL می‌گشتم تا اینکه کتابی پیدا کردم که نوشته سی‌جی‌دیت بود. تا آن زمان نام وی را نشنیده بودم ولی کتاب خوبی به نظر می‌آمد. داشتم در مورد خرید آن تصمیم می‌گرفتم که ..... وای قیمت کتاب! کتابی که در دستم بود حدود نیم سانتی متر ضخامت داشت و حجم آن کمتر از ۲۰۰ صفحه بود با این حال ۳۰ دلار قیمت داشت. قدری با خودم کلنجار رفتم سپس کتاب را سرجایش گذاشتم و به راهم ادامه دادم.

چه اشتباهی! هیچ گرانی بی علت نیست و من در آن زمان متوجه این موضوع نبودم.

در نهایت رئیسم را متقاعد کردم که شرکت باید کتاب را برایم بخرد و این کار حدود یک ماه طول کشید ولی بالاخره کتاب بدستم رسید و آن را خواندم. نه یک بار که چندین بار. یکی از بهترین و آموزنده ترین بخش‌های کتاب (برای من) ضمائم آن بود که در آن انتقادات کریس از زبان SQL رایج در آن زمان مطرح شده بود.

بوسیله این کتاب چیزهای زیادی از کریس یاد گرفتم.من مجذوب کار با بانک‌های اطلاعاتی و زبانهای اعلانی مانند SQL شدم: می‌توانستم نتایجی که بدنبال آن بودم را توصیف کنم تا موتور بانک اطلاعاتی آنها را برای من بدست آورد. علاوه بر این بیشتر با کریس و نقش وی در همکاری با کاد در آغار عصر رابطه‌ای آشنا شدم. سالهای متوالی مشترک مجله طراحی و برنامه نویسی پایگاه داده‌ها بودم فقط به خاطر اینکه کریس یک ستون ثابت در آن داشت.

اطلاعاتی که از طریق کریس درباره SQL بدست آوردم – شروع آن بوسیله کتاب نازک و گران که نسبت به خرید آن بی میل بودم- از سالها قبل تاکنون در زندگی من اثرگذار بوده است. آشنایی من با بانک‌های اطلاعاتی در آن زمان نقطه عطف بزرگی در زندگی من بود. دقت و شیوایی قلم کریس اثر عمیقی در دید من نسبت به SQL و بطور کلی بانک‌های اطلاعاتی داشت. از آن زمان SQL همواره موضوع مورد علاقه من برای آموختن و تالیف بوده است.

من از این فکر که اگر کتاب کریس به دست من نیافتاده بود امروز در چه حالی بودم به لرزه می‌افتم. شما اشتباه مرا تکرار نکنید و این کتاب را به قفسه کتابفروشی برنگردانید. آنرا بخوانید! از کسی که در اختراع و تدوین مدل رابطه‌ای همکاری کرده است چیزهایی بیاموزید که آینده کاری شما به آن بستگی دارد. ممکن است با همه آنچه کریس گفته است موافق نباشید —و نیازی هم نیست که باشید- ولی گفته‌های او را بفهمید. برای درک دید او از مدل رابطه‌ای وقت بگذارید. برای فهم نظرات منتشر شده از وی در مورد نحوه پیاده سازی‌های (بعضا غلط) این مدل در محصولات مختلف را درک کنید. این کارها را انجام دهید و خود را یک سر و گردن بالاتر از کسانی که برای یادگیری اصول بنیادی وقت نگذاشته‌اند ببینید.

اگر بخواهم کلامم را با ذکر نکته ای به پایان برم این نکته تاکید بر اهمیت شور و شوق برای آموختن است. برای من باعث افتخار بود که در کنار کریس دیت به ویرایش این کتاب پرداختم. من از صحبت با کریس و خواندن دست نوشته‌های وی چیزهای زیادی آموختم. ممکن است بخواهید بدانید که چگونه همکاری من با این کتاب شروع شد؟ تصور می‌کنم عامل این کار بطور غیر مستقیم مباحثی بود که با افراد هوشمند گروه اوراکل-ال در رابطه با بهینه سازی کوئری‌های SQL صورت می‌گرفت. ولیکن من تصور می‌کنم این همکاری ثمره مطالعه نوشته‌های کریس برای نخستین بار و در سالها پیش بود. حس کنجکاوی و عشق به آموختن مرا در کارم جدی کرد و از آن پس در مسیر پیشرفت قرار گرفتم و این عوامل می‌تواند در شما هم همین تاثیر را داشته باشد.

جاناتان جنیک

مانیسینگ، میشیگان

مارس ۲۰۰۵

پیش‌گفتار

پس از سال‌ها کار در بخش‌های مختلف جامعه بانک اطلاعاتی، احساس کردم که کتابی برای حرفه‌ای‌ها (و نه نوآموزها) مورد نیاز است که در آن اصول پایه نظریه رابطه‌ای – نه به گونه‌ای که توسط خصوصیات عجیب محصولات موجود، سمبل کاری‌های تجاری و استاندارد SQL لوث شده است- بیان شود. من این کتاب را برای پاسخگویی به این نیاز نوشتم. مخاطب اصلی من بانک اطلاعاتی کارها و یا افراد شاغل در این رشته هستند که به عدم درک تئوری زمینه تخصصی خود -آنطور که باید و شاید-، صادقانه اعتراف می‌کنند. درست است که این تئوری فقط مدل رابطه‌ای است و اصول بنیادی این نظریه ساده‌اند. ولی این حقیقت هم وجود دارد که این اصول بسیار گمراه‌کننده و سطحی‌نگرانه –و یا هردو حالت- آموزش داده شده‌اند به گونه‌ای که در بیشتر موارد چنین به نظر می‌رسد که اساسا درک نشده‌اند. برای مثال در اینجا چند سوال در این مورد مطرح شده است. شما جواب چند تا از آنها را می‌دانید؟

۱- فرم اول نرمال دقیقا یعنی چه؟

۲- رابطه‌ها و گزاره‌ها چه ارتباطی با همدیگر دارند؟

۳- بهینه‌سازی معنایی چیست؟

۴- وابستگی پیوندی چیست؟

۵- نیم تفاضل چه اهمیتی دارد؟

۶- چرا معلق‌کردن کنترل قواعد جامعیت قابل قبول نیست؟

۷- متغیر رابطه‌ای چیست؟

۸- تجزیه بدون کاستی یعنی چه؟

۹- آیا یک رابطه می‌تواند دارای ویژگی‌هایی باشد که مقادیر آنها خود یک رابطه باشد؟

۱۰- تفاوت بین مدل رابطه‌ای و SQL چیست؟

۱۱- چرا «قانون اطلاعات» مهم است؟

۱۲- چگونه XML با مدل رابطه‌ای مطابقت می‌کند؟

این کتاب پاسخگویی به سوالات فوق و بسیاری سوالات دیگر را امکان پذیر می‌کند و بطور کلی به بانک اطلاعاتی کارها را در درک عمیق تئوری رابطه‌ای و استفاده از آن در فعالیت‌های روزمره حرفه‌ای‌شان، یاری می‌نماید.

چه چیزی این کتاب را متمایز می‌کند؟

من در یک کلام می‌گویم که این کتاب ذاتا کتابی جدید است. من در کتاب‌های قبلی خود بارها گفته‌ام که فقط جستجویی در اطراف خود کرده و مطالبی را که نیاز به تکرارشان بوده است را گزینش نموده‌ام. ولی اکنون سعی کردم که آنها را به گونه‌ای متفاوت بیان کنم. ترتیبی متفاوت، به وجود آوردنی متفاوت، سبک و گفتاری متفاوت و مخاطبینی متفاوت (آخری از همه مهمتر است). ممکن است برخی از قسمتهای این کتاب در جاهای دیگری مطرح شده باشد ولی با این حال این کتاب بطور کلی کتابی جدید است. برخی قسمتهای کتاب به ناچار شبیه کتاب‌هایی است که قبلا نوشته‌ام چرا که همه آنها از یک منبع سرچشمه می‌گیرند: از مغز من و تجربیاتم و سمینارهایی که طی سالها برگزار کرده‌ام. این‌ها تقلید به حساب نمی‌آیند. به هر حال من آگاهانه از تعدادی از مثال‌های قدیمی دوباره استفاده کرده‌ام چرا که این مثال‌ها دقیقا همان نکاتی را که مایل به توضیح آنها هستم –نه کمتر و نه بیشتر– را نشان می‌دهند.

اجازه دهید که به موضوع مخاطبین این کتاب برگردیم. همانطور که اشاره شد من قبلا کتاب‌های زیادی در مورد تکنولوژی بانک‌های اطلاعاتی منتشر کرده‌ام. پس این کتاب چه تفاوتی با آنها دارد؟ آیا این کتاب با آنها رقابت می‌کند؟

به نظر من پاسخ سوال آخر«نه» است. من دو کتاب دیگر دارم که ممکن است در نگاه اول رقیب این کتاب به نظر برسند.

- آشنایی با پایگاه داده‌ها. ویرایش هشتم ۲۰۰۴
- پایگاه داده‌ها، انواع آن و مدل رابطه‌ای. ویرایش سوم ۲۰۰۶

کتاب اول عملا تمامی مباحث مربوط به بانک اطلاعاتی، و نه فقط مدل رابطه‌ای را پوشش می‌دهد. بدین معنا که در درجه اول کتابی دانشگاهی است و فرض را بر این

می‌گذارد که خواننده هیچ آگاهی و تجربه‌ای در مورد بانک اطلاعاتی ندارد. همچنین متن آن رسمی‌تر از این کتاب است همانطور که باید برازنده یک کتاب درسی باشد.

کتاب دوم اجرای مجددی است از کتاب قبلی من و هیوج داروین بعنوان «بنیانی برای آینده پایگاه داده‌ها» است. این کتاب متن درسی پیشرفته‌ای است که از کتاب قبلی رسمی‌تر است. با وجود اینکه این کتاب‌ها از نظر موضوع اشتراک‌هایی با همدیگر دارند من حقیقتا رقابتی بین این سه کتاب نمی‌بینم.

تفاوت مهم دیگر این کتاب جنبه خودآموز بودن آن است (البته می‌توان در مباحثات دانشگاهی هم از آن استفاده کرد). برای کمک به فهم مطالب تمرین‌های زیادی در کتاب وجود دارد. گرچه انجام آنها الزامی نیست با این حال من فکر می‌کنم که حل کردن –حداقل برخی از– آنها کار مفیدی است.

برای برطرف کردن شبهه رقابت بین کتاب‌هایم در مورد دو کتاب دیگر هم توضیح می‌دهم:

- بانک اطلاعاتی و مدل رابطه‌ای: بازبینی و تجزیه و تحلیل ۲۰۰۱
- داده‌های گذرا و مدل رابطه‌ای ۲۰۰۳

به نظر من کتاب اول مکمل این کتاب است و در آن مقاله‌های تدکاد -که طی آن مدل رابطه‌ای را برای نخستین بار به جهانیان معرفی کرد- را به گونه‌ای بی‌تکلف بازنگری کرده‌ام. در کتاب دوم –همانطور که از نام آن مشخص است– به نظریه رابطه‌ای نمی‌پردازد و فقط به یکی از شاخه‌های این نظریه مربوط می‌شود. با این حال ممکن است چنین به نظر برسد که فصل اول این کتاب که مروری کلی بر نظریه رابطه‌ای دارد، با این کتاب تا حدودی شباهت دارد ولیکن من واقعا چنین تصوری ندارم.

چکیده تمامی مطالب فوق این است که : با وجود اینکه من قبلا در مورد این موضوعات مطالب زیادی نوشته‌ام که برخی مطالب آنها به صورت اجتناب ناپذیری با هم مشابه بوده اند ولیکن تصور نمی‌کنم که هیچکدام از کتاب‌های دیگر من –و تا آنجا که می‌دانم هیچ کس دیگر– مطالب این کتاب را در یک جا و با این روش پوشش داده باشد.

مقدمات دیگر

من بایستی نکات دیگری را هم متذکر شوم. اولا همانطور که گفتم من از مثال‌های کتابها و مقالات قبلی خودم - خصوصا مثال اجرایی معروف بانک اطلاعاتی توزیع‌کنندگان و قطعات -در این کتاب هم استفاده کرده‌ام. من از اینکه این دایناسور را یک بار دیگر به دنبال خودم می‌کشم عذر می‌خواهم ولیکن توجه داشته باشید که من در ابتدا مثال‌ها را به دقت طراحی کرده‌ام تا دقیقا نکاتی را که قصد بیان آنها را دارم را روشن کنند.

ثانیا، درک من از مدل رابطه‌ای طی سال‌ها افزایش یافته است و این روند هم اکنون نیز ادامه دارد. این کتاب آخرین اندیشه‌های من در ارتباط با این موضوع را نشان می‌دهد بنابراین اگر بین این کتاب و مطالب قبلی من تناقضی مشاهده کردید (که البته میزان آن بسیار اندک است)، مطالب این کتاب جانشین کتاب‌های خواهد بود و به هر حال من تاکید میکنم چنین اختلاف‌هایی اکثرا بسیار جزیی هستند. من همیشه سعی کرده ام که –در صورت احساس نیاز- جزء اولین کسانی باشم که مطالب جدید را بیان می‌کنند.

ثالثا، من قصد دارم مطالب را بصورت تئوری بیان کنم. ولی به این جمله اعتقاد دارم که «تئوری همیشه به کار می‌آید». این نکته را به این علت بیان کردم که بسیاری از مردم با آن مخالف‌اند و تصور می‌کنند اگر چیزی تئوری شد، دیگر نمی‌تواند کاربردی باشد. ولی حقیقت اینست که تئوری –حداقل تئوری که من در موردش صحبت می‌کنم یعنی تئوری رابطه‌ای- بسیار کاربردی است. هدف این تئوری تنها این نیست که یادگرفته شود. بلکه هدف این است که ما را قادر به ساخت سیستم‌هایی صددرصد کاربردی نماید. ایجاد هر یک از قسمت‌های این تئوری دارای دلایل محکم کاربردی است. به راستی بخش عمده این تئوری نه تنها کاربردی است که بنیادی، سهل، ساده، سودمند و احتمالا لذت بخش هم هست. (امیدوارم این موضوع در هنگام مطالعه کتاب ثابت شود.)

(در حقیقت ما نباید به دنبال چیزی فراتر از تئوری رابطه‌ای باشیم. این نظر که تئوری کاربردی است آن‌قدر بدیهی است که نیازی به دفاع ندارد. برای نمونه یک صنعت میلیارد دلاری فقط می‌تواند بر پایه یک اندیشه تئوریک بنا شود. یک فرد بدبین ممکن

است بگوید «بله، ولی تئوری تا به حال چه فایده‌ای برای من داشته است؟» می‌توان گفت که اگر اهمیت تئوری را دریابیم، آنگاه می‌توانیم همواره خود را در مقابل به منتقدین برحق بدانیم. این هم دلیل دیگری است که بر اساس آن من فکر می‌کنم کتاب‌هایی از این دست مورد نیاز هستند.)

و نکته‌ای دیگر: زبان استاندارد «رابطه‌ای» SQL است و من فرض کرده‌ام که شما با این زبان و همچنین مفاهیم کلی در مورد بانک اطلاعاتی آشنا هستید. با این حال - همانطور که بزودی خواهید دید- من با صراحت از SQL انتقاد کرده‌ام. حقیقت تاسف بار این است که SQL در بسیاری موارد نمی‌تواند به خوبی از مدل رابطه‌ای پشتیبانی کند. این زبان از نظر ناقص بودن دستورات و نحوه اجرای آنها دارای مشکلات بسیاری است.(برای تاکید بر این موضوع، «رابطه‌ای» را داخل کوتیشن قرار دادم.) این فقط یکی از دلایلی است که برای یادگیری مدل رابطه‌ای وجود دارد. چرا که پشتیبانی SQL از این مدل بسیار ناقص و نارسا است. این زبان مانند طنابی در دست شماست که می‌توانید با آن خود را دار بزنید. پس برای اینکه چنین اتفاقی برایتان نیافتد نیاز به تئوری دارید. شما نیازمند تئوری هستید تا خود را مقید به مقرراتی کنید که بایستی SQL شما را وادار به رعایت آنها می‌کرد، ولی این کار را نکرده است. تکرار سطرها در این مورد مثال خوبی است. این کار در مدل رابطه‌ای غیر مجاز است ولی SQL آن را مجاز می‌داند. شما باید بدانید چرا این کار در تئوری ممکن نیست تا بدانید چرا نباید از این «ویژگی» موجود در SQL «بهره‌برداری» کنید. استفان فارولت بعنوان خواننده و منتقد این کتاب چنین نوشت: «اگر به اندازه ذره‌ای تجربه داشته باشید، به خوبی درک می‌کنید که هیچ راهی برای فرار از تئوری وجود ندارد.»

و یک نکته دیگر در مورد SQL : در هر کجای این کتاب که من این کلمه را به کار برده‌ام منظورم فقط SQL استاندارد است و نه لهجه‌های مختلف این زبان که هر یک متعلق به محصول خاصی می‌باشد (بجز مواردی که بطور صریح خلاف این مطلب عنوان شده باشد) و خصوصا زمانی که کلمه «اس-کیو-ال» را به کار می‌برم منظورم «sequel» نیست.

در پایان توجه شما را به کنفرانس‌هایی که در مورد مطالب بیان شده در این کتاب ارائه نموده‌ام جلب می‌کنم. برای اطلاعات بیشتر به آدرس http://www.dbdebunk.com و یا http://www.thethirdmanifesto.com مراجعه نمایید.

پاسخ به سوالات

شما می‌توانید برای عضویت در لیست پستی ما و یا درخواست کاتالوگ با ما به نشانی زیر مکاتبه کنید:

info@oreilly.com

همچنین برای طرح سوالات فنی و یا اظهار نظر در مورد مطالب کتاب به آدرس زیر نامه بنویسید:

bookquestions@oreilly.com

ما وب سایتی برای این کتاب داریم که می‌توانید مثال‌های بیشتر و غلط نامه کتاب را در آن مشاهده کنید. آدرس این سایت عبارت است از:

http://www.oreilly.com/catalog/databaseid

قدردانی

در اینجا از تمام کسانی که بطور مستقیم و غیر مستقیم در بوجود آمدن این کتاب دخالت داشتند تشکر می‌کنم. در اینجا نام بازبینان این کتاب را ذکر می‌کنم. استفان فارولت، جاناتن جنیک، لکس‌هان، آناتونی‌مولینارو، پیترابسون و میشل‌ونر. که شرح‌های سودمندی برای دست‌نوشته‌های من نوشتند. همچنین از ناگراج آلور و هیوج داروین بخاطر مباحثات فنی متعددی که به من داشتند تشکر می‌کنم. همچنین از همسرم لیندی بخاطر حمایت و پشتیبانی‌اش در هنگام انجام این کار -و همچنین تمامی پروژه‌های بانک اطلاعاتی طی سالیان متوالی- قدردانی می‌کنم. از تمامی افراد انتشارات اوریلی خصوصا جاناتان جنیک و ژنویو انترمونت بخاطر دلگرمی و همیاری و پشتیبانی‌شان در تمامی زمان نوشتن این کتاب سپاسگزارم.

سی جی دیت
هلدبورگ، کالیفرنیا، ۲۰۰۵

فصل یک

مقدمه

متخصصین – با هر انضباط و تفکری – بایستی اصول رشته کاری خود را بدانند. بنابراین اگر شما یک متخصص بانک اطلاعاتی هستید باید مدل رابطه‌ای را بشناسید چرا که این مدل اصل (و یا بخش عمده) بانک اطلاعاتی است. امروزه مدل رابطه‌ای در هر دوره دانشگاهی و یا تجاری پایگاه‌داده‌ها –حداقل بصورت سطحی- تدریس می‌شود ولی این کار در اکثر موارد این امر به خوبی انجام نشده است و نتیجه مطلوبی از این تدریس حاصل نمی‌شود. متاسفانه مدل رابطه‌ای توسط عموم افرادی که با بانک اطلاعاتی سر و کار دارند، به خوبی درک نشده است. برخی دلایل این امر عبارتند از:

- فهمیدن این مدل مانند یادگیری در خلاء است. درک رابطه بین اشیا – حداقل برای تازه کارها – و همچنین شناسایی مسائل و راه حل آنها، مشکل است.
- خود مدرسین نیز بطور کامل این مفاهیم را درک نکرده‌اند.
- اکثرا در افراد تجربی- این مدل اصلا تدریس نشده است و بجای آن زبان SQL و یا برخی لهجه‌های آن مانند لهجه اوراکل تدریس شده است.

بنابراین کتاب حاضر برای بانک اطلاعاتی کارهای حرفه‌ای و خصوصا افراد تجربی است، که با این مدل آشنایی دارند ولی دانش آنها کمتر از چیزی است که باید باشد. با این وجود این کتاب برای شروع کار کسانی که هیچ چیز از بانک اطلاعاتی نمی‌دانند مناسب نیست. من می‌دانم که شما چیزهایی در مورد SQL می‌دانید. ولی اگر – اگر در اینجا لحن صحبتم کمی تند است ببخشید – اگر آگاهی شما از مدل رابطه‌ای تماما از SQL ناشی شده است، متاسفانه شما مدل رابطه‌ای را آنطور که باید و شاید نمی‌شناسید و در زمره افرادی که «به اندازه کافی نمی‌دانند» قرار دارید.

توجه

SQL ≠ مدل رابطه‌ای

در اینجا برخی از مواردی که SQL به وضوح آنها را روشن نکرده است را بیان کرده ام (و تعداد بیشماری از این موارد وجود دارد):

- بانک‌های اطلاعاتی، رابطه‌ها و تاپل‌ها واقعا چه هستند
- تفاوت بین نوع‌ها و رابطه‌ها
- تفاوت بین مقادیر رابطه‌ای و متغیرهای رابطه‌ای
- رابطه بین گزاره‌ها و استنتاج‌ها
- صحت ویژگی‌های رابطه‌ای
- نقش حیاتی قیود جامعیت

تمامی موارد فوق و بسیاری موارد دیگر در این کتاب عنوان شده اند.

دوباره تاکید می‌کنم: اگر آگاهی شما از مدل رابطه‌ای فقط از SQL ناشی شده است، در زمره افرادی که «به اندازه کافی نمی‌دانند» قرار دارید. یکی از پیامدهای ناپسند این امر این است که شما مجبورید برخی مطالب را از نو بیاموزید و فراموش‌کردن و بازآموزی مطالبی که قبلا به اشتباه یاد گرفته‌اید، کاری دشوار است. یک نکته دیگر... خواهش می‌کنم به این دلیل که تصور می‌کنید به طور کامل با مبحثی آشنایی دارید، از مطالعه آن صرفنظر نکنید. برای مثال مطمئن هستید که می‌دانید که کلید چیست. و عبارات رابطه‌ای. و یا پیوند.

درباره اصطلاحات فنی

با دیدن لیست مباحثی رابطه‌ای که در قسمت قبل مطرح کردن احتمالا بلافاصله متوجه شده‌اید که از کلمات رسمی مانند رابطه، تاپل* و ویژگی استفاده کرده‌ام. SQL از اینگونه عبارات استفاده نمی‌کند و بجای آن عبارات دوستانه‌تری مانند جدول، سطر و ستون را به کار می‌برد. من به طور کل با این نظر که عبارات دوستانه، مطالب را

* بر وزن قابل

پذیرفتنی‌تر می‌کند موافقم. ولی در این مورد به نظر می‌رسد که این عبارات ساده مطلب را مطبوع‌تر نمی‌کند بلکه آنرا تحریف می‌کنند و در واقع باعث نفهمیدن اصل مطلب می‌شوند. واقعیت این است که رابطه همان جدول نیست، تاپل سطر نیست و ویژگی هم ستون نیست. و فقط می‌توانند در متون غیر رسمی خود را بجای آنها جا بزنند – خود من هم در بسیاری کتاب‌ها این کار را کرده ام- می‌خواهم بگویم که عبارات دوستانه فقط به واقعیت شباهت دارند و از بیان ذات آنچه به دنبال آن هستید، ناتوانند. به زبان دیگر: اگر شما کیفیت چیزی را به درستی فهمیده باشید، آنگاه مجاز خواهید بود که برای نامیدن آن از عبارات ساده استفاده کنید ولی برای موضوعی که برای نخستین بار قصد درگیر شدن با آنرا دارید، واقعا لازم است که از اصطلاحات رسمی شروع کنید. بنابراین در این کتاب در اکثر موارد من از اصطلاحات رسمی استفاده کرده‌ام و البته در موقع لزوم آنها را بطور کامل توضیح داده‌ام.(البته نه در فصل یک کتاب)

و نکته‌ای دیگر در مورد اصطلاحات فنی: گفته شد که SQL سعی دارد که مجموعه‌ای از عبارات را ساده‌تر کند. و در اینجا باید بگویم که در نهایت کار را پیچیده‌تر می‌کند. برای نمونه کلمات اپراتور، تابع، پروسیجر، روتین و متد همگی در اصل به یک موضوع اشاره می‌کنند (با تفاوت‌های بسیار جزئی) و من در این کتاب برای همگی از کلمه اپراتور استفاده کرده‌ام.

و نکته‌ای دیگر در مورد SQL ؛ منظورم از این کلمه فقط SQL استاندارد است و نه لهجه‌های مختلف این زبان که هر یک متعلق به محصول خاصی می‌باشد بجز مواردی که بطور صریح خلاف این مطلب را عنوان کرده‌ام.(این نکته را در پیشگفتار کتاب هم آورده ام ولی می‌دانم که اکثر مردم پیشگفتار کتاب را نمی‌خوانند). انتقادات من هم به همان SQL استاندارد برمی‌گردد. بنابراین اگر انتقادی مطرح گردیده به محصول پر طرفدار شما که خوشبختانه شنیده‌ام بسیار خوب و عالی است مربوط نمی‌شود.

قوانین، نه محصولات

سوال مهمی که ممکن است از من بپرسید این است که چرا مدعی هستم که شما به عنوان متخصص بانک اطلاعاتی بایستی با مدل رابطه‌ای آشنایی داشته باشید. پاسخ این

است که مدل رابطه‌ای وابسته به محصول نیست بلکه وابسته به قوانین است و اگر بخواهید معنی قواعد را بدانید به شرح زیر است(به نقل از لغت نامه چیمبر قرن بیستم):

قانون: ماخذ، ریشه، اصل، اصل بنیادی، اصل طبیعی، پایه تئوری، واقعیت بنیادی که دیگران آنرا کشف کرده‌اند و یا بدیهی باشد.

نکته این است که قوانین پابرجا هستند و محصولات و فن‌آوری‌ها همواره در حال تغییرند (و SQL هم از این امر مستثنا نیست). برای مثال فرض کنید که شما اوراکل را می‌شناسید و یا اصلا در اوراکل خبره هستید. با این حال دانش شما به اوراکل محدود می‌شود و قابل انتقال به محیط DB2 و یا SQLServer نیست.(و ممکن است مانع پیشرفت شما در شناخت محیط جدید هم باشد.) ولی اگر شما قوانین اساسی – و به زبان دیگر مدل رابطه‌ای- را بدانید، دانش و مهارتی دارید که قابل انتقال است: دانش و مهارتی که شما را قادر به کار در هر محیطی می‌کند و هرگز قدیمی نمی‌شود.

بنابراین در کتاب حاضر قوانین اهمیت دارند و نه محصولات. اصول بنیادی موضوع بحث است و نه مدهای زودگذر. البته من می‌دانم که در دنیای واقعی گاه شما مجبور به مصالحه هستید. مثلا برخی اوقات ممکن است دلایلی عملی برای طراحی بانک اطلاعاتی بصورت غیر بهینه وجود داشته باشد. (این موضوع در فصل ۷ مورد بحث قرار گرفته است.) در مورد SQL هم می‌توان چنین شرایطی را در نظر داشت. گر چه استفاده از SQL بصورت رابطه‌ای قطعا امکان پذیر است (در برخی اجزا و به هر قیمتی)، ولی در برخی موارد به این نتیجه می‌رسید که – به این دلیل که پیاده‌سازی موجود بسیار ناقص است- چنین کاری بهره‌وری را پایین خواهد آورد. در چنین مواردی ممکن است برخی قسمت‌ها را از «کاملا رابطه‌ای» بودن معاف کنید. (مثلا با نوشتن یک کوئری عجیب و غیر طبیعی برای پیاده‌سازی استفاده از یک ایندکس). به هر صورت من اعتقاد راسخ دارم که شما باید چنین مصالحه‌ای را از منظر قوانین به انجام رسانید.

- شما باید بدانید که در زمان مصالحه، دارید چه کاری انجام می‌دهید.
- شما باید رفتار صحیح و قانونی بشناسید و دلیل خوبی برای ترک آن داشته باشید.

- شما باید دلایل خود را مستند کنید، چرا که اگر در آینده قصد بهم زدن این مصالحه را داشته باشید - مثلا اگر محصول بعدی در همین رابطه ابزارهای بیشتری داشت- این کار انجام‌پذیر باشد.

جمله زیر – که منسوب به لئوناردو داوینچی است و ۵۰۰ سال قبل بیان شده است!- فوت و فن دریا نوردی را بیان می‌کند:

کسی که شیفته انجام تجربه بدون پشتوانه تئوری است مانند ناخدایی است که بدون سکان یا قطب‌نما به کشتی می‌رود و هیچ‌گاه نمی‌تواند اطمینان داشته باشد که به کجا می‌رسد. تجربه بایستی همیشه بر پایه معلومات تئوری بنا شود.

مروری بر مدل اصلی

شما در زمینه بانک اطلاعاتی کار می‌کنید، بنابراین تا حدودی با مدل رابطه‌ای آشنایی دارید. هدف از این بخش شروع صحبت در مورد مباحث آتی است و مروری بر برخی از جنبه‌های ابتدایی این مدل –که بصورت اصلی تدوین شده است- خواهیم داشت. به عبارت «بصورت اصلی تدوین شده است» توجه کنید! یک تصور غلط در مورد مدل رابطه‌ای اینست که آنرا کاملا ایستا فرض می‌کنند در حالی که چنین نیست. مدل رابطه‌ای از این نظر شبیه ریاضیات است: ریاضیات هم ایستا نیست و در گذر زمان تغییر پیدا می‌کند. در حقیقت مدل رابطه‌ای می‌تواند بعنوان شاخه کوچکی از ریاضیات در نظر گرفته شود. در طی زمان همواره قضایای جدید اثبات و نتایج جدید کشف می‌شوند و همیشه افراد شایسته در این روند مشارکت دارند. و باز هم مشابه ریاضیات، مدل رابطه‌ای نیز توسط فردی ابداع شد که از زمره مردان کوشای این جهان بود.

شاید نام وی را ندانید. باید بگویم که وی ای‌اف کاد * نام داشت و در آن زمان یکی از محققین موسسه IBM بود. در اواخر سال ۱۹۶۸ کاد ریاضیدان تجربی برای نخستین بار دریافت که قواعد ریاضی می‌توانند در تدوین قوانین مربوط به پایگاه داده‌ها

* ای برای ادگار و اف برای فرانک، ولی او همیشه با نام مستعار امضا می‌کرد. برای دوستانش و از جمله خود من او تد بود.

–که در آن زمان بسیار ناقص بود- به‌کار گرفته شوند. تدوین اصلی این مدل در گزارش تحقیقاتی IBM در سال ۱۹۶۹ نمودار گشت و من مطالب بیشتری در مورد این مقاله در ضمیمه ب عنوان نموده‌ام.

ویژگی‌های ساختاری

مدل اصلی دارای سه جز مهم است. ساختار، جامعیت و کار با داده‌ها. من بطور خلاصه هریک از آنهارا بصورت اجمالی توضیح می‌دهم و در فصل‌های آینده بطور کامل در مورد آنها بحث خواهم کرد.

اول ساختار. اصلی ترین ساختار خود رابطه است و همانطور که می‌دانید رابطه‌ها را بصورت جدول بر روی کاغذ نمایش می‌دهند (شکل ۱-۱ را بعنوان یک مثال بدون نیاز به توضیح ببینید). رابطه‌ها بر روی انواع تعریف می‌شوند.(که بعنوان دامنه‌ها هم شناخته می‌شوند). یک نوع، استخری خیالی از مقادیر مختلف است که ویژگی‌های واقعی در رابطه‌های واقعی، مقادیر واقعی خود را از آن برمی‌دارند. با مراجعه به مثال بانک اطلاعاتی کارمندان و شعب که در شکل ۱-۱ نمایش داده شده است، مثلا نوعی که DNO («کد شعبه») نام دارد مجموعه‌ای از تمامی کدهای معتبر برای شعب می‌باشد. ویژگی DNO در رابطه DEPT و همچنین ویژگی DNO در رابطه EMP می‌توانند شامل مقادیری باشند که از استخری خیالی برداشته شده‌اند.(ولی لازم نیست نام این ویژگی‌ها مانند نوعشان با هم یکی باشد و معمولا هم چنین نیست. در آینده مثال‌های زیادی در این مورد خواهیم دید.)

DEPT

DNO	DNAME	BUDGET
D1	Marketing	10M
D2	Development	12M
D3	Research	5M

EMP

ENO	ENAME	DNO	SALARY
E1	Lopez	D1	40K
E2	Cheng	D1	42K
E3	Finzi	D2	30K
E4	Saito	D2	35K

DEPT.DNO *referenced by* EMP.DNO

شکل ۱-۱. بانک اطلاعاتی شعب و کارمندان – مقادیر نمونه

همان طور که قبلا گفتم، جدول‌هایی مانند دو جدول شکل ۱-۱ رابطه‌ها را به تصویر می‌کشند. یک جدول n تایی می‌تواند با جدولی n ستونی نمایش داده شود. ستون‌های جدول با ویژگی‌ها رابطه و همچنین سطرها با تاپل‌ها معادل‌اند. n می‌تواند یک مقدار صحیح و غیر منفی باشد. یک رابطه یکتایی یونری، رابطه دوتایی باینری و رابطه سه تایی ترنری نامیده می‌شوند. مدل رابطه‌ای انواع گوناگون کلید را پشتیبانی می‌کند. هر رابطه دارای حداقل یک کلید کاندید است.* کلید کاندید فقط یک شناسه یکتا است. به زبان دیگر ترکیبی است از ویژگی‌ها –اکثرا و نه همیشه، این «ترکیب» فقط یک ویژگی را در برمی‌گیرد- که هر تاپل رابطه در ترکیب فوق الذکر دارای مقداری یکتا است. در شکل ۱-۱ برای مثال هر شعبه دارای یک کد یکتا و هر کارمند دارای یک شماره پرسنلی یکتا است. بنابراین می‌توانیم بگوییم که {DNO} یک کلید کاندید برای DEPT و {ENO} یک کلید کاندید برای EMP است. در مورد آکولاد: کلیدهای کاندید همیشه ترکیب‌ها، یا مجموعه‌هایی از ویژگی‌ها هستند-حتی زمانی که فقط یک ویژگی را شامل شود- و بصورت عرفی اعضای یک مجموعه را بین آکولاد قرار می‌دهند.

و موضوع بعد کلید اصلی، کلید کاندیدی است که به دلیل دارا بودن برخی مشخصات خاص، برای این منظور انتخاب شده است. بدیهی است که اگر رابطه فقط یک کلید کاندید داشته باشد، می‌توان به راحتی آنرا کلید اصلی نامید. ولی اگر رابطه دارای دو –ویا بیشتر- کلید کاندید باشد برای انتخاب یکی از آنها بعنوان کلید اصلی مختار خواهیم بود. به این معنا که بالاخره یکی از آنها «از بقیه برابرتر است». برای مثال فرض کنید که یک کارمند دارای یک شماره کارمندی یکتا و یک نام یکتا است. در این صورت {ENO} و {ENAME} هر دو کلید کاندید EMP خواهند بود و ممکن است که ما {EMP} را بعنوان کلید اصلی تعیین کنیم.

* به زبان دقیق این جمله باید به این صورت گفته شود:«هر رلور دارای حداقل یک کلید کاندید است» (قسمت رابطه‌ها در مقابل رلورها را ببینید).

توجه کنید که گفتم در انتخاب کلید اصلی آزاد هستید ولی اگر فقط یک کلید کاندید موجود باشد دیگر نه اختیاری وجود دارد و نه مشکلی. اما اگر بیش از یک کلید کاندید وجود داشته باشد، انتخاب از بین آنها تا حدودی حالت دلبخواه دارد. در بعضی موقعیت‌ها به نظر می‌رسد که هیچ دلیل منطقی برای انتخاب وجود ندارد. در این کتاب من معمولا از قاعده کلید اصلی پیروی می‌کنم. در تصاویری مانند شکل ۱-۱ من ویژگی کلید اصلی را با دو خط در زیر آن متمایز می‌کنم. بر این نکته تاکید می‌کنم که کلید کاندید همان کلید اصلی نیست و این موضوع از نقطه نظر رابطه‌ای اهمیت فراوان دارد. از این به بعد در برخی موارد من از کلمه کلید استفاده کرده‌ام که منظور از آن کلید کاندید است.(در مواردی ممکن است متعجب شوید که کلید اصلی از کدام «مشخصات خاص» در برابر سایر کلیدهای کاندید بهره‌مند بوده است؟ این مشخصه معمولا به ذات ویژگی انتخاب شده باز می‌گردد. بگذریم، این موضوع چندان مهم نیست.)

و در نهایت کلید خارجی، مجموعه‌ای از ویژگی‌های یک رابطه است که لازم است با مقادیر کلید کاندید یک رابطه دیگر (و یا شاید همان رابطه) مطابقت داشته باشد. با مراجعه به شکل ۱-۱ مثلا {DNO} کلید خارجی EMP است و لازم است مقادیر آن با کلید کاندید {DNO} در رابطه DEPT مطابقت داشته باشد. (برای مشخص شدن این موضوع، آنها را در شکل با فلش به هم مرتبط کرده‌ام). منظور از لزوم مطابقت این است که اگر مثلا EMP دارای تاپلی با مقدار D2 برای DNO باشد آنگاه DEPT هم می‌تواند دارای تاپلی باشد که مقدار DNO آن D2 است در غیر اینصورت EMP دارای کارمندی خواهد بود که وجود خارجی ندارد و بانک اطلاعاتی هم «نمونه قابل اعتمادی از واقعیت» نخواهد بود.

ویژگی‌های جامعیت

قید جامعیت در اصل فقط عبارتی منطقی است که باید همیشه درست باشد. مثلا در مورد شعب و کارمندان می‌توان یک قید عملی در مورد SALARY داشت بدین صورت که مقدار آن همیشه بزرگتر از صفر باشد. هر بانک اطلاعاتی دارای قیودی

است ولی این قیود لزوما با توجه به رابطه‌های یک بانک اطلاعاتی خاص تدوین شده و ویژه همان بانک اطلاعاتی خواهند بود. در مقابل مدل رابطه‌ای (حداقل بصورت سنتی) دارای دو قانون جامعیت عمومی است. عمومی بدین معنا که بر هر بانک اطلاعاتی اعمال می‌شود. یکی از آنها در مورد کلید اصلی است و دیگری در مورد کلید خارجی.

جامعیت وجودی

کلید اصلی نمی‌تواند تهی باشد.

جامعیت ارجاعی

نبایستی هیچ مقدار غیر منطبقی در کلید خارجی وجود داشته باشد.

اجازه دهید تا اول دومی را توضیح دهم: منظورم از عبارت «کلید خارجی غیر منطبق» ، مقدار کلید خارجی است که مقدار معادل آن در کلید کاندید متناظر وجود نداشته باشد. در مثال شعب و کارمندان، اگر EMP دارای تاپلی باشد که مقدار DNO آن D2 است ولی تاپل متناظر آن در DEPT وجود نداشته باشد، جامعیت ارجاعی نقض شده است. بنابراین قانون جامعیت ارجاعی فقط بر پیاده‌سازی صحیح کلید خارجی تاکید می‌کند و عنوان جامعیت ارجاعی از این واقعیت ناشی می‌شود که هر مقدار کلید خارجی بعنوان مرجع به تاپلی با مقدار مساوی در کلید کاندید متناظر وابسته است. در واقع این قانون می‌گوید:«اگر B به A ارجاع داشته باشد، آنگاه A باید حتما وجود داشته باشد.»

در مورد قانون جامعیت وجودی، در اینجا من مشکلی دارم. من موضوع «تهی» را بطور کلی رد می‌کنم و شدیدا معتقدم که تهی جایی در مدل رابطه‌ای ندارد.(کاد خلاف این نظر را داشت اما من برای این حرف دلایل محکمی دارم.) برای تشریح قاعده جامعیت وجودی من مجبورم نظر خود را موقتا کنار بگذارم ولی خواهشمندم مطالبی را که در این مورد در فصل۳ بیان کرده ام را درک کنید.

«نشانگر» تهی ذاتا به معنای مقدار نامعلوم است. (نکته بسیار مهم: تهی یک مقدار نیست بلکه یک نشانگر و یا پرچم است.) برای مثال فرض کنید که میزان حقوق کارمند E2 را نمی‌دانیم. بنابراین بجای وارد کردن یک مقدار SALARY واقعی در تاپل مربوط به کارمند مذکور –نمی‌توانیم این کار را انجام دهیم چون مقدار واقعی حقوق را نمی‌دانیم- محل مربوط به SALARY را با تهی علامت‌گذاری می‌کنم.

ENO	ENAME	DNO	SALARY
E2	Cheng	D1	

همانطور که می‌بینید تاپل در محل SALARY هیچ مقداری ندارد. در این کتاب از هاشور برای نمایش مکان‌های خالی استفاده کرده‌ام و شما می‌توانید مکان‌های هاشور خورده را دارای پرچم و یا نشانگر تهی در نظر بگیرید.

قانون جامعیت وجودی از نقطه نظر رابطه EMP تقریبا چنین است که هر کارمند می‌تواند نام، شعبه محل خدمت و حقوق نامعلوم داشته باشد اما شماره کارمندی وی باید حتما معلوم باشد چرا که در غیر این صورت مشخص نیست که ما در مورد کدام کارمند (و یا کدام «موجودیت») صحبت می‌کنیم.

این تمام مطالبی بود که می‌خواستم اینجا درباره تهی بیان کنم. این موضوع را تا فصل ۳ فراموش کنید.

ویژگی کار با داده‌ها

بخش کار با داده‌ها در مدل رابطه‌ای شامل قسمت‌های زیر است:

- مجموعه‌ای از عملگرهای رابطه‌ای مانند تفاضل (یا MINUS) است که تمامی آنها در مجموع جبر رابطه‌ای نامیده می‌شوند.
- عملگر انتساب رابطه‌ای که یک عبارت رابطه‌ای مانند r MINUS s (که r و s رابطه هستند) را به یک رابطه دیگر نسبت می‌دهد.

عملگـر رابطـه‌ای انتسـاب چگـونگی بـه روز رسـانی* را در مـدل رابطـه‌ای تشـریح مـی‌کنـد و در بخـش «رابطـه‌هـا، و یـا رلـورهـا» بیشـتر بـه ایـن موضـوع خـواهم پرداخـت. و در مـورد جبـر رابطـه‌ای،(تقریبـا) مجموعـه‌ای از عملگرهـای رابطـه‌ای اسـت کـه رابطـه‌هـای «جدیـد» را از روی رابطـه‌هـای «قـدیمی» بدسـت مـی‌آورد. و بـه عبـارت دقیـق‌تـر هـر عملگـر حـداقل یـک رابطـه را بعنـوان ورودی دریافـت کـرده، رابطـه‌ای دیگـر را بعنـوان خروجـی تولیـد مـی‌کنـد. مـثلا تفاضـل (و یـا MINUS) دو رابطـه را بعنـوان ورودی دریافـت کـرده، یکـی را از دیگـری «تفریـق» نمـوده و رابطـه دیگـری را بعنـوان خروجـی بوجـود مـی‌آورد. و ایـن خیلـی مهـم اسـت کـه رابطـه خروجـی یـک رابطـه دیگـر اسـت. ویژگـی شـناخه شـده تراگـذری (تعـدی) در جبـر رابطـه‌ای وجـود دارد و باعـث مـی‌شـود کـه بتـوان عبـارات تـو در تـو نوشـت. تـا زمـانی کـه خروجـی یـک عملگـر بـا نـوع ورودی مـورد نیـاز عملگـر بعـدی مطابقـت داشـته باشـد،خروجـی یـک عملگـر مـی‌توانـد بعنـوان ورودی عملگـر دیگـر مـورد اسـتفاده قـرار گیـرد. بـرای مثـال مـی‌تـوان تفاضـل دو رابطـه r و s را بدسـت آورد و نتیجـه را بعنـوان ورودی اجتمـاع بـا u در نظـر گرفـت و مجـددا نتیجـه حاصـل را بعنـوان یکـی از دو ورودی عملگر اشتراک با v مورد استفاده قرار داد و به همین ترتیب ...

مـی‌تـوان تعـدادی بیشـماری عملگـر تعریـف کـرد کـه همگـی در تعریـف «حـداقل یـک رابطـه بعنـوان ورودی و دقیقـا یـک رابطـه خروجـی» صـدق مـی‌کننـد. امـا وقتـی کـه از عملگـر صـحبت مـی‌کنـیم معمـولا منظـور هشـت عملگـر سـنتی اسـت(کـه کـاد در اولـین مقالـه‌اش آنهـا را معرفـی کـرد). در فصـل پـنج تعـدادی از عملگرهـای اضـافی را را معرفـی و در موردشـان بحـث خـواهم کـرد. شـکل ۱-۲ معرفـی تصـویر ایـن هشـت عملگـر مـی‌باشـد. توجـه: اگـر بـا برخـی از ایـن عملگـرهـا ناآشـنا هسـتید –خصوصـا تقسـیم!- و فهـم ایـن توضـیحات مختصـر برایتـان مشـکل اسـت، نگـران نباشـید. مـن قصـد دارم در آینـده مجـددا بـا مثال و توضیحات بیشتر به آنها بپردازم.(عمدتا در فصل ۵)

*در این کتاب به روز رسانی بعنوان کلمه‌ای عمومی برای عملگرهای INSERT، DELETE و UPDATE به کار برده شده است و زمانی که منظور دقیقا UPDATE باشد آنرا با حروف بزرگ انگلیسی خواهم نوشت.

گزینش

تمامی تاپل‌هایی از یک رابطه که دارای شرایطی خاص باشند را برمی‌گرداند. مثلا ممکن است رابطه EMP را گزینش کنیم تا تاپل‌هایی که مقدار EMP آنها D2 است را بدست آوریم.

پرتو

رابطه‌ای را برمی‌گرداند که حاوی تمام (زیر)تاپل‌های یک رابطه، بعد از حذف برخی ویژگی‌های خاص می‌باشد. برای مثال ممکن است رابطه EMP را فقط بر روی ویژگی‌های ENO و SALARY پرتو کنیم.

ضرب

رابطه‌ای را برمی‌گرداند که حاوی تمامی تاپل‌های ممکن از ترکیب دو تاپل است که هر یک از آنها متعلق به یکی از دو رابطه مشخص است. ضرب به نام‌های ضرب دکارتی، عملیات ضربدری، پیوند ضربدری و پیوند دکارتی شناخته می‌شود.(در حقیقت این عمل نوع خاصی از پیوند است، همانطور که در فصل ۵ خواهید دید.)

اشتراک

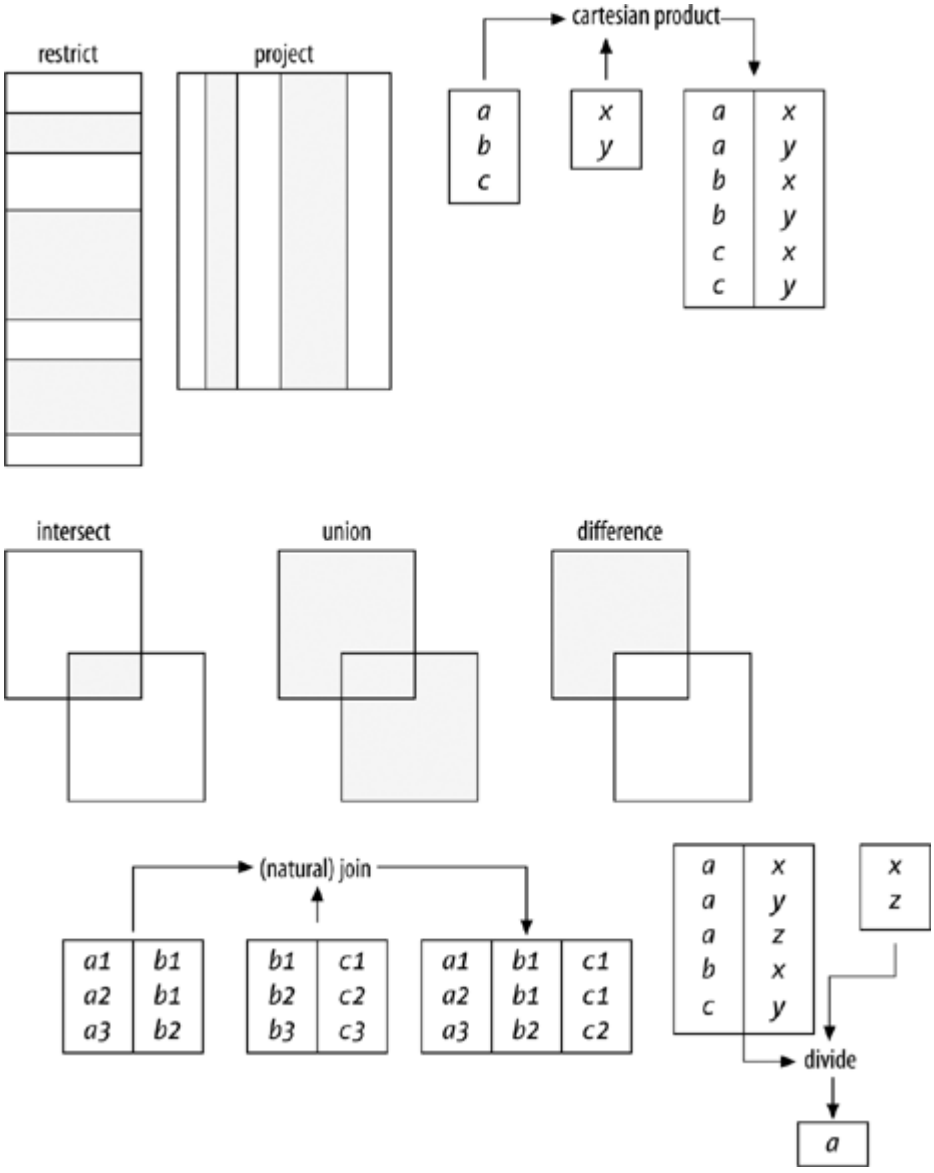
رابطه‌ای را برمی‌گرداند که حاوی تاپل‌های موجود در هر دو رابطه است.(در واقع اشتراک هم نوع خاصی از پیوند است.)

اجتماع

رابطه‌ای را برمی‌گرداند که حاوی تمامی تاپل‌های موجود در یکی و یا هر دو رابطه معین است.

تفاضل

رابطه‌ای را بازمی‌گرداند که حاوی تمام تاپل‌هایی است که در رابطه اول حضور دارند ولی در رابطه دو موجود نیستند.



شکل ۱-۲. جبر رابطه‌ای اصلی (نتایج هاشور خورده‌اند)

پیوند

رابطه‌ای را برمی‌گرداند که حاوی تمامی تاپل‌های حاصل از ترکیب‌های ممکن دو تاپل، که هر یک از آنها متعلق به یکی از رابطه‌هاست می‌باشد با این شرط که دو تاپل شرکت کرده در هر تاپل بدست آمده، دارای مقدار مشترک برای ویژگی‌هایی خاص در هر دو رابطه باشند.(و این مقدار مشترک فقط یک بار –و نه دو بار- در تاپل بدست آمده دیده شود).

توجه: این نوع پیوند بطور سنتی پیوند طبیعی نامیده می‌شود. از آنجا که پیوند طبیعی اهمیت فوق‌العاده دارد، کلمه پیوند به پیوند طبیعی اطلاق می‌شود و من هم در این کتاب از این امر پیروی خواهم کرد.

تقسیم

دو رابطه که یکی از آنها باینری و یکی یونری است را دریافت می‌کند و رابطه‌ای را برمی‌گرداند که شامل تمام مقادیر یکی از ویژگی‌های رابطه باینری است به گونه‌ای که (ویژگی دیگر آن) با تمامی مقادیر موجود در رابطه یونری مطابقت داشته باشد.

آخرین نکته قبل از خاتمه این بحث: همانطور که احتمالا می‌دانید چیز دیگری هم بنام حساب رابطه‌ای وجود دارد. حساب رابطه‌ای را می‌توان بعنوان جایگزینی برای جبر رابطه‌ای به حساب آورد. به این معنا که بجای بیان عملیات به گونه جبر رابطه‌ای(به همراه انتساب)، از بیان آنالیز رابطه‌ای(به همراه انتساب) استفاده کنیم. این دو با هم مساوی و قابل جایگزینی با یکدیگر هستند. به نظر می‌رسد که برای هر عبارت جبر رابطه‌ای یک عبارت آنالیز رابطه‌ای وجود دارد و بالعکس. در ضمیمه آ کمی بیشتر در مورد حساب رابطه‌ای صحبت خواهم کرد.

مثال اجرایی

این مرور مختصر را با معرفی مثالی به پایان می‌برم که پایه اکثر مباحث آتی کتاب خواهد بود: مثال شناخته شده توزیع کنندگان و قطعات(شکل ۱-۳ را ببینید)

توزیع کنندگان

رابطه S توزیع کنندگان را مشخص می کند.(به بیان دقیقتر توزیع کنندگان طرف قرارداد) هر توزیع کننده یک شماره توزیع کننده (SNO) که بصورت یکتا به وی اختصاص یافته است(و {SNO} کلید اصلی است)، یک نام (SNAME) نه لزوما یکتا (در شکل ۱-۳ تصادفا یکتا است)، یک رتبه و یا پایه (STATUS) و یک محل(CITY)، دارد.

قطعه‌ها

رابطه P قطعات را مشخص می کند. هر قطعه یک شماره قطعه (PNO) که یکتا است (و {PNO} کلید اصلی است)، یک نام (PNAME)، یک رنگ (COLOR)، یک وزن (WEIGHT) و یک مکان که این گونه قطعات در آن انبار شده اند(CITY)، دارد.

سفارش‌ها

رابطه SP سفارش‌ها را مشخص می کند.(نشان می‌دهد که کدام قطعات توسط کدامیک از توزیع کنندگان، تامین شده است.). هر سفارش یک شماره توزیع کننده (SNO)، یک شماره قطعه (PNO) و یک تعداد (QTY) دارد. من فرض کرده‌ام که در هر زمان حداکثر یک سفارش برای یک قطعه و یک توزیع کننده خاص وجود دارد.(بنابراین {SNO,PNO} کلید اصلی و همچنین {SNO} و {PNO} هر کدام کلید

خارجی هستند و با کلید اصلی S و P مطابقت دارند.توجه کنید که مطابق شکل ۱-۳ توزیع‌کننده‌ای وجود دارد –توزیع کننده S5- که دارای هیچ سفارشی نیست.

S

SNO	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

SNO	PNO	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

P

PNO	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Oslo
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London

شکل ۱-۳. بانک اطلاعاتی توزیع‌کنندگان و قطعات – با مقادیر نمونه

طراحی در برابر پیاده‌سازی

قبل از هر چیز بایستی موضوعی را روشن کنم که پایه تمامی مطالب این کتاب است. بدیهی است که مدل رابطه‌ای، یک مدل داده‌ای است. متاسفانه کلمه مدل داده‌ای در دنیای پایگاه‌داده‌ها دارای دو معنای کاملا متفاوت است. معنای بنیادی‌تر چنین است:

تعریف: «مدل داده» (اولین معنی) یک تعریف تجریدی، خودکفا و منطقی از ساختارهای داده‌ای، عملگرها و ... می‌باشد که در کنار هم ماشینی انتزاعی تشکیل می‌دهند که قابلیت تراکنش با کاربران را دارد.(طراحی)

این برداشت مستقیما به مدل رابطه‌ای مربوط می‌شود. با این تعریف می‌توانیم گام بعدی را با تمایز شناخت مدل رابطه‌ای و پیاده سازی آن برداریم که بصورت زیر تعریف می‌شود:

تعریف: «پیاده‌سازی» یک مدل داده‌ای، تحقق فیزیکی ماشین انتزاعی و اجزای تشکیل دهنده مدل، بر روی یک ماشین واقعی است.

من این مفاهیم را با نگاه ویژه به مدل رابطه‌ای تشریح خواهم کرد. اول مفهوم شناخته شده رابطه: استفاده کنندگان باید بدانند که رابطه‌ها چه هستند و بدانند که از تاپل‌ها و ویژگی‌ها تشکیل شده‌اند. تمامی اینها اجزایی از مدل هستند. در مقابل نیازی نیست که بدانند که چگونه رابطه‌ها بصورت فیزیکی بر روی دیسک ذخیره می‌شوند و مقادیر منحصر به فرد داده‌ها چگونه کدگذاری می‌شوند و ایندکس‌ها و مسیرهای دستیابی به اطلاعات چگونه ایجاد می‌شوند. تمامی اینها اجزای پیاده سازی –و نه اجزای مدل– هستند.

مثلا در مورد «پیوند»:کاربران باید بدانند که پیوند چیست و چگونه باید آنرا درخواست کرد و نتیجه آن چه شکلی دارد. تمام اینها مربوط به مدل (طراحی) هستند. ولی نباید بدانند که پیوند چگونه بصورت فیزیکی پیاده‌سازی می‌شود و یا در پشت پرده چه کارهایی انجام می‌گردد و یا کدام ایندکس‌ها و مسیرهای دستیابی مورد استفاده قرار می‌گیرند و در دستگاه‌های ورودی و خروجی چه اتفاقی رخ می‌دهد. تمامی اینها به پیاده سازی – و نه طراحی – مربوط می‌شوند.

بطور خلاصه:

- طراحی چیزی است که استفاده کننده باید آنرا بداند.
- پیاده‌سازی چیزی است که استفاده کننده نباید آنرا بداند.

(البته من نمی‌گویم که کاربران حق ندارند چیزی در مورد پیاده سازی بدانند، بلکه صحبت این است که می‌توان آنها را از این کار معاف کرد. بنابراین تمام مسائل مربوط به پیاده سازی بایستی –حداقل بصورت بالقوه– از دید کاربران پنهان نگه داشته شود.)

در اینجا برخی نتایج مطلب فوق را می‌بینید: اولا توجه کنید که کارایی موضوعی است که (بر خلاف تصور عامه) به پیاده سازی مربوط می‌شود ونه به طراحی. ما اکثرا می‌گوییم که پیوند به آهستگی انجام می‌شود. ولی این جمله بی معنا است. پیوند بخشی از طراحی است و طراحی نمی‌تواند سریع یا کند باشد. بلکه این امر بایستی به

پردازش‌های موجود در پیاده سازی اطلاق شود. بنابراین می‌توان گفت محصول فلان از نظر نوع خاصی پیوند، از محصول بهمان سریعتر است.

توجه

من قصد ندارم که یک عقیده غلط را جا بیاندازم. درست است که کارایی در اصل به پیاده‌سازی مربوط می‌شود ولی این به این معنا نیست که اگر از مدل(طراحی) بصورت نادرست استفاده شود پیاده‌سازی خوبی بدست خواهد آمد ! این یکی از دلایلی است که به موجب آن لازم است مدل را بشناسید. (در پاراگراف قبل فرض من این بود که هیچکس از مدل بد استفاده نمی‌کند).

اگر عبارتی مثل S JOIN SP بنویسید، در محدوده حقوق خود عمل کرده‌اید و پیاده سازی هم درخواست شما را به بهترین نحو انجام می‌دهد. ولی اگر اصرار داشته باشید که نحوه انجام آنرا هم خودتان مشخص کنید، مانند این:

```
do for all tuples in S ;
    fetch S tuple into TNO, TN, TS, TC ;
    do for all tuples in SP with SNO = TNO ;
      fetch SP tuple into TNO, TP, TQ ;
      emit tuple TNO, TN, TS, TC, TP, TQ ;
    end ;
  end ;
```

نخواهید توانست که به کارائی مناسبی برسید. سیستم‌های رابطه‌ای نباید مانند روش‌های ذخیره و بازیابی مورد استفاده قرار گیرند.

ثانیا، همانطور که احتمالا می‌دانید، تمایز منطقی بین طراحی و پیاده‌سازی ما را قادر به دستیابی استقلال داده‌ای می‌کند. استقلال داده‌ای (کلمه چندان مناسبی نیست با این حال احتمالا آنرا رها نخواهیم کرد.) بدین معناست که دارای این آزادی باشیم که نحوه ذخیره و بازیابی فیزیکی اطلاعات بر روی رسانه را تغییر دهیم بدون اینکه نیازی به تغییرات متناظر در دید کاربران وجود داشته باشد. دلیل این که ممکن است بخواهیم روش ذخیره و بازیابی را تغییر دهیم، بهبود کارایی است. ایجاد چنین تغییراتی بدون ایجاد تغییر در دید کاربران بدین معناست که برنامه‌های کاربردی موجود، کوئری‌ها و...

همچنان قادر به کار خواهند بود. نکته بسیار مهم این است که استقلال داده‌ای به معنای «حفاظت از سرمایه‌گذاری انجام گرفته برای پرورش نیروی انسانی و برنامه‌های کاربردی» می‌باشد.

همان‌طور که ملاحظه کردید تمایز طراحی و پیاده‌سازی امری منحصر به فرد و در عین حال بسیار مهم است و بیشتر به عنوان تمایز سطوح فیزیکی و منطقی شناخته می‌شود. متاسفانه اکثر سیستم‌های بانک‌اطلاعاتی امروزی – حتی آنهایی که ادعای رابطه‌ای بودن را دارند- آنطور که باید و شاید این تمایز را به وجود نیاورده‌اند و در نتیجه استقلال داده‌ای را آنطور که شایسته است و سیستم‌های رابطه‌ای در تئوری قابلیت آنرا دارند، فراهم نمی‌کنند. من به این موضوع در بخش بعد و همین‌طور فصل ۷ خواهم پرداخت.

اکنون می‌خواهم به دومین معنای مدل داده بپردازم و معتقدم که با آن مانوس هستید:

تعریف: «مدل داده» (دومین معنی) یعنی طرحی برای نگهداری داده‌های ماندگار یک موسسه بخصوص.

به بیان دیگر مدل داده‌ای (در معنای دوم) به طرح بانک اطلاعاتی اطلاق می‌شود. برای مثال ممکن است از مدل داده‌ای برای یک بانک،یک بیمارستان و یا یک اداره دولتی صحبت کنیم.

حال که این دو معنای متفاوت شرح داده شدند، می‌خواهم توجه شما را به مقایسه‌ای جلب کنم که تصور می‌کنم رابطه آنها را به خوبی روشن می‌کند:

- یک مدل داده در معنای اول مانند یک زبان برنامه نویسی است که ممکن است برای حل برخی مسائل مورد استفاده قرار گیرد ولی در نهایت خود مستقیما به هیچ مساله خاصی مربوط نمی‌شود.
- یک مدل داده در معنای دوم مانند برنامه به خصوصی است که به زبان فوق نوشته شده است. این برنامه از امکانات فراهم شده توسط مدل در معنای اول، برای حل یک مساله خاص استفاده می‌کند.

به هر حال، نتیجه منطقی تمام مطالب بالا این است که اگر در مورد مدل‌های داده در معنای دوم صحبت می‌کنیم منظور به طور کل مدل رابطه‌ای و یا مدل رابطه‌ای

خاصی(بدون تعریف مشخص) است. ولی اگر از مدل‌های داده در معنای اول صحبت کنیم منظور فقط یک مدل رابطه‌ای (با تعریف مشخص) است. در فصل ۸ بیشتر در این باره خواهم گفت.

در باقیمانده این کتاب لفظ مدل‌داده‌ای – و یا فقط مدل–، را در معنای اول(طراحی) به کار خواهم برد.

ویژگی‌های رابطه‌ها

اکنون به مفاهیم پایه رابطه‌ای بازمی‌گردیم. در این فصل می‌خواهم بحث را بر روی برخی ویژگی‌های رابطه متمرکز کنم. قبل از هر چیز هر رابطه یک عنوان و یک بدنه دارد. عنوان مجموعه‌ای از ویژگی‌هاست(منظور از ویژگی زوج ویژگی نام:نوع است) و بدنه مجموعه‌ای از تاپل‌ها می‌باشد که با عنوان مطابقت دارند. برای مثال در رابطه توزیع کنندگان شکل ۱-۳ چهار ویژگی در عنوان و پنج تاپل در بدنه وجود دارد. توجه داشته باشید که رابطه در واقع دارای تاپل نیست، بلکه دارای بدنه است و این بدنه است که تعدادی تاپل دارد. ولی ما معمولا برای سادگی چنین می‌گوییم.

کاملا صحیح است که بگوییم عنوان از زوج‌های ویژگی نام: نوع تشکیل شده است. با این وجود معمولا نوع‌ها از تصاویر (مانند شکل ۱-۳) حذف می‌شوند و چنین به نظر می‌رسد که عنوان فقط مجموعه‌ای از نام ویژگی‌ها است.مثلا ویژگی STATUS دارای یک نوع است (فرض کنید INTEGER) ولی من آن را در شکل ۱-۳ نشان نداده‌ام ولی شما هم نباید این موضوع را فراموش کنید!

تعداد ویژگی‌های عنوان درجه (و گاهی اریتی) و تعداد تاپل‌های بدنه کاردینالیتی خوانده میشود. برای مثال در رابطه‌های S، P و SP در شکل ۱-۳ دارای درجه به ترتیب ۴، ۵ و ۳ بوده کاردینالیتی آنها به ترتیب ۵، ۶، و ۱۲ می‌باشد.

رابطه‌ها هرگز دارای تاپل‌های تکراری نیستند. این ویژگی قابل استنباط است چرا که بدنه مجموعه‌ای از تاپل‌ها است و مجموعه‌ها در ریاضیات دارای اعضای تکراری نیستند. و اما SQL اینجا درمی‌ماند: همانطور که می‌دانید جداول SQL وجود سطرهای تکراری را مجاز می‌شمارند در نتیجه نمی‌توان جدول‌های SQL را همواره رابطه به

حساب آورد. لطفا درک کنید که من در این کتاب کلمه «رابطه» را فقط به معنای رابطه، بدون تاپل‌های تکراری و عینا طبق تعریف به کار برده‌ام. همچنین متوجه باشید که عملگرهای رابطه‌ای نیز –باز هم طبق تعریف- نتایجی با تاپل‌های تکراری تولید نمی‌کنند برای مثال پرتو رابطه توزیع‌کنندگان بر روی CITY در شکل ۱-۳ نتیجه سمت چپ –و نه سمت راست- را تولید می‌کند.

CITY
London
Paris
Athens

CITY
London
Paris
Paris
London
Athens

(نتیجه سمت چپ ممکن است بوسیله این کوئری SQL بدست آید:

SELECT DISTINCT S.CITY FROM S

با حذف DISTINCT نتیجه‌ای غیر رابطه‌ای سمت راست حاصل خواهد شد. توجه کنید که در جدول سمت راست از دو خط زیر عنوان استفاده نکرده‌ام به این دلیل که این جدول کلید اصلی ندارد.)

موضوع دیگر اینکه تاپل‌های رابطه از بالا تا پایین بدون ترتیب هستند. این ویژگی نیز قابل استنباط است چرا که بدنه یک مجموعه است و اعضای مجموعه ترتیبی ندارند.(مثلا {a,b,c} و {c,a,b} در ریاضیات با هم مساویند و این امر طبعا در مدل رابطه‌ای نیز صادق است.) البته وقتی ما یک جدول را بر روی کاغذ رسم میکنیم مجبوریم سطرها را به ترتیب نشان دهیم ولیکن این ترتیب نشانه هیچ چیز در مدل رابطه‌ای نیست. برای مثال من می‌توانستم سطرهای جدول توزیع‌کنندگان را در شکل ۱-۳ به هر ترتیب دیگری از جمله S3 بعد S1 بعد S5 بعد S4 بعد S2 در شکل نشان دهم و شکل همچنان نشان دهنده همان رابطه باشد.

توجه

واقعیت این است که تاپل‌های رابطه نامرتب‌اند ولی این به این معنا نیست که کوئری‌ها نمی‌توانند دارای ORDER_BY باشند.بلکه بدین معناست که نتایج تولید شده از چنین کوئری‌هایی رابطه نیست. ORDER_BY برای نمایش نتایج سودمند است ولی در عین حال یک عملگر رابطه‌ای نیست.

به همین صورت، ویژگی‌های رابطه نیز از چپ به راست بدون ترتیب اند چرا که عنوان نیز یک مجموعه است ولی وقتی یک جدول را بر روی کاغذ رسم می‌کنیم مجبوریم ستون‌ها را به ترتیب از چپ به راست نشان دهیم ولیکن این ترتیب نیز نشانه هیچ چیز در مدل رابطه‌ای نیست. در جدول توزیع کنندگان شکل ۱-۳ می‌توانستم ستون‌ها را به هر ترتیبی از چپ به راست –مثل STATUS, SNAME, CITY, SNO – بچینم و شکل همچنان نشان دهنده همان رابطه باشد. اتفاقا SQL در اینجا هم ناتوان است. ستون‌ها در SQL حتما دارای ترتیب هستند.(دلیل دیگر برای اینکه جدول‌های SQL رابطه نیستند.) مثلا دو شکل زیر نمایشگر یک رابطه و در عین حال دو جدول SQL متفاوت هستند:

SNO	CITY
S1	London
S2	Paris
S3	Paris
S4	London
S5	Athens

CITY	SNO
London	S1
Paris	S2
Paris	S3
London	S4
Athens	S5

(کوئری‌های ایجاد کننده این دو جدول عبارت است از:

```
SELECT S.SNO, S.CITY FROM S
SELECT S.CITY, S.SNO FROM S
```

ممکن است تصور کنید که این تفاوت اهمیت چندانی ندارد، ولی در واقع این امر نتایج و مسائل مهمی را دربردارد که در فصل‌های بعد بدان خواهیم پرداخت.)

مطلب بعدی اینکه رابطه‌ها همیشه نرمال شده‌اند.(یعنی در صورت اول نرمال یا 1NFهستند) به زبان ساده یعنی در نمایش تصویری جدول، در محل تلاقی هر سطر و ستون تنها یک مقدار دیده می‌شود و به زبان رسمی‌تر می‌توان گفت که هر یک از ویژگی‌های متعلق به هر تاپل رابطه دارای یک مقدار منفرد -و متناسب با نوع آن- می‌باشد. من در این مورد مطالب زیادی دارم که در فصل بعد خواهم گفت.

نکته دیگر اینکه ما بین رابطه‌های پایه و رابطه‌های ناشی شده تفاوت قائل می‌شویم. همانطور که قبلا گفته شد عملگرهای جبر رابطه‌ای ما را قادر می‌سازند که با استفاده از چند رابطه موجود –مانند آنچه در شکل ۱-۳ وجود دارد–، رابطه‌های جدیدی بدست آوریم. رابطه‌های داده شده پایه و بقیه ناشی شده هستند. پس یک سیستم رابطه‌ای بایستی ابزارهایی جهت تعریف روابط پایه در اختیار داشته باشد. در SQL این کار با دستور CREATE_TABLE انجام می‌شود.(در SQL جدول پایه معادل رابطه پایه است.) و روابط پایه بایستی نام‌گذاری شوند.

```
CREATE TABLE SP ... ;
```

رابطه‌های ناشی شده که «دید» نامیده می‌شوند نیز دارای نام هستند. یک دید (که بعنوان رابطه مجازی هم شناخته می‌شود) رابطه نامداری است که مقدار آن در هر لحظه از زمان، با ارزیابی یک عبارت رابطه‌ای است که در همان لحظه بدست می‌آید. مانند مثال SQL زیر:

```
CREATE VIEW SST_PARIS AS
    SELECT S.SNO, S.STATUS
    FROM   S
  WHERE  S.CITY = 'Paris' ;
```

اگر دیدها رابطه‌های پایه باشند شما می‌توانید با آنها کار کنید. ولی آنها رابطه‌های پایه نیستند. با این وجود می‌توانید تصور کنید که دیدها جدول شده‌اند و یا فرض کنید که دیدها رابطه‌های پایه‌ای هستند که بصورت پویا و در زمان مراجعه ایجاد می‌شوند.(در هر صورت مجبوریم که اینطور تصور کنیم که دیدها به هنگام مراجعه جدول هستند چرا که راه دیگری وجود ندارد. ولی این اتفاقی نیست که در واقعیت رخ

می‌دهد. اینکه دیدها واقعا چگونه کار می‌کنند موضوعی است که در فصل ۴ مورد بحث قرار خواهد گرفت.)

در اینجا بایستی به نکته مهمی اشاره کنم. اکثر مردم تفاوت بین رابطه‌های پایه و دیدها را به این صورت بیان می‌کنند.

- رابطه‌های پایه واقعا وجود دارند و بصورت فیزیکی در بانک اطلاعاتی ذخیره شده‌اند.
- دیدها وجود خارجی ندارند آنها فقط راه‌های مختلفی برای نگریستن به رابطه‌های پایه در اختیار ما قرار می‌دهند.

ولی مدل رابطه‌ای هیچ صحبتی در مورد اینکه چه چیزی بصورت فیزیکی ذخیره می‌شود، ندارد و نمی‌گوید که رابطه‌های پایه باید به صورت فیزیکی ذخیره می‌شوند. تنها باید تناظری بین رابطه‌های پایه و آنچه بصورت فیزیکی ذخیره می‌شود وجود داشته باشد و رابطه‌های پایه بایستی در زمان نیاز به طریقی ایجاد شوند. بنابراین رابطه‌های پایه می‌توانند به این طریق (تناظر با داده‌های ذخیره شده) و یا هر روش دیگری تولید شوند. برای مثال می‌توان پیوند دو رابطه توزیع‌کنندگان و سفارش‌ها را بصورت فیزیکی ذخیره کرد و یا بجای این کار روابط S و SP را بطور جداگانه ذخیره نمود و پیوند را در زمان نیاز بصورت مجازی ایجاد کرد. به بیان دیگر، مطابق مدل رابطه‌ای، رابطه‌های پایه از دیدها فیزیکی‌تر نیستند.

مدل رابطه‌ای عمدا هیچ چیز در مورد ذخیره فیزیکی نمی‌گوید تا سازندگان سیستم‌های بانک اطلاعاتی بتوانند آزادانه و از هر راهی که می‌خواهند-و تصور می‌کنند از آن راه به کارایی بهتری می‌رسند، البته بدون از دست دادن استقلال داده‌ها - به پیاده‌سازی آن بپردازند. واقعیت تاسف‌بار این است که اکثر تولیدکنندگان SQL این موضوع را درک نکرده‌اند. آنها جدول‌های پایه را مستقیما به دستگاه‌های ذخیره سازی فیزیکی* منتسب می‌کنند و (همانطور که در قسمت قبل گفته شد) از این رو

* همه می‌دانند که محصولات SQL امروزی خودشان امکاناتی همچون پارتیشن‌بندی، ایندکس‌بندی، کلاستربندی و در کل سازمان‌دهی داده‌های ذخیره شده در دیسک را فراهم می‌کنند. با این وجود من همچنان معتقدم که نگاشت به دستگاه‌های ذخیره‌سازی فیزیکی در این محصولات «تا حدودی مستقیم» است.

محصولات‌شان از استقلال داده‌ای کافی-در حدی که تئوری رابطه‌ای تعیین می‌کند- برخوردار نیست. این مشکل در خود SQL استاندارد -و همچنین اکثر نگارش‌های SQL- که از عباراتی نظیر «جدول‌ها و دیدها» استفاده کرده اند، وجود دارد. بدیهی است کسی که با چنین عباراتی سروکار دارد تصور می‌کند که جدول‌ها و دیدها دو چیز متفاوت هستند و بر این اساس به این نتیجه می‌رسد که جدول‌ها فیزیکی و دیدها غیر فیزیکی هستند. اما نکته‌ای که در مورد دیدها پوشیده مانده، این است که آنها هم جدول (و یا من ترجیح میدهم بگویم رابطه) هستند. به همین علت است که ما میتوانیم عملگرهای رابطه‌ای را بر آنها (مانند رابطه‌های معمولی) اعمال کنیم. چرا که دیدها، «رابطه‌های معمولی» اند. در این کتاب من از لغت رابطه به معنای رابطه استفاده خواهم کرد.(خواه رابطه‌ای پایه باشد، خواه یک دید، خواه نتیجه یک کوئری و یا هر چیز دیگر) و اگر منظورم رابطه پایه باشد، آنگاه کلمه «رابطه پایه» را به کارخواهم برد و به شما هم شدیدا توصیه می‌کنم که از این مقررات تبعیت کنید و دچار عوام‌زدگی نشوید و تصور نکنید کلمه «رابطه» فقط به معنای رابطه پایه است.

رابطه‌ها در برابر رل‌ورها

این احتمال وجود دارد که از مطالبی که تا کنون مطرح گردید تا حدودی مطلع باشید و امیدوارم که اینطور باشد.(این امیدواری به این معنا نیست که مباحث برای شما ملال‌آور بوده است.) در هر صورت هم اکنون قصد دارم مطلبی را ارائه کنم که احتمالا آن را نمی‌دانید. واقعیت این است که در مفاهیم رابطه‌ها—در معنایی که بیان شد- و «متغیر»های رابطه‌ای اختلاط وجود دارد.

برای چند لحظه بانک اطلاعاتی را فراموش کنید و مثال برنامه‌نویسی زیر را در نظر بگیرید. فرض کنید این دستور در یک زبان برنامه‌نویسی داده شده باشد:

```
DECLARE N INTEGER ... ;
```

در این صورت N یک integer نیست. بلکه یک متغیر است که مقادیر integer ای را می‌پذیرد.(در دفعات متفاوت مقادیر صحیح مختلف) موافقید؟ مطمئم که شما هم این مطلب را تایید می‌کنید. دقیقا به همین صورت اگر در SQL بگوییم:

```
CREATE TABLE T ... ;
```

در اینصورت T یک جدول نیست بلکه یک متغیر یک جدول و یا یک متغیر رابطه‌ای است که مقادیر از نوع رابطه را می‌پذیرد.(در دفعات متفاوت رابطه‌های مختلف)

شکل ۱-۳ را مجددا در نظر بگیرید. این شکل سه مقدار رابطه‌ای را نشان می‌دهد: بطور مشخص وضعیت بانک اطلاعاتی در یک زمان خاص نشان داده شده است. ولی اگر در زمان دیگری به بانک اطلاعاتی نگاه می‌کردیم، سه مقدار رابطه‌ای دیگر را در آنجا می‌دیدیم. به زبان دیگر S، P و SP در این بانک‌اطلاعاتی واقعا متغیرانند: دقیقا متغیرهای رابطه‌ای. برای مثال فرض کنید که متغیر رابطه‌ای S هم اکنون دارای مقداری است- مقدار رابطه‌ای- که در شکل ۱-۳ نشان داده شده است. فرض کنید که برخی تاپل‌ها (در واقع فقط یک تاپل) که مربوط به توزیع‌کنندگان مستقر در آتن است را حذف می‌کنیم:

```
DELETE S WHERE CITY = 'Athens' ;
```

نتیجه چنین خواهد بود:

S

SNO	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London

می‌توان چنین تصور کرد که مقداری جدید جایگزین مقدار قبلی S شده است.البته مقدار قدیمی(با پنج تاپل) و مقدار جدید با هم شباهت زیادی دارند ولی با این وجود آنها دو مقدار متفاوت اند. در حقیقت DELETE معادل منطقی و خلاصه شده درستور انتساب رابطه‌ای زیر است [*]:

```
S := S WHERE NOT ( CITY = 'Athens' ) ;
```

[*] من نمی‌توانم این دستور را به SQL بنویسم چرا که SQL دستور انتساب را مستقیما پشتیبانی نمی‌کند.در این کتاب من مثال‌ها را تا جایی که ممکن است با SQL نشان می‌دهم و در جایی که به هر دلیل این کار ممکن نیست-مانند همین جا- از یک زبان کمابیش بدون شرح(و واقعا رابطه‌ای) بنام تاتوریال‌دی استفاده خواهم کرد.این زبان را هیوج داروین و من در کتاب پایگاه‌های داده‌ها، نوع‌ها و مدل رابطه‌ای مورد استفاده قرار داده‌ایم و شما میتوانید برای درک مفاهیم انتزاعی مدل رابطه‌ای آن را مطالعه کنید (چیزی که متاسفانه SQL فاقد آن است).

در تمامی انتساب‌ها، «عبارت مبدا» (a) در سمت راست ارزیابی شده، و نتیجه این ارزیابی به «متغیر مقصد» (b) در سمت چپ منتسب می‌شود.

عبارت‌های مانوس INSERT و DELETE نیز خلاصه نویسی شده دستورات انتساب رابطه‌ای هستند. همانطور که در قسمت «مروری بر مدل اصلی» عنوان شد انتساب رابطه‌ای عملگر به روزرسانی اصلی مدل رابطه‌ای و در حقیقت تنها عملگر به‌روزرسانی مورد نیاز است.

بنابراین ما به دو موضوع مختلف سروکار داریم، مقادیر رابطه‌ای و متغیرهای رابطه‌ای. مشکل این است که در گذشته به دلیل اختصار، برای هر دو آنها از کلمه «رابطه» استفاده شده است و این روش قطعا موجب سردرگمی می‌شود. بنابراین در این کتاب از این پس من بین این دو مفهوم با دقت تمایز قائل شده‌ام و اگر منظور مقدار رابطه‌ای است از کلمه مقدار رابطه‌ای و اگر منظور متغیر رابطه‌ای است از کلمه متغیر رابطه‌ای استفاده کرده‌ام. هر چند که من هم اکثر اوقات مقدار رابطه‌ای را، رابطه (همان طور که مقدار بیشتر اوقات مقدار integer ای را integer می‌نامند) و متغیر رابطه‌ای را رل‌ور نامیده‌ام. برای مثال می‌گویم که بانک اطلاعاتی توزیع کنندگان و قطعات دارای سه رل‌ور است.

برای تمرین می‌توانید به مطالب قبلی این فصل بازگردید و ببینید من در کجاها از کلمه رابطه استفاده کرده‌ام در حالی که بایستی بجای آن از رل‌ور استفاده می‌کردم.

مقدارها در برابر متغیرها

تفاوت بین رابطه‌ها و رل‌ورها حالت خاصی از تفاوت عمومی بین مقدارها و متغیرها است و من می‌خواهم زمان کوتاهی را صرف این تفاوت عمومی‌تر کنم.(شاید کمی انحراف از موضوع باشد ولی تصور می‌کنم که ارزش آنرا دارد. چرا که روشن شدن این امر می‌تواند به فهم بسیاری از مطالب بعدی کمک زیادی نماید.) به چند تعریف توجه کنید:

تعریف: یک مقدار یک «ثابت منحصر بفرد» است: برای مثال عدد صحیح منحصر بفرد ۳. یک مقدار جایی در زمان و مکان ندارد. مقدارها می‌توانند در حافظه، معنای

خاصی را به ذهن بیاورند، و این به ذهن آوردن هم جایی در زمان و مکان ندارد. در واقع هر مقدار می‌تواند برای نمایش چیزهای مختلفی در زمان و مکان بکار رود. این تقریبا به این معناست که تعدادی متغیرمختلف –طبق تعریف زیر- می‌توانند، در یک زمان یا زمان‌های مختلف دارای مقداری یکسان باشند. نکته مهم اینکه یک مقدار قابل به‌روزرسانی نیست و اگر به فرض محال این کار انجام شود، بعد از به‌روزرسانی دیگر یک مقدار نخواهد بود.

تعریف: یک متغیر نگهداری کننده یک مقدار است. یک متغیر در ظرف زمان و مکان جا دارد و همچنین یک متغیر بر خلاف یک مقدار قابل به‌روزرسانی است. مقدار فعلی متغیر می‌تواند با مقدار دیگری جایگزین شود.

لطفا دقیقا توجه کنید که تنها چیزهای ساده مانند عدد صحیح ۳، «مقدار» های قابل‌قبول نیستند. مقدارها می‌توانند تا حد دلخواه پیچیده باشند. برای مثال یک نقطه هندسی، یک چند ضلعی، یک اشعه ایکس، یک سند XML، یک اثر انگشت، یک آرایه، یک پشته، یک لیست و یک رابطه، همگی مقدار هستند. مقیاس‌ها برای معنادهی به متغیرها بوجود آمده‌اند. من در مورد این موضوع در دو فصل آینده توضیحات بیشتری می‌دهم.

ممکن است برایتان عجیب باشد که چگونه مردم در تفاوت بین مقدارها و متغیرها دچار اشتباه می‌شوند. ولی واقعیت این است که این مساله برای بسیاری از افراد پیش می‌آید. برای روشن شدن این موضوع ، متن زیر را از کتاب خودآموز بانک‌های اطلاعاتی شی گرا نقل قول کرده‌ام. (مطالب داخل کروشه را خودم اضافه کرده‌ام):

ما بین دو مفهوم متغیر و.. تمایز قائل می‌شویم. شی به مقدار فعلی متغیر گفته می‌شود. [شی یک مقدار است]. ما بین اشیا و مقدارها تمایز قائل می‌شویم[پس شی مقدار نیست] یک «تغییر دهنده» می‌تواند بر برخی اشیا اثر بگذارد[پس شی یک متغیر است].

خلاصه

بیشتر در این فصل مقدماتی را بیان کردم و امیدوارم اکنون آنها را فراگرفته باشید.(ممکن است این مطالب برایتان آسان باشد). به هر حال خلاصه این مطالب به شرح زیر است:

- توضیح دادم که چرا باید به قواعد توجه کنیم و نه به محصولات. همچنین چرا از اصطلاحات رسمی مانند رابطه، تاپل و ویژگی استفاده می‌کنیم و نه از معادل‌های دوستانه آنها در SQL.
- ادعا کردم که مدل رابطه‌ای و SQL یکی نیستند. برخی از این تفاوت‌ها را هم اکنون می‌دانیم–مثلا SQL سطرهای تکراری را مجاز می‌شمارد.– و بسیاری تفاوت‌های دیگر را هم در فصل‌های بعد خواهیم دید.
- آشنایی مختصری با مدل اصلی ارائه شد و این موضوعات مورد بررسی قرار گرفت: نوع، رابطه n اری، تاپل، ویژگی، کلید کاندید، کلید اصلی، کلید خارجی، جامعیت وجودی، جامعیت ارجاعی، انتساب رابطه‌ای و جبر رابطه‌ای. به خاصیت شرکت‌پذیری نیز اشاره کردم و بصورت بسیار مختصر عملگرهای گزینش، پرتو، ضرب، اشتراک، اجتماع، تفاضل، پیوند و تقسیم را شرح دادم.
- در مورد مشخصات رابطه بحث کردم و عبارات عنوان، بدنه، کاردینالیتی و درجه را معرفی کردم. رابطه‌ها تاپل تکراری، ترتیب تاپل‌ها از بالا به پایین و ترتیب ویژگی‌ها از چپ به راست را ندارند. همچنین در مورد دیدها بحث کردم.
- تفاوت‌های میان مدل و پیاده‌سازی، رابطه‌ها و رل‌ورها و مقادیر و متغیرها را روشن کردم. مدل در برابر پیاده سازی بحثی است که ما را به استقلال داده‌ها می‌رساند.

و آخرین نکته (که تاکنون آنرا مطرح نکرده‌ام ولی امیدوارم از مباحث گذشته به این نتیجه رسیده باشید): در مجموع مدل رابطه‌ای طبیعتی اعلان‌گرا –و نه روال گرا– دارد. ما روش‌های اعلان‌گرا را بر روش‌های روندگرا ترجیح میدهیم چرا که ملموس‌تر هستند. دلیل این امر هم روشن است: اعلان‌گرا به این معناست که سیستم کارها را انجام

می‌دهد و روال‌گرا به این معناست که کاربر کارها را انجام می‌دهد. از همین رو مدل رابطه‌ای از کوئری‌های اعلانی، به‌روزرسانی‌های اعلانی، تعریف دید بصورت اعلانی و تعریف قواعد جامعیت بصورت اعلانی، پشتیبانی می کند.*

تمرین‌ها

۱- (تکراری از متن کتاب با اندکی تغییر) اگر تاکنون آنرا انجام نداده‌اید، مطالب این فصل را مرور کنید و در هر کجا که کلمه رابطه را به کار برده‌ام، بگویید که صحیح است و یا باید بجای آن رل‌ور را به کار می‌بردم.

۲- ای اف کاد که بود؟

۳- دامنه چیست؟

۴- از جامعیت ارجاعی چه فهمیدید؟

۵- لغات عنوان، بدنه، ویژگی، تاپل، کاردینالیتی و درجه که در قسمت مقادیر رابطه‌ای تعریف شدند، به همان صورت قابل تعمیم به رل‌ورها هم هست. مطمئن شوید که این جمله را فهمیده‌اید.

۶- دو معنای متفاوت مدل داده‌ای را شرح دهید.

۷- تفاوت بین مدل و پیاده‌سازی را به زبان خودتان بیان کنید.

---

* وقتی که این کتاب در دست چاپ بود مطلع شدم که حداقل یکی از محصولات شناخته شده SQL کلمه «اعلانی» را به معنای عدم انجام کار بوسیله سیستم استفاده کرده است! به این معنا که کاربر مجاز است دستورات را بصورت اعلانی بگوید ولی مجبور به این کار نیست. چنین استفاده‌های نابجایی از اصطلاحات فنی، هیچ کمکی به فهم بهتر مطلب نمی‌کند.

۸- در متن گفتم که جدول‌های رسم شده مانند ۱-۱ و ۱-۳ رابطه نیستند، بلکه تصویر شده رابطه هستند. تفاوت‌هایی که بین این تصاویر و روابط متناظر با آنها وجود دارد چه هستند؟

۹- استقلال داده‌ای را به زبان خودتان تعریف کنید.

۱۰- (سعی کنید این تمرین را بدون نگاه کردن به متن کتاب انجام دهید) چه رل‌ورهایی در بانک اطلاعاتی توزیع‌کنندگان و قطعات وجود دارد؟ چه ویژگی‌هایی در هر یک آنها وجود دارد؟ کلیدهای اصلی و خارجی کدامند؟ (نکته این تمرین این است که خود را با این مثال آشنا می‌کنید. واضح است که نبایستی جزئیات داده‌های واقعی را حفظ کنید.)

۱۱- «فقط یک مدل رابطه‌ای وجود دارد.» این جمله را توضیح دهید.

۱۲- مطلب زیر از یک کتاب جدید پایگاه داده‌ها نقل قول شده است: «مهم است که بین رابطه‌های ذخیره شده —جدول‌ها- و رابطه‌های مجازی —دیدها- تمایز قائل شویم. ما بایستی کلمه رابطه را برای یک جدول و یا دید قابل استفاده بکار بریم و زمانی که قصد داریم بر رابطه‌های ذخیره شده تاکید کنیم از لغات رابطه پایه و یا جدول پایه استفاده می‌کنیم.» این متن دارای تناقضات و اشتباهات زیادی در مورد مدل رابطه‌ای است. هر تعداد از آنها را که می‌توانید پیدا کنید.

۱۳- متن زیر از کتاب جدید دیگری در رابطه با پایگاه‌داده‌ها استخراج شده است:«مدل رابطه‌ای ... برای هر رابطه، جدول‌هایی ساده و روابط چند به چند تعریف می‌کند. کلیدهای ارجاعی جدول‌ها را به هم متصل می‌کنند و رابطه بین آنها را نشان می‌دهند. ایندکس‌های اولیه و ثانویه راه سریعی برای دستیابی به اطلاعات فراهم می‌کنند.» این متن تعریف مدل رابطه‌ای است... چه اشکالی در آن وجود دارد؟

۱۴- چند دستور SQL بصورت CREATE TABLE بنویسید که بانک اطلاعاتی توزیع‌کنندگان و قطعات را تعریف کند.

۱۵- دستور INSERT زیر در SQL و مربوط به بانک‌اطلاعاتی توزیع‌کنندگان و قطعات است.

```
INSERT INTO SP ( SNO, PNO, QTY )
    VALUES ( SNO('S5'), PNO('P6'), QTY(250) ) ;
```

معادل این دستور را با انتساب رابطه‌ای بنویسید. توجه: فرض کرده‌ام ویژگی‌های SNO، PNO و QTY از نوع SNO، PNO و QTY هستند و به ترتیب عبارات SNO('S5')،PNO('P6') و QTY(250) با این انواع مطابقت دارند. من مطالب بیشتری در این مورد در دو فصل آینده دارم. من می‌دانم که هنوز نحوه و جزئیات دستور انتساب رابطه‌ای را شرح نداده‌ام. زیاد نگران درست نوشتن پاسخ نباشید و فقط نهایت سعی خود را بکنید.

۱۶- دستور UPDATE زیر مربوط به پایگاه توزیع‌کنندگان و قطعات است.

```
UPDATE S
SET    STATUS = 25
WHERE  S.CITY = 'Paris' ;
```

معادل این دستور را با انتساب رابطه‌ای بنویسید.(هدف از این تمرین آن است که در مورد آن تفکر کنید. من هنوز آنقدر در این مورد برای شما توضیح نداده‌ام که بتوانید پاسخ کاملا صحیح به این سوال بدهید. فصل ۵ را جهت مباحث آتی ببینید.)

۱۷- در این فصل گفتم که SQL بطور مستقیم دستور انتساب را پشتیبانی نمی‌کند. آیا بطور غیر مستقیم این کار ر ا انجام می‌دهد؟ اگر بلی چگونه؟

۱۸- از نقطه نظر کاربردی فکر می‌کنید چرا تاپل‌های تکراری، ترتیب بالا به پایین تاپل‌ها و ترتیب چپ به راست ویژگی‌ها ایده‌های بسیار بدی هستند؟ (این سه سوال در متن پاسخ داده نشده‌اند و این تمرین بیشتر برای بحث در گروه‌ها مناسب است. ما در آینده بیشتر به این قبیل مطالب خواهیم پرداخت.