جهاد دانشگاهی
واحد اصفهان

اصول و طراحی پایگاه داده‌ها

(مخصوص دانشجویان کاردانی و کارشناسی کامپیوتر)

تألیف: مهندس تالین ساهاکیان
عضو هیئت علمی دانشگاه آزاد اسلامی

فهرست مطالب

عنوان	صفحه
فصل ۱ – تاریخچه و مفاهیم پایگاه داده‌ها	۹
۱-۱) مفاهیم مربوط به پایگاه داده‌ها	۹
۱-۲) تاریخچهٔ پایگاه داده‌ها	۱۱
۱-۲-۱) ریشه‌های تاریخی پایگاه داده‌ها: سیستمهای فایل	۱۱
۱-۲-۲) مفاهیم مربوط به سیستمهای فایل	۱۲
۱-۲-۳) معایب سیستمهای فایل	۱۴
۱-۳) سیستمهای پایگاه داده‌ها	۱۹
۱-۴) اجزاء یک سیستم پایگاه داده‌ها	۲۱
۱-۵) انواع سیستمهای پایگاه داده‌ها	۲۳
۱-۶) وظایف DBMS	۲۵
۱-۷) معماری سیستمهای پایگاه داده‌ها	۳۰
۱-۸) مدلهای پایگاه داده‌ها	۳۵
۱-۸-۱) مدل سلسله مراتبی	۳۷
۱-۱-۸-۱) اساس مدل سلسله مراتبی	۳۷
۱-۱-۸-۲) بررسی عملیات گوناگون در مدل سلسله مراتبی	۳۹
۱-۱-۸-۳) معایب مدل سلسله مراتبی	۴۱
۱-۸-۲) مدل شبکه‌ای	۴۱
۱-۲-۸-۱) اساس مدل شبکه‌ای	۴۱
۱-۲-۸-۲) بررسی عملیات گوناگون در مدل شبکه‌ای	۴۳
۱-۲-۸-۳) محاسن مدل شبکه‌ای نسبت به مدل سلسله مراتبی	۴۴

۴-۲-۸-۱) معایب مدل شبکه‌ای ۴۴
۳-۸-۱) مدل رابطه‌ای ۴۵
۱-۳-۸-۱) اساس مدل رابطه‌ای ۴۵
۲-۳-۸-۱) بررسی عملیات گوناگون در مدل رابطه‌ای ۴۵
۳-۳-۸-۱) محاسن مدل رابطه‌ای نسبت به مدل شبکه‌ای ۴۷
تمرینهای فصل ۴۸
فصل ۲ـ پایگاه داده‌های رابطه‌ای ۴۹
۱-۲) مفاهیم پایگاه داده‌های رابطه‌ای ۴۹
۲-۲) قوانین حاکم بر پایگاه داده‌های رابطه‌ای: قوانین جامعیت داده‌ای ۶۳
تمرینهای حل شده ۶۴
تمرینهای فصل ۸۲
فصل ۳ـ جبر رابطه‌ای ۹۱
۱-۳) عملگرهای یکتایی ۹۴
۱-۱-۳) عملگر انتخاب ۹۴
۲-۱-۳) عملگر پرتو ۹۶
۳-۱-۳) عملگر بسط ۹۷
۴-۱-۳) عملگر خلاصه‌سازی یا گروه‌بندی ۹۸
۲-۳) عملگرهای اجتماع، اشتراک و تفاضل ۹۹
۱-۲-۳) عملگر اجتماع ۱۰۰
۲-۲-۳) عملگر اشتراک ۱۰۱
۳-۲-۳) عملگر تفاضل ۱۰۲
۳-۳) عملگرهای ضرب و پیوند ۱۰۲
۱-۳-۳) عملگر ضرب ۱۰۲
۲-۳-۳) عملگر پیوند ۱۰۴
۱-۲-۳-۳) پیوند شرطی یا پیوند تتا ۱۰۴
۲-۲-۳-۳) پیوند طبیعی ۱۰۶
۳-۲-۳-۳) شبه پیوند ۱۰۸
۴-۲-۳-۳) عملگرهای پیوند خارجی ۱۰۹
۱-۴-۲-۳-۳) پیوند خارجی چپ ۱۰۹
۲-۴-۲-۳-۳) پیوند خارجی راست ۱۱۰

۳-۴-۲-۳-۳) پیوند خارجی کامل ۱۱۱
۴-۳) عملگر تقسیم ۱۱۱
۵-۳) عملگر جایگزینی ۱۱۳
تمرینهای حل شده ۱۱۴
تمرینهای فصل ۱۲۱
فصل ٤ـ زبان پرس و جوی ساخت یافته ۱۲۵
۱-۴) دستورات تعریف داده‌ها ۱۲۶
۱-۱-۴) دستور ایجاد پایگاه داده‌ها ۱۲۶
۲-۱-۴) دستور ایجاد جدول ۱۲۶
۳-۱-۴) ایجاد ایندکس ۱۲۸
۴-۱-۴) اضافه کردن یک ستون جدید به یک جدول ۱۲۹
۵-۱-۴) تغییر مشخصات یک ستون از یک جدول ۱۲۹
۶-۱-۴) حذف یک جدول ۱۲۹
۷-۱-۴) حذف یک ایندکس ۱۲۹
۲-۴) دستورات دستکاری داده‌ها ۱۳۰
۱-۲-۴) دستور انتخاب ۱۲۰
۲-۲-۴) دستور ایجاد دید خارجی یا دیدگاه ۱۴۲
۳-۲-۴) حذف یک دیدگاه ۱۴۴
۴-۲-۴) درج یک تاپل ۱۴۵
۵-۲-۴) اصلاح تاپلها ۱۴۵
۶-۲-۴) حذف تاپلها ۱۴۵
۳-۴) دستورات کنترل داده‌ها ۱۴۶
۱-۳-۴) دستور واگذاری مجوز ۱۴۶
۲-۳-۴) دستور باز پس‌گیری مجوز ۱۴۷
تمرینهای حل‌شده ۱۴۸
تمرینهای فصل ۱۷۸
فصل ۵: نرمال‌سازی ۲۰۵
۱-۵) جداول آنرمال ۲۰۶
۲-۵) جداول نرمال۱ ۲۰۶
۳-۵) جداول نرمال ۲ ۲۰۹

۴-۵) جداول نرمال ۳ و BCNF ۲۱۱
۱-۴-۵) جداول نرمال ۳ ۲۱۱
۲-۴-۵) جداول نرمال BCNF ۲۱۴
۵-۵) جداول نرمال ۴ ۲۱۷
۶-۵) جداول نرمال ۵ ۲۲۱
تمرینهای حل شده ۲۲۷
تمرینهای فصل ۲۳۱
فصل ٦ـ مدلسازی داده‌ها با استفاده از نمودارهای موجودیت رابطه (ERD) ۲۳۵
۱-۶) گروه‌بندی مدلهای داده براساس درجات تجرید داده‌ها ۲۳۶
۱-۱-۶) مدل انتزاعی ۲۳۶
۲-۱-۶) مدل داخلی ۲۳۸
۳-۱-۶) مدل خارجی ۲۳۹
۴-۱-۶) مدل فیزیکی ۲۴۰
۲-۶) نمودارهای ER ۲۴۰
۱-۲-۶) اجزاء اصلی نمودارهای ER ۲۴۰
۱-۱-۲-۶) موجودیتها ۲۴۰
۲-۱-۲-۶) ویژگیها ۲۴۰
۳-۱-۲-۶) رابطه‌ها ۲۴۳
۱-۳-۱-۲-۶) انواع رابطه‌ها ۲۴۳
۲-۳-۱-۲-۶) مفاهیم مربوط به رابطه‌ها ۲۴۶
۲-۲-۶) انواع موجودیتها ۲۴۸
۳-۲-۶) نوع و زیرنوعهای موجودیت ۲۵۲
۴-۲-۶) نگاشت یک نمودار ER به جداول پایگاه داده‌ها ۲۵۵
تمرینهای حل شده ۲۶۵
تمرینهای فصل ۲۸۱

۱

تاریخچه و مفاهیم پایگاه داده‌ها

در این فصل، تاریخچه و سیر تکاملی پایگاه داده‌ها را بررسی کرده، مدلهای پایگاه داده‌ها را مقایسه کرده و اجزاء و روند کار سیستمهای پایگاه داده‌ها را مورد بحث قرار می‌دهیم.

۱-۱) مفاهیم مربوط به پایگاه داده‌ها

قبل از تعریف پایگاه داده‌ها لازم است تفاوت میان داده[1] و اطلاع[2] را درک کنیم. داده، واقعیتی خام است که نمی‌تواند هیچ تأثیری در تصمیم‌گیریهای سیستم داشته باشد. به عنوان مثال، موارد ذیل سه داده در یک سیستم فروشگاه می‌باشند:

شمارهٔفاکتور = ۱۰۰۱

تاریخ فاکتور = ۸۳/۴/۴

مبلغ کل فاکتور = ۶۰۰۰۰

---

1 - Data

2 - Information

کاملاً مشخص است که هیچ یک از موارد فوق نمی‌توانند مدیران فروشگاه را در اتخاذ یک تصمیم یاری کنند. فرض کنید در این فروشگاه دو بخش فروش وجود داشته باشد. با انجام برخی محاسبات روی مبالغ کل فاکتورهای هر بخش می‌توان میزان فروش کل هر یک از بخشها را بدست آورده، از آن به عنوان معیاری برای مقایسهٔ بازدهی دو بخش استفاده کرد. فرض کنید انجام این مقایسه، مدیران فروشگاه را به این نتیجه برساند که میزان فروش بخش ۱ نسبت به میزان فروش بخش ۲، ۵۰ درصد بیشتر بوده است. درک این واقعیت می‌تواند به مدیران فروشگاه کمک کند تا در قبال دو بخش تصمیمهای مناسبی اتخاذ کنند. در واقع، این واقعیت که میزان فروش بخش ۱ نسبت به میزان فروش بخش ۲، ۵۰ درصد بیشتر بوده است یک اطلاع است. می‌توان چنین استنباط کرد که:

- داده‌ها، واحدهای سازندهٔ اطلاعات هستند.

و یا:

- اطلاع، از پردازش داده‌ها بدست می‌آید.

اطلاعات، مبنای تصمیم‌گیری در سازمانها هستند. از طرف دیگر، اطلاعات مفید و جامع از داده‌های خوب بدست می‌آیند. به همین دلیل، برای تولید، ذخیره و بازیابی صحیح داده‌ها مکانیزم موثری مورد نیاز است. چنین مکانیزمی مدیریت داده‌ها[1] نامیده می‌شود. آنچه مسلم است در دنیای کنونی برای مدیریت موثر داده‌ها باید از پایگاه داده‌های کامپیوتری استفاده شود. پایگاه داده‌ها، یک ساختار کامپیوتری یکپارچه[2] شامل داده‌های مورد نیاز کاربران نهایی[3] و فرا داده‌ها[4] می‌باشد که توسط کلیهٔ کاربران یک محیط عملیاتی (مثل دانشگاه، فروشگاه و ...) به صورت اشتراکی مورد استفاده قرار می‌گیرد.

برای درک تفاوت میان داده‌ها و فرا داده‌ها، محیط یک دانشگاه را در نظر بگیرید. در این سیستم داده‌هایی برای کاربران نهایی اهمیت دارند که یکی از موجودیتهای محیط عملیاتی (مانند

---

1 - Data Management
2 - Integrated
3 - End Users
4 - MetaData

دانشجو، استاد، وام و ...) و یا یکی از روابط محیط عملیاتی(مانند ثبت نام یک دانشجو در یک درس، اخذ وام توسط دانشجو و ...) را تشریح می‌کنند. علاوه بر این نوع داده‌ها، دستهٔ دیگری از داده‌ها وجود دارند که مربوط به هیچ موجودیت یا رابطهٔ عملیاتی نیستند. این دسته از داده‌ها که خود ساختار داده‌های اصلی را توصیف می‌کنند و یا برای سهولت و افزایش کارآیی کار با داده‌های اصلی مورد استفاده قرار می‌گیرند، فراداده نامیده می‌شوند. مثلاً یک فراداده ممکن است ساختار جدول دانشجو را تشریح کند یعنی نام، نوع و طول فیلدهای جدول دانشجو را مشخص کند.

۲-۱) تاریخچهٔ پایگاه داده‌ها

۱-۲-۱) ریشه‌های تاریخی پایگاه داده‌ها: سیستمهای فایل

قبل از اختراع کامپیوتر، شرکتها و سازمانها از سیستمهای دستی برای نگهداری اطلاعات خود استفاده می‌کردند. در این سیستمها هر دسته از اطلاعات در پوشهٔ مجزایی نگهداری می‌شد. مثلاً در یک دانشگاه، اطلاعات مربوط به دانشجویان در یک پوشه، اطلاعات مربوط به اساتید در یک پوشه، اطلاعات مربوط به ثبت نام در یک پوشه و ... طبقه‌بندی می‌شد.

با رشد روزافزون حجم اطلاعات در سازمانها و شرکتها و همچنین، نیاز به تهیهٔ گزارشات دقیق و متنوع که فراتر از حد توان فکری و محاسباتی یک اپراتور و خارج از وقت و حوصلهٔ مدیران بود، چنین به نظر می‌رسید که سیستمهای دستی پاسخگوی نیازهای اطلاعاتی نیستند. خوشبختانه با اختراع کامپیوتر و ترویج استفادهٔ همه‌گیر از آن، راه حلی برای این مسئله پیدا شد و سیستمهای دستی با سیستمهای فایل جایگزین شدند. به این ترتیب، در هر سازمان، اطلاعات هر پوشه در یک فایل کامپیوتری ذخیره شد و برنامه‌نویسان، با استفاده از زبانهای نسل سوم مانند کوبول و بیسیک برنامه‌هایی برای دستکاری اطلاعات موجود در فایلها و همچنین، تهیهٔ گزارشات براساس این اطلاعات نوشتند. مهمترین مشخصه و در واقع عیب سیستمهای فایل، عدم یکپارچگی آنها بود. در این سیستمها هر قسمت از سازمان سیستم فایل مجزایی داشت. به عنوان مثال، در یک دانشگاه قسمت آموزش از فایلها و برنامه‌های مجزا و قسمت امور مالی از فایلها و برنامه‌های مجزا استفاده می‌کرد . اگرچه ممکن بود بسیاری از فایلهای مورد استفاده میان دو قسمت مشترک باشد، این دو قسمت از هیچ فایلی به صورت مشترک استفاده نمی‌کردند مثلاً هر یک از آنها یک نسخهٔ مجزا از

فایل دانشجویان را در اختیار داشتند. (در واقع، دو کپی از فایل دانشجویان مورد استفاده قرار می‌گرفت یکی در قسمت آموزش و دیگری در قسمت امور مالی)

۲-۲-۱) مفاهیم مربوط به سیستمهای فایل

- **داده**[1]: عبارت است از حقیقتی خام که هیچ مفهوم خاصی ندارد مثلاً وقتی از عدد ۵ صحبت می‌کنیم، مشخص نمی‌شود این ۵ تعداد فرزندان یک کارمند است یا وزن یک جسم یا سنوات تدریس یک استاد یا ...

تذکر: مفهوم داده در سیستمهای فایل با مفهوم داده در حالت کلی تفاوت دارد.

- **فیلد**[2]: گروهی از کاراکترهاست که خصوصیتی از یک موجودیت را توصیف می‌کنند مثل age و name و weight. هنگامیکه داده‌ای در داخل یک فیلد ریخته می‌شود، مفهوم و معنا پیدا می‌کند مثلاً اگر عدد ۵ را داخل فیلد weight بریزیم، وزن یک موجودیت را مشخص می‌کند.
- **رکورد**[3]: مجموعه‌ای از فیلدهاست که مربوط به یک موجودیت خاص هستند. مثلاً رکورد یک دانشجو می‌تواند به شکل ذیل باشد:

Stno	Name	Course
۷۸۰۱	آرش راد	ریاضی

فایل[4]: مجموعه‌ای از رکوردهاست که مربوط به یک نوع موجودیت هستند. مثلاً فایل دانشجویان یک دانشگاه می‌تواند به شکل ذیل باشد:

---

1 - Data
2 - Field
3 - Record
4 - File

Stno	Name	Course
۷۸۰۱	علی راد	هنر
۷۸۰۹	علی تقوی	ریاضی
۷۹۰۱	مینا رسولی	ریاضی
۷۹۰۵	آرش رضوی	هنر

- **سیستم فایل**[1]: مجموعه‌ای از فایلها و برنامه‌های لازم برای کار با آنهاست. مثلاً سیستم فایل یک دانشگاه شامل:

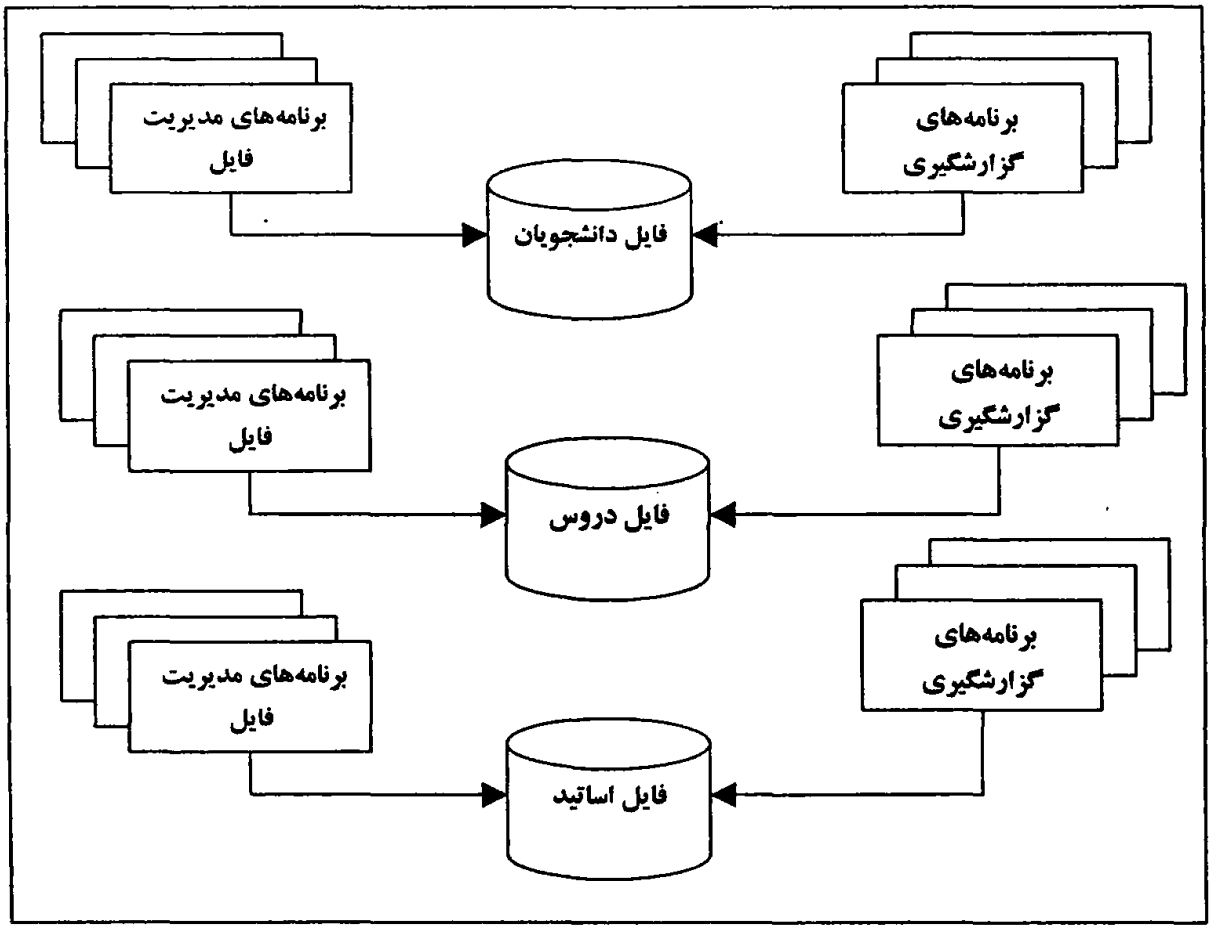
- فایل دانشجویان و برنامه‌های مدیریت این فایل (برنامه‌هایی که برای درج، حذف و یا تغییر اطلاعات دانشجویان بکار می‌روند) و برنامه‌های مورد استفاده برای تهیهٔ گزارشات براساس اطلاعات این فایل (تهیهٔ لیستی از دانشجویان ممتاز، تهیهٔ لیستی از دانشجویان اخراجی و یا مشروط، تهیهٔ لیستی از دانشجویان ورودی ۷۹ و ...)

- فایل دروس و برنامه‌های مربوط به مدیریت این فایل (درج، حذف و یا تغییر اطلاعات دروس) و برنامه‌های مورد استفاده برای تهیهٔ گزارشات لازم براساس اطلاعات دروس (لیستی از دروس عملی، لیستی از دروس تخصصی کامپیوتر و ...)

- فایل اساتید و برنامه‌های مدیریت و گزارشگیری مربوط به این فایل و ... می‌باشد.

شکل ذیل، سیستم فایل یک دانشگاه را نشان می‌دهد:

1 - File System



۳-۲-۱) معایب سیستمهای فایل

سیستمهای فایل، در کنار کلیهٔ محاسنی که نسبت به سیستمهای دستی دارا بودند، معایب زیادی داشتند که مهمترین آنها عبارت بودند از:

۱- نیاز به برنامه‌نویسی زیاد و پیچیده

در سیستمهای فایل برای نوشتن ساده‌ترین برنامه‌ها که با فایلها در ارتباط بودند، لازم بود برنامه‌نویس از جزئیات نحوهٔ ذخیره‌سازی فایلها کاملاً آگاه بوده و برای انجام کار مورد نظر قطعه کدی طولانی و پیچیده بنویسد.

۲- وابستگی داده‌ای[1]

فرض کنید در فایل دانشجویان، برنامه‌نویس حداکثر ۲۰ کاراکتر را به فیلد name اختصاص داده باشد و پس از مدتی بخواهد طول این فیلد را به ۳۰ کاراکتر تغییر دهد. در این صورت، علاوه بر آنکه مجبور است اطلاعات موجود در فایل قدیمی را به یک فایل جدید با ساختار جدید منتقل کند(که البته خود این کار نیز ساده نمی‌باشد)، مجبور است کد کلیهٔ قسمتهایی از برنامه که با فایل دانشجویان مرتبط می‌باشند را نیز تغییر دهد چرا که برنامه‌ها کاملاً به ساختار فیزیکی فایل وابسته‌اند. در واقع، وابستگی داده‌ای به "لزوم تغییر کد برنامه‌های مربوط به یک فایل پس از اعمال یک تغییر در مشخصات یکی از فیلدهای فایل(مثل تغییر نوع یک فیلد از String به integer و یا تغییر طول یک فیلد)" اطلاق می‌شود. در سیستمهای فایل، وابستگی داده‌ای به شدت زیاد است.

۳- وابستگی ساختاری[2]

فرض کنید برنامه‌نویس بخواهد در فایل دانشجویان که مدتها مورد استفاده بوده است، یک فیلد جدید برای ذخیرهٔ شمارهٔ شناسنامه اضافه کند و یا یکی از فیلدهای موجود در آن را حذف کند. در این صورت، علاوه بر مشکلات مربوط به انتقال اطلاعات از فایل قدیمی به فایلی با ساختار جدید، برنامه‌نویس مجبور خواهد بود کلیهٔ برنامه‌هایی که به نحوی به فایل دانشجویان مربوط می‌شوند را نیز تغییر دهد. در واقع، وابستگی ساختاری به "لزوم تغییر کد برنامه‌های مربوط به یک فایل پس از اعمال هر تغییر در ساختار فایل (اضافه کردن یا حذف فیلد)" اطلاق می‌شود.

٤- افزونگی داده‌ها[3]

همان گونه که قبلاً نیز به این نکته اشاره شد، هر یک از بخشهای سازمانهایی که از سیستمهای فایل استفاده می‌کردند، سیستم فایل مجزایی داشتند و حتی در صورت مشترک بودن داده‌های مورد استفاده، بخشهای مختلف سازمان از فایلها به صورت اشتراکی استفاده نمی‌کردند مثلاً قسمت آموزش یک دانشگاه از یک نسخه از فایل دانشجویان و قسمت امور مالی از نسخهٔ دیگری از فایل

---

1 - Data Dependence
2 - Structural Dependence
3 - Data Redundency

دانشجویان که عیناً مشابه هم بودند استفاده می‌کردند. به عبارت دیگر، یک قلم داده در چندین جای مختلف ذخیره می‌شد. این افزونگی یا تکرار داده‌ها، سیستم فایل را با مشکلاتی مواجه می‌کرد:

- هدر رفتن فضا و نیروی کار

ذخیرهٔ یک قلم داده در چند جای مختلف علاوه بر آنکه فضای حافظهٔ زیادی را به هدر می‌داد، باعث به هدر رفتن نیروی کاری نیز می‌شد. مثلاً برای وارد کردن و اصلاح مداوم اطلاعات مربوط به دانشجویان یک دانشگاه دو اپراتور مورد نیاز بود: یک اپراتور برای درج و اصلاح اطلاعات دانشجویان در قسمت آموزش و دیگری برای درج و اصلاح اطلاعات دانشجویان در قسمت امور مالی.

- ناسازگاری داده‌ها[1]

فرض کنید یکی از دانشجویان به نام علی راد، قسمت امور مالی را از تغییر شماره تلفن خود آگاه کند. در این صورت، اگر اپراتور امور مالی شماره تلفن وی را اصلاح کند و اپراتور قسمت آموزش این تغییر را اعمال نکند، برای فیلد شماره تلفن این دانشجو در دو فایل مختلف دو مقدار متفاوت وجود خواهد داشت. (می‌توانید تصور کنید اگر اپراتور قسمت امور مالی و اپراتور قسمت آموزش به مدت یکماه با یکدیگر قهر باشند چه اتفاقی می‌افتد!!!) در واقع، ناسازگاری داده‌ها به "وجود دو یا چند مقدار متفاوت برای یک قلم داده‌ای" اطلاق می‌شود. وجود افزونگی ذاتاً عامل مهمی برای بروز ناسازگاری داده‌ها می‌باشد.

- بروز انواع ناهنجاری

افزونگی داده‌ها، باعث بروز سه نوع ناهنجاری می‌شود. برای درک انواع ناهنجاری، سیستم یک دانشگاه را در نظر بگیرید. قسمت امور مالی این دانشگاه، فایلی به نام Tutor دارد که حاوی اطلاعات اساتید دانشگاه است:

---

1 - Data Inconsistency

Tutor	Tname	Ttel
۱۰۰	آرش راد	۶۲۴۴۴۴
۱۰۱	مینا رضوی	۶۱۵۲۵۲
۱۰۲	عسل رسولی	۲۱۹۳۹۴

از طرف دیگر، قسمت آموزش از فایلی بنام Project برای درج اطلاعات پروژه‌های فارغ‌التحصیلی دانشجویان استفاده می‌کند و چون به فایلهای موجود در قسمت امور مالی دسترسی ندارد، مجبور است اطلاعات استاد راهنمای پروژه را نیز در این فایل نگهداری کند.

St #	Proje-title	Tutor	Tname	Ttel
۷۸۰۱	طراحی و پیاده‌سازی یک سایت خرید و فروش اینترنتی	۱۰۰	آرش راد	۶۲۴۴۴۴
۷۹۰۲	طراحی وپیاده‌سازی یک سیستم ثبت نام دانشگاه	۱۰۲	عسل رسولی	۲۱۹۳۹۴
۸۰۰۱	بهینه سازی پرس وجوها در پایگاه داده‌ها	۱۰۰	آرش راد	۶۲۴۴۴۴

سازماندهی فایلها به صورت فوق می‌تواند باعث بروز ناهنجاریهای ذیل شود:

***ناهنجاری اصلاح**[1]: فرض کنید آرش راد شماره تلفن خود را به ۶۲۷۷۷۷ تغییر دهد. در این صورت، برای جلوگیری از بروز ناسازگاری لازم است هر دو قسمت امور مالی و آموزش این تغییر را اعمال کنند. بدیهی است قسمت آموزش برای اعمال این تغییر مجبور است کلیهٔ رکورد های مربوط به پروژه‌های اخذ شده با آقای آرش راد را تغییر دهد. به عبارت دیگر، اگر آقای آرش راد تاکنون راهنمایی ۱۰۰ پروژه را برعهده گرفته باشد، شماره تلفن وی در هر یک از این ۱۰۰ رکورد باید تغییر کند چرا که در غیر این صورت شماره تلفن وی در رکوردهای مختلف دارای مقادیر متفاوت خواهند بود. این نوع اصلاح به اصلاح منتشر شونده[2] معروف است.

1 - Modification Anomaly
2 - Propagating Modification

* **ناهنجاری درج**[1]: برای درج هر پروژهٔ جدید در فایل Project لازم است اطلاعات استاد راهنمای مربوطه نیز درج شود حتی اگر قبلاً اطلاعات وی در پروژه‌های دیگر درج شده باشد. این مسئله علاوه بر هدر دادن فضا و نیروی کاری، خطر بروز ناسازگاری را نیز افزایش می‌دهد چرا که در ورود دوبارهٔ اطلاعات استاد توسط اپراتور امکان خطا وجود دارد.

* **ناهنجاری حذف**[2]: فرض کنید قسمت امور مالی اطلاعات مربوط به آقای آرش راد را از فایل Tutor حذف کند. در این صورت، در فایل Project برخی از دانشجویان استاد راهنمایی خواهند داشت که دیگر وجود ندارد. پس برای جلوگیری از ناسازگاری داده‌ها قسمت آموزش باید کلیهٔ پروژه‌هایی را که با این استاد اخذ شده‌اند، حذف کرده یا اطلاعات فیلدهای مربوط به استاد را در این رکوردها خالی و یا با اطلاعات استاد دیگری جایگزین کند.

از طرف دیگر، فرض کنید دانشجوی ۷۹۰۲ تنها دانشجویی باشد که با عسل رسولی پروژه گرفته است. در این صورت با حذف این دانشجو، قسمت آموزش، اطلاعات مربوط به عسل رسولی را نیز از دست می‌دهد. در واقع، با حذف یک قلم اطلاعاتی، به طور ناخواسته اطلاعات دیگری نیز حذف می‌شود.

۵-عدم وجود امکانات لازم برای تأمین امنیت داده‌ها[3]

سیستمهای فایل فاقد امکانات لازم برای تعیین حدود اختیارات هر کاربر بودند. به عبارت دیگر، در این سیستمها هر کاربر با هرسمت و اختیاراتی می‌توانست به کلیهٔ داده‌های ذخیره شده در فایلها دسترسی داشته باشد. به راحتی می‌توانید خطرات ناامنی در دسترسی به داده‌ها را درک کنید!

---

1 - Insertion Anomaly
2 - Deletion Anomaly
3 - Data Security

۳-۱) سیستمهای پایگاه داده‌ها[1]:

وجود معایب متعدد در سیستمهای فایل باعث شد که این سیستمها با سیستمهای پایگاه داده‌ها جایگزین شوند. در سیستمهای پایگاه داده‌ها برخلاف سیستمهای فایل، کلیهٔ داده‌ها به صورت یکپارچه و تنها در یک محل ذخیره می‌شوند و کلیهٔ کاربران می‌توانند به صورت اشتراکی و همزمان از این داده‌ها استفاده کنند. در سیستمهای پایگاه داده‌ها، برخلاف سیستمهای فایل هیچ یک از کاربران به صورت مستقیم به داده‌ها دسترسی ندارند بلکه درخواستهای خود را در قالب یک دستور سطح بالا به یک نرم افزار از پیش ساخته شده بنام نرم افزار مدیریت پایگاه داده‌ها یا DBMS[2] تحویل می‌دهند. در واقع، DBMS نقش واسط و مترجم میان کاربران(یا برنامه‌های کاربردی[3] آنها) و پایگاه داده‌ها (شامل داده‌های مورد نیاز کاربران و فراداده‌ها) را بازی می‌کند و بسیاری از وظایفی را که در سیستمهای فایل بر عهدهٔ کاربران بود، خود به تنهایی به دوش می‌کشد. در سیستمهای پایگاه داده‌ها، نیازی به نوشتن صدها خط کد برای تهیهٔ یک گزارش ساده و یا دستکاری داده‌ها نیست. کاربر و یا برنامهٔ کاربردی که توسط برنامه‌نویس نوشته شده است (مانند سیستم حسابداری، سیستم حقوق و دستمزد، سیستم ثبت نام دانشگاه و ...) به منزلهٔ یک رئیس که دستورات سطح بالا صادر می‌کند و علاقه‌ای به اطلاع از نحوهٔ اجرای این دستورات ندارد عمل می‌کند. DBMS ، هر دستور سطح بالای کاربر یا برنامهٔ کاربردی را به کدهای پیچیدهٔ لازم تبدیل کرده، این کدهای پیچیده را روی پایگاه داده‌ها اعمال کرده، نتیجهٔ دریافتی را به فرمتی قابل فهم برای کاربر یا برنامهٔ کاربردی وی تبدیل کرده و در اختیار او قرار می‌دهد. به عنوان مثال فرض کنید کاربری بخواهد نام دانشجویان ممتاز را بدست آورد. کاربر می‌تواند دستور خود را مستقیماً به DBMS بدهد یا از قسمتی از برنامهٔ آموزش دانشگاه که توسط برنامه‌نویس و به همین منظور نوشته شده است استفاده کند. در هر صورت دستوری که توسط کاربر و یا برنامهٔ کاربردی به DBMS داده می‌شود، می‌تواند به سادگی دستور ذیل باشد:

```
select name from student where ave>17
```

---

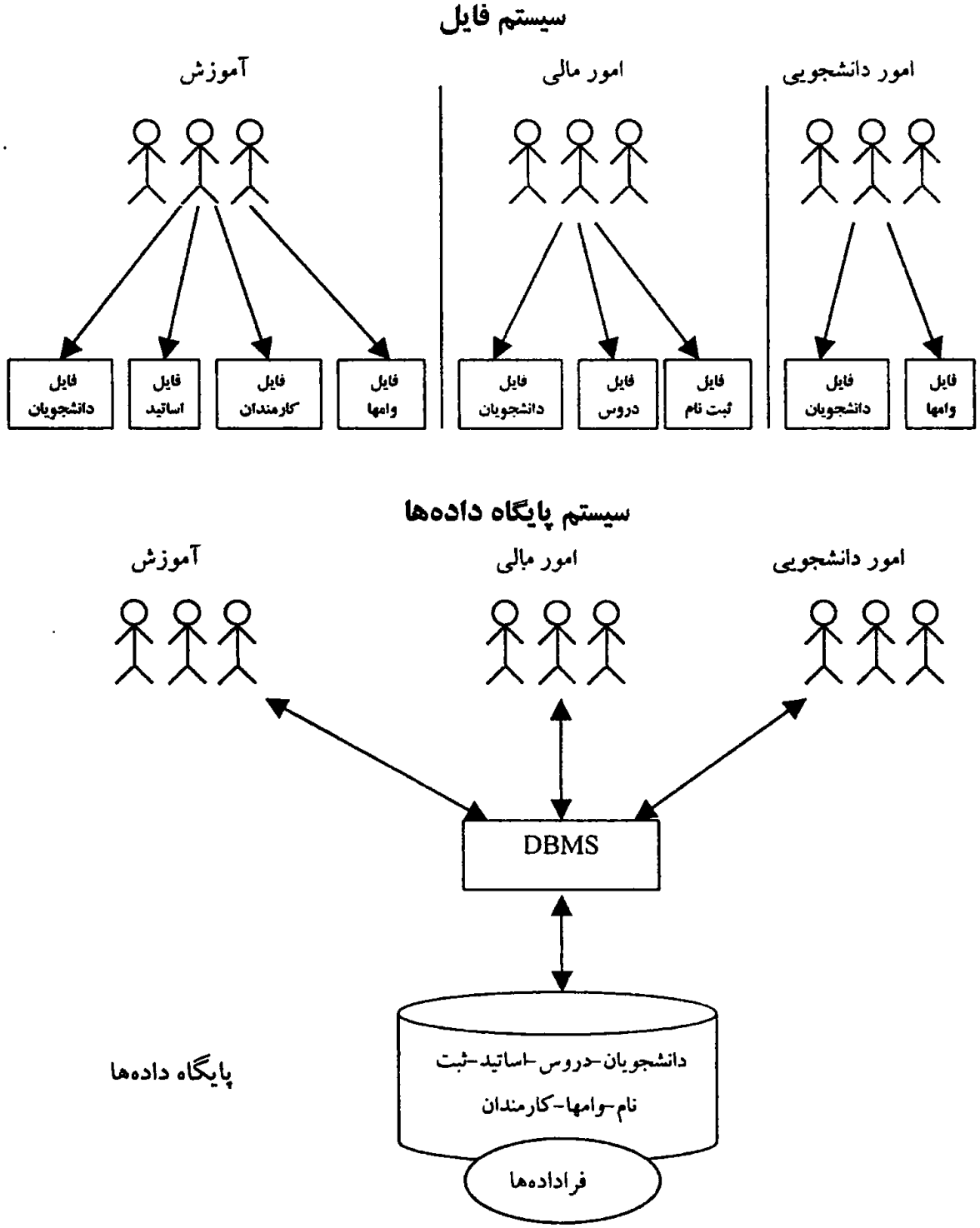
1 - Database Systems

2 - DataBase Management System

3 - Applications

DBMS، این دستور را به کدهای پیچیدهٔ لازم ترجمه کرده، این کدها را روی داده‌های موجود در پایگاه داده‌های دانشگاه اجرا کرده و نتیجه را در قالب یک جدول ساده در اختیار کاربر و یا برنامهٔ کاربردی مورد استفاده توسط وی قرار می‌دهد.

شکل ذیل یک سیستم فایل را با سیستم پایگاه داده‌ها مقایسه می‌کند:



٤-١) اجزاء یک سیستم پایگاه داده‌ها:

یک سیستم پایگاه داده‌ها شامل اجزاء ذیل است:

۱- سخت افزار: سخت افزار یک سیستم پایگاه داده‌ها شامل کلیهٔ وسایل فیزیکی است که برای ورود/خروج (I/O) و ذخیرهٔ اطلاعات بکار می‌روند. به عبارت ساده‌تر، سخت‌افزار یک سیستم پایگاه داده‌ها شامل کامپیوترهای مورد استفاده در محیط می‌باشد. در صورتی که از پایگاه داده‌ها در یک محیط شبکه‌ای استفاده شود، کلیهٔ تجهیزات مربوط به شبکه نیز جزء سخت افزار پایگاه داده‌ها محسوب می‌شوند.

۲- نرم افزار: در سیستمهای پایگاه داده‌ها از سه نوع نرم افزار استفاده می‌شود:

-**سیستم عامل:** مسلماً قبل از نصب نرم افزار DBMS لازم است روی کلیهٔ کامپیوترهای محیط پایگاه داده‌ها یک سیستم عامل مثل ویندوز یا UNIX یا ... نصب شود.

-**DBMS:** همان گونه که قبلاً نیز به این نکته اشاره کردیم، DBMS نرم افزاری از پیش تهیه شده و پیچیده است که برنامه‌نویسان و طراحان هیچگونه دخل و تصرفی در عملکرد آن ندارند بلکه نسخه‌های مختلف آن را خریداری کرده، بر روی کامپیوترهای محیط موردنظر نصب کرده و از امکانات آن برای ایجاد پایگاه داده‌ها ، پشتیبانی پایگاه داده‌ها و انجام عملیات گوناگون روی داده‌ها استفاده می‌کنند. SQL SERVER، Oracle، DB2 و Informix از معروفترین DBMS-های کنونی هستند.

-**برنامه‌های کاربردی:** برنامه‌های کاربردی برنامه‌هایی هستند که توسط برنامه‌نویسان و مخصوص یک محیط عملیاتی نوشته می‌شوند مثل برنامه‌های حسابداری، برنامه‌های حقوق و دستمزد و ... این برنامه‌ها از طریق DBMS با پایگاه داده‌ها ارتباط برقرار کرده، اطلاعات موجود در آن را مورد دستکاری قرار داده و یا بر اساس اطلاعات موجود در پایگاه داده‌ها گزارشاتی تهیه می‌کنند.

۳- **کاربران**[1]: کاربران سیستمهای پایگاه داده‌ها را می‌توان به پنج دستهٔ اصلی تقسیم کرد:

-**مدیران سیستم**[2]: این گروه از کاربران بر عملکرد کلی سیستم پایگاه داده‌ها نظارت می‌کنند.

-**مدیران پایگاه داده‌ها یا DBA**[3]**-ها:** این دسته از مدیران که نقش مهمی در سیستمهای پایگاه داده‌ها ایفا می‌کنند، استفادهٔ بهینه و درست و کارآمد از پایگاه داده‌ها را مورد بررسی قرار می‌دهند. در واقع، DBA-ها خط مشی و سیاستهای کلی استفاده و کار با پایگاه داده‌ها را مشخص می‌کنند مثلاً تعیین می‌کنند که از چه DBMS-یی استفاده شود و یا چگونه نسخه‌های جدید DBMS در محیط پایگاه داده‌ها نصب شوند و یا برای بهینه سازی کارآیی پایگاه داده‌ها از چه استانداردها و مکانیزمهایی استفاده شود.

-**طراحان پایگاه داده‌ها**[4]: این دسته از کاربران، در واقع معماران پایگاه داده‌ها هستند. این گروه از کاربران، پایگاه داده‌ها را مطابق با خواسته‌های مدیران پایگاه داده‌ها و بر اساس نیازهای کلیهٔ کاربران نهایی طراحی می‌کنند.

-**برنامه‌نویسان:** برنامه‌نویسان، برنامه‌های کاربردی لازم برای کار با پایگاه داده‌ها را مطابق با نیازمندیهای کاربران طراحی و پیاده‌سازی می‌کنند. در واقع وظیفهٔ آنها طراحی و ایجاد صفحات ورود داده‌ها و گزارشات مورد نیاز کاربران نهایی و همچنین، پروسیجرهایی است که کاربران نهایی از آنها برای دسترسی به داده‌ها و دستکاری آنها استفاده می‌کنند.

---

1 - Users
2 - System Administrators
3 - DataBase Administrators
4 - Database Designers

-**کاربران نهایی**[1]: کاربران نهایی اپراتورها و مدیران سازمانها هستند. این دسته از کاربران، اطلاعات زیادی در مورد پایگاه داده‌ها و یا برنامه‌نویسی ندارند و تنها برای انجام امور مربوط به خود و استفاده از برنامه‌های کاربردی نوشته شده توسط برنامه‌نویسان آموزش می‌بینند.

٤- **پروسیجرها**[2]: پروسیجرها شامل دستوراتی هستند که برای اجرای قوانین حاکم بر سیستم نوشته می‌شوند. به عنوان مثال، در یک سیستم فروشگاه به محض صدور فاکتور برای مشتری باید موجودی کالاهای خریداری شده توسط مشتری به روز رسانده شود. برای انجام این کار معمولاً از یک پروسیجر استفاده می‌شود.

٥- **داده‌ها**: همان گونه که قبلاً نیز در این مورد توضیح داده شد، داده‌های یک سیستم پایگاه داده‌ها شامل داده‌های موردنیاز کاربران نهایی و فراداده‌ها می‌باشند.

٥-١) انواع سیستمهای پایگاه داده‌ها

DBMS هستهٔ اصلی هر سیستم پایگاه داده‌ها است. DBMS-ها را براساس تعداد کاربران، تمرکز و یا توزیع شدگی داده‌ها و نوع کاربرد می‌توان به چند دسته تقسیم کرد.

چنانچه در هر لحظه تنها یک کاربر اجازهٔ کار با پایگاه داده‌ها را داشته باشد، DBMS را تک کاربره[3] نامند. در یک سیستم پایگاه داده‌های تک کاربره چنانچه کاربر A مشغول کار با پایگاه داده‌ها باشد، کاربران B و C باید منتظر بمانند تا کار کاربر A به اتمام برسد و سپس به نوبت با پایگاه داده‌ها کار کنند. چنانچه DBMS تک کاربره روی یک کامپیوتر شخصی (PC)، اجرا شود آن را پایگاه دادهٔ رومیزی[4] نامند. در مقابل، DBMS-های چند کاربره به کاربران مختلف اجازه می‌دهند به صورت همزمان با پایگاه داده‌ها کار کنند. چنانچه تعداد مجاز برای استفادهٔ همزمان از

---

1 - End Users
2 - Procedures
3 - Sirgle-User
4 - Desktop Database

پایگاه داده‌ها کمتر از ۵۰ کاربر در هر لحظه باشد، پایگاه داده‌ها را پایگاه داده‌های گروه کاری[1] و چنانچه توسط کل بخشهای یک سازمان مورد استفاده قرار گیرد (غالباً در چنین حالتی صدها کاربر در هر لحظه با پایگاه داده‌ها کار می‌کنند) پایگاه داده‌های شرکتی[2] نامیده می‌شود.

نحوهٔ توزیع داده‌ها نیز یکی از معیارهای طبقه‌بندی پایگاههای داده‌هاست. DBMS-هایی که اجازه نمی‌دهند داده‌ها روی چند سایت پخش شوند، DBMS-های متمرکز[3] و DBMS-هایی که امکان توزیع داده‌ها روی چند سایت مختلف را فراهم می‌کنند، DBMSهای توزیع شده[4] نامیده می‌شوند.

برای درک مفهوم پایگاههای دادهٔ توزیع شده فرض کنید قسمتی از ویژگیهای دانشجویان در سایت آموزش و قسمتی دیگر در سایت امور مالی ذخیره شده باشد و یا اطلاعات برخی از مشتریان یک بانک روی یک کامپیوتر و اطلاعات مشتریان دیگر روی یک کامپیوتر مجزا در یک شعبهٔ دیگر ذخیره شده باشد.

موارد کاربرد پایگاه داده‌ها نیز یکی از معیارهای گروه بندی پایگاههای داده‌ها می‌باشد. پایگاههای داده و یا به طور دقیقتر DBMS-ها را می‌توان بر اساس کاربرد آنها به دو دستهٔ تراکنش‌گرا[5] و تصمیم‌گیرنده[6] تقسیم کرد. DBMS- هایی که در سیستمهای مبتنی بر تراکنش[7] مانند سیستمهای بانک، فروش و... مورد استفاده قرار می‌گیرند DBMS-های تراکنش‌گرا نامیده می‌شوند. در این سیستمها لازم است در واکنش به عملی از طرف کاربر، به سرعت پاسخی تولید شده و یا عملیاتی صورت گیرد مثلاً در یک سیستم فروش به محض فروش یک کالا لازم است موجودی کالای موردنظر در انبار به روز رسانده شود. در مقابل، وظیفهٔ یک DBMS تصمیم‌گیرنده، تولید اطلاعاتی است که مدیران رده بالای سازمان را در اتخاذ تصمیمهای استراتژیک یاری می‌کنند. به عنوان مثال، یک DBMS تصمیم‌گیرنده ممکن است داده‌های فراوانی را از منابع مختلف دریافت

---

1 - Work-group Database
2 - Enterprise Database
3 - Centralized DBMS
4 - Distributed DBMS
5 - Transactional
6 - Decision Support
7 - Transaction

کرده و فرمول مناسب برای تضمین قیمت یک محصول را استخراج کند و یا محل سرمایه‌گذاری بهینه را تعیین کند و یا وضعیت آب و هوا را پیش‌بینی کند. آنچه در سیستمهای تراکنش‌گرا اهمیت دارد سرعت پاسخگویی و واکنش DBMS است. در واقع، در این سیستمها لازم است DBMS به طور لحظه‌ای نسبت به ورود و یا دستکاری داده‌ها واکنش نشان دهد ولی در پایگاههای دادهٔ تصمیم‌گیرنده، آنچه حائز اهمیت است دقت و صحت اطلاعات تولید شده توسط DBMS است و نه سرعت تولید آنها.

انتخاب DBMS مناسب برای هر محیط عملیاتی کاملاً به سه عامل فوق بستگی دارد. بی‌شک یک انتخاب درست می‌تواند عملکرد سیستم پایگاه داده‌ها را کاملاً تحت الشعاع قرار دهد.

۶-۱) وظایف DBMS:

DBMS در راستای تضمین سازگاری و جامعیت داده‌ها[1] وظایف مهمی برعهده دارد. جزئیات انجام این وظایف از دید کاربر پنهان است و DBMS اجازه نمی‌دهد کاربر با جزئیات و پیچیدگیهای انجام این وظایف درگیر شود. مهمترین این وظایف عبارتند از:

۱- **تأمین استقلال داده‌ای و ساختاری:** DBMS به طور اتوماتیک یک دیکشنری داده[2] برای پایگاه داده‌ها تهیه می‌کند و مشخصات کلیهٔ داده‌ها (اعم از داده‌های مربوط به موجودیتها و روابط میان موجودیتها) را در این دیکشنری ذخیره می‌کند. قبل از رجوع به هر قلم داده‌ای، DBMS به دیکشنری داده‌ها رجوع کرده و از مشخصات ساختاری آن آگاه می‌شود. مثلاً اگر کاربر از DBMS بخواهد کلیهٔ اطلاعات موجود در جدول Student را استخراج کند، DBMS ابتدا به دیکشنری داده مراجعه کرده، از چند و چون ساختار این جدول مطلع می‌شود. چنانچه طراح پایگاه داده‌ها فیلد جدیدی با نام address و طول ۵۰ کاراکتر را در جدول Student اضافه کند، این تغییر در دیکشنری داده‌ها منعکس می‌شود. در نتیجه در مراجعات بعدی به جدول Student، از آنجا که

1 - Data Integrity
2 - Data Dictionary

DBMS فیلد address را نیز جزء فیلدهای جدول Student می‌بیند، می‌داند چگونه با جدول اصلی که اکنون شامل فیلد address نیز هست برخورد کند. به همین ترتیب اگر طراح فیلد name از جدول Student را از ۲۰ کاراکتر به ۳۰ کاراکتر تغییر دهد، این تغییر در دیکشنری داده‌ها منعکس می‌شود. در نتیجه DBMS در مراجعات بعدی به جدول Student و تنها با مشاهدهٔ دیکشنری داده متوجه می‌شود که برای فیلد name باید ۳۰ کاراکتر در نظر بگیرد. به این ترتیب، نیازی به تغییر کد برنامه‌هایی که با جدول Student سر و کار دارند نمی‌باشد. در واقع، DBMS با ایجاد دیکشنری داده، غالباً باعث مصونیت کدهای برنامه‌نویسی از تغییرات انجام شده در ساختارهای پایگاه داده‌ها می‌شود و با این کار خدمت بزرگی در حق برنامه‌نویسان می‌کند!!!

۲- **مدیریت ذخیره‌سازی داده‌ها**[1]: DBMS ساختارهای پیچیدهٔ لازم برای ذخیرهٔ داده‌ها را خود ایجاد کرده و طراحان پایگاه داده‌ها را از وظیفهٔ سنگین تعریف خصوصیات فیزیکی داده‌ها معاف می‌کند. اکثر DBMS-های کنونی نه تنها ذخیره‌سازی داده‌ها را خود مدیریت می‌کنند، بلکه ابزارهایی در اختیار کاربران خود قرار می‌دهند تا فرمهای لازم برای ورود داده‌ها را طراحی کرده، گزارشات مورد نیاز خود را تعریف کرده، قوانین اعتبارسنجی[2] محیط عملیاتی مورد نظر را تعیین کرده(مثلاً در محیط دانشگاه، بازهٔ قابل قبول برای نمرهٔ دانشجو عددی بین ۰ تا ۲۰ است. طراح پایگاه داده‌ها به راحتی می‌تواند این قانون را به DBMS ابلاغ کند. پس از تعریف این قانون چنانچه کاربری بخواهد نمره‌ای خارج از محدودهٔ مجاز را وارد کند، DBMS از انجام عمل مورد نظر وی سر باز می‌زند)، نحوهٔ ورود تصاویر و ویدئوها در پایگاه داده‌ها را مشخص کند و ...

1 - Data Storage
2 - Validation Rules

۳- **تبدیل فرمت داده‌ها میان محیط فیزیکی و محیط منطقی:** فرمت و نحوهٔ ذخیره‌سازی داده‌ها روی محیط فیزیکی ذخیره‌سازی با فرمت داده‌های موردنظر کاربران بسیار متفاوت است. DBMS وظیفه دارد داده‌های وارد شده توسط کاربران را به فرمت قابل قبول برای ذخیره‌سازی روی محیط فیزیکی تبدیل کند. از طرف دیگر، فرمت و ساختار داده‌هایی که DBMS از روی محیط فیزیکی جمع‌آوری می‌کند، برای کاربران قابل درک نیست. DBMS وظیفه دارد هنگام بازیابی داده‌ها از محیط فیزیکی، آنها را به فرمت قابل فهم توسط کاربران تبدیل کرده و سپس داده‌ها را در اختیار کاربران قرار دهد.

٤- **مدیریت امنیت:** در سیستمهای پایگاه داده‌های کنونی امنیت از اهمیت ویژه‌ای برخوردار است. امنیت در پایگاه داده‌ها به این معناست که هر کاربر تنها در حوزهٔ اختیارات و وظایف خود به داده‌ها دسترسی داشته باشد. مثلاً در محیط دانشگاه یک کارمند امور مالی نباید به نمرات دانشجویان دسترسی داشته باشد. یکی از وظایف اصلی مدیر پایگاه داده‌ها این است که با انجام مصاحبه با مدیران سازمان از حدود اختیارات هر کاربر آگاه شده، با توجه به این اختیارات مجوزهای تک تک کاربران را برای DBMS مشخص کند مثلاً ممکن است طراح برای DBMS مشخص کند که کاربر یا کاربرانی که شناسهٔ آنها mali است(هر کاربر یک شناسه و کلمهٔ عبور دارد) اجازهٔ مشاهدهٔ کلیهٔ فیلدهای جداول Student و Vam را دارند ولی تنها فیلد amount از جدول Vam را می‌توانند تغییر دهند. هنگامیکه کاربر تقاضای انجام عملیاتی را صادر می‌کند، DBMS کنترل می‌کند آیا کاربر مورد نظر مجوز انجام چنین عملیاتی را دارد یا خیر؟ چنانچه کاربر مجوز انجام عملیات را نداشته باشد، DBMS از انجام عملیات درخواست شده توسط وی خودداری می‌کند.

٥- **تأمین امکان دسترسی مشترک چندین کاربر به پایگاه داده‌ها:** DBMS با استفاده از الگوریتمهای پیچیده این امکان را به وجود می‌آورد که چندین کاربر به

صورت همزمان و مشترک از داده‌های موجود در پایگاه داده‌ها استفاده کنند بدون آنکه در پایگاه داده‌ها ناسازگاری به وجود آید.

۶- **مدیریت تهیهٔ نسخه پشتیبان[1] و ترمیم[2] پایگاه داده‌ها:** تصور کنید در اثر خرابی‌دیسکهایی که داده‌ها روی آن ذخیره شده‌اند یا بروز حوادث طبیعی یا قطع برق اطلاعات موجود در پایگاه داده‌ها دچار خرابی شوند. در این صورت وضعیت سازمانی که اطلاعات چندین هفته، چندین ماه و یا چندین سال خود را از دست داده است چه خواهد شد؟ برای جلوگیری از بروز چنین مشکلی در کلیهٔ سازمانها طبق یک برنامهٔ زمانی مشخص (مثلاً یک روز در هفته یا هر روز یا روزی ۲ بار) از کل اطلاعات پایگاه داده‌ها یک کپی روی یک دیسک سخت و یا دیسکت و یا CD و یا در صورت حساسیت زیاد اطلاعات روی یک کامپیوتر که در یک شهر دیگر قرار دارد و از طریق شبکه به کامپیوتر ما متصل شده است، تهیه می‌شود. در صورت بروز هر نوع خرابی، پایگاه داده‌ها با استفاده از اطلاعات موجود در آخرین نسخهٔ پشتیبان بازسازی می‌شود مثلاً اگر آخرین نسخهٔ پشتیبان روز دوشنبه تهیه شده باشد و روز چهارشنبه پایگاه داده‌ها دچار خرابی شود، تنها اطلاعات روزهای سه شنبه و چهارشنبه از دست می‌روند. به عملیات بازسازی اطلاعات پایگاه داده‌ها، ترمیم پایگاه داده‌ها گفته می‌شود. در سیستمهای پایگاه داده‌های قدیمی، اپراتور هر قسمت، مسئول تهیهٔ نسخهٔ پشتیبان بود ولی DBMS-های کنونی این وظیفه را خود برعهده می‌گیرند. DBA می‌تواند برنامهٔ زمانی موردنظر برای تهیهٔ نسخهٔ پشتیبان و محل ذخیره شدنِ آن را برای DBMS مشخص کند. DBMS وظیفه دارد طبق برنامهٔ دیکته شده عمل تهیهٔ نسخهٔ پشتیبان را انجام دهد. مثلاً ممکن است DBA از DBMS بخواهد که ساعت ۰۰: ۱۴ روزهای یکشنبه عمل تهیهٔ نسخهٔ پشتیبان را انجام دهد و نسخهٔ پشتیبان را در مسیر F:\backup ذخیره کند. در این

1 - Backup
2 - Recovery

صورت، DBMS در زمان مشخص شده و به صورت اتوماتیک عمل تهیهٔ نسخهٔ پشتیبان را انجام می‌دهد.

۷- **تأمین جامعیت داده‌ای**: مفهوم جامعیت داده‌ای شامل دو مفهوم اعتبار داده‌ها[1] و سازگاری داده‌ها[2] می‌باشد. اعتبار داده‌ها به این معناست که هیچ مقدار نامعتبری در پایگاه داده‌ها وارد نشود، مثلاً نمرهٔ ۲۲ در سیستمی که بازهٔ قابل قبول برای نمره در آن اعداد بین ۰ تا ۲۰ است، یک دادهٔ نامعتبر محسوب می‌شود. سازگاری داده‌ها به این معناست که در صورت ذخیرهٔ یک داده در چندین محل، مقدار همهٔ آنها یکسان باشد مثلاً شماره تلفن یک استاد در یک محل ۶۲۴۴۴۴ و در محل دیگر ۶۲۴۴۴۳ نباشد و یا چنانچه در جدول دانشجویان دانشجویی با شمارهٔ ۷۸۰۱ وجود نداشته باشد، در جدول ثبت نام نیز ثبت نامی برای دانشجوی ۷۸۰۱ وجود نداشته باشد. DBMS با به حداقل رساندن افزونگی داده‌ها سازگای داده‌ها را به حداکثر می‌رساند و از طرف دیگر، همواره مراقب است تا قوانین اعتبار داده‌ها که توسط DBA مشخص شده‌اند نقض نشوند و از انجام هرگونه عملیات که ممکن است سازگای داده‌ها و یا اعتبار داده‌ها را به خطر اندازد جلوگیری می‌کند.

۸- **تأمین زبان پرس و جو[3] و ابزارهای مدیریت پایگاه داده‌ها**: DBMS یک زبان پرس و جو در اختیار کاربران پایگاه داده‌ها قرار می‌دهد. زبان پرس و جو، زبانی بسیار ساده است که به کاربران اجازه می‌دهد دستوری را روی پایگاه داده‌ها صادر کنند بدون آنکه از چند و چون نحوهٔ انجام آن مطلع باشند. DBMS، این دستورات را به کدهای پیچیدهٔ لازم ترجمه کرده، کدهای بدست آمده را روی پایگاه داده‌ها اجرا می‌کند. علاوه بر این، DBMS ابزارهایی در اختیار طراحان پایگاه داده‌ها و DBA قرار می‌دهد تا به راحتی پایگاه داده‌های موردنظر خود را ایجاد، پیاده‌سازی و پشتیبانی کرده، بر عملکرد آن نظارت کنند.

---

1 - Data Validation
2 - Data Consistency
3 - Query Language

۹- **تأمین امکان دسترسی به پایگاه داده‌ها از طرق مختلف:** DBMS-های کنونی این امکان را به کاربران می‌دهند که از طریق وب[1] داده‌هایی را ارسال کرده، این داده‌ها را در پایگاه داده‌ها ذخیره کنند و یا از طریق وب گزارشی بر اساس اطلاعات موجود در پایگاه داده‌ها درخواست کنند. DBMS این گزارشات را با فرمتی که قابل نمایش در کلیهٔ مرورگرهای وب[2] می‌باشد برای کاربران ارسال می‌کند.

۷-۱)معماری سیستمهای پایگاه داده‌ها

اولین وظیفهٔ DBMS تبدیل درخواستهای سطح بالای کاربران به کدهای پیچیدهٔ لازم برای انجام آن دستورات و اعمال این کدها روی پایگاه داده‌ها می‌باشد به گونه‌ای که جزئیات ذخیره و بازیابی داده‌ها کاملاً از دید کاربر پوشیده باشد. برای تأمین این پوشیدگی ANSI یک معماری سه لایه برای سیستمهای پایگاه داده‌ها ارائه کرده است. لایه‌های این مدل عبارتند از:

۱) لایهٔ فیزیکی یا داخلی[3]

در این لایه، داده‌های فیزیکی همان گونه که روی محیط فیزیکی ذخیره شده‌اند نمایش داده می‌شود.

۲) لایهٔ ادراکی یا انتزاعی[4]

لایهٔ ادراکی شامل دید ادراکی است. دید ادراکی دیدی است که طراح پایگاه داده‌ها نسبت به کلیهٔ موجودیتها و ارتباطات میان آنها دارد. این دید یک دید جامع است که دید کلیهٔ کاربران نهایی از روی آن استخراج می‌شود. در ابتدای کار، طراح پایگاه داده‌ها با کلیهٔ کاربران سیستم مصاحبه کرده و با توجه به نیازهای متفاوت و گاه متضاد آنها کلیهٔ موجودیتهای سیستم، ارتباطات میان آنها و خصوصیات هر موجودیت را به گونه‌ای که نیازهای کلیهٔ کاربران برآورده شود تعریف می‌کند.

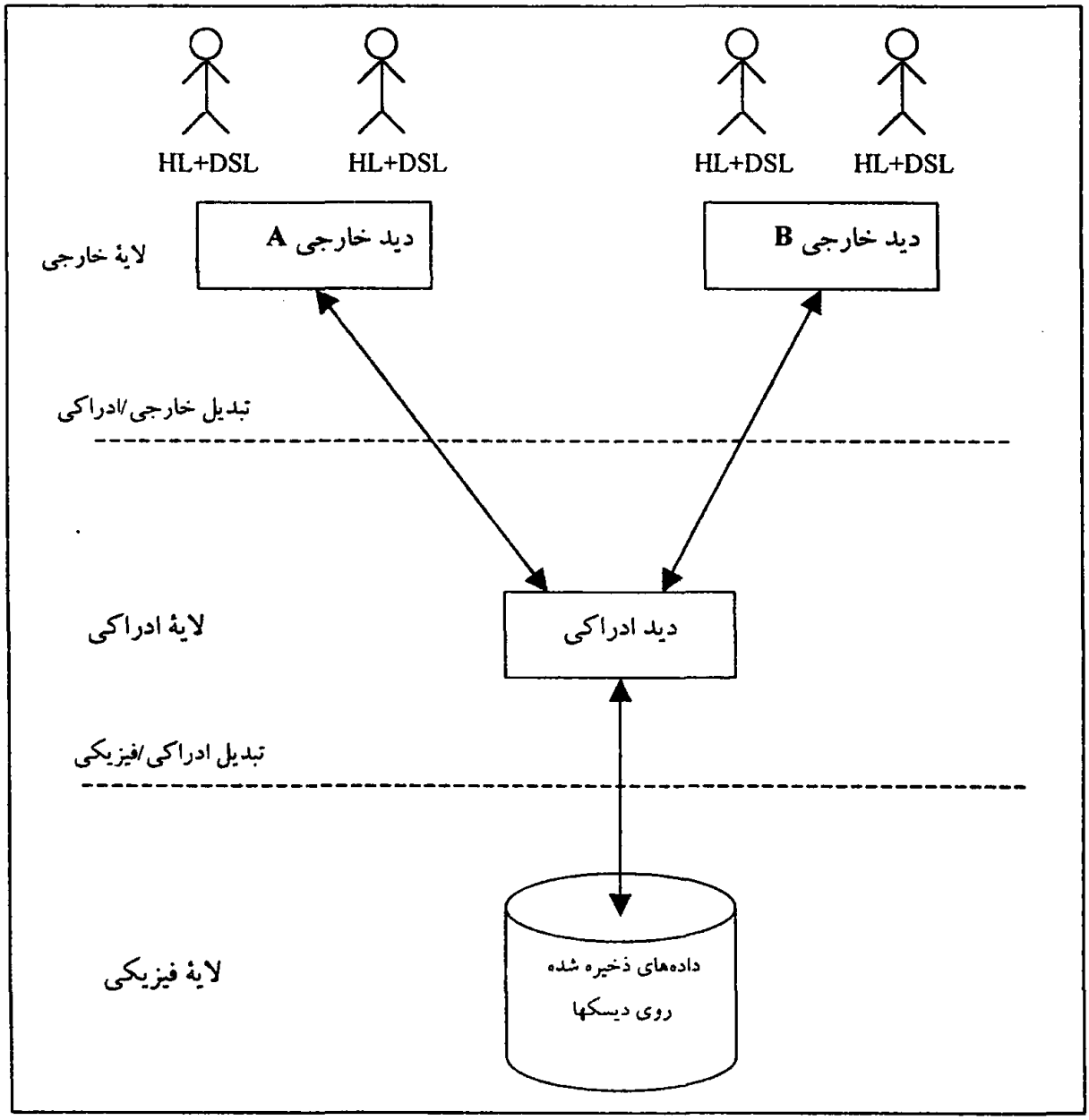
---

1 - Web
2 - Web Browsers
3 - Physical or Internal Layer
4 - Conceptual Layer

۳) لایهٔ خارجی[1]

لایهٔ خارجی شامل دید خارجی کلیهٔ کاربران است. دید خارجی دیدی است که هر کاربر نسبت به اطلاعات ذخیره شده دارد. این دید لزوماً با دید ادراکی یکسان نیست و هر کاربر می‌تواند از نقطه نظر متفاوتی به داده‌ها نگاه کند. ممکن است دید چندین کاربر یکسان باشد. همچنین ممکن است هر کاربر از چند دید خارجی متفاوت استفاده کند.

شکل زیر این معماری را نشان می‌دهد:



1 - External Layer

انجام تبدیلات بین سه لایه وظیفهٔ DBMS است. DBMS باید طبق دستورالعمل DBA دیدهای خارجی را براساس دید ادراکی و دید ادراکی را براساس دید فیزیکی و بالعکس بدست آورد.

برای درک تفاوت میان دید ادراکی و دید خارجی، محیط یک دانشگاه را در نظر بگیرید. دید طراح باید شامل نیازمندیهای کلیهٔ بخشهای دانشگاه شامل آموزش، امور مالی، امور دانشجویی و ... باشد. مسلماً این دید، اطلاعات بسیار وسیعی را در بر می‌گیرد. فرض کنید این دید شامل جداول:

دانشجو(شماره دانشجویی، نام، نام خانوادگی، رشته، سال ورود، شماره شناسنامه، محل صدور، تاریخ تولد، نوع بیمه، شغل و...)

درس(شمارهٔ درس، نام درس، تعداد واحد، نوع درس، قیمت هر واحد و...)

استاد(شمارهٔ استاد، نام، نام خانوادگی، مدرک و...)

نمرات(شماره دانشجویی، شمارهٔ درس ، ترم، نمره)

... و بسیاری جداول دیگر باشد. بسیاری از این اطلاعات ممکن است خارج از محدودهٔ کاری یک بخش باشند مثلاً برای کارمند آموزش، شماره شناسنامه و یا نوع بیمهٔ یک دانشجو اهمیتی ندارد پس دلیلی وجود ندارد که این فیلدها را ببیند. از طرف دیگر، ممکن است در بسیاری از گزارشات قسمت آموزش معدل کل دانشجو مورد استفاده باشد. در هیچ یک از جداولی که طراح در نظر گرفته است، ویژگی معدل وجود ندارد ولی می‌توان مقدار این ویژگی را بر اساس نمرات دانشجو و تعداد واحد هر درس محاسبه کرد. پس DBA می‌تواند یک دید خارجی مخصوص کارمندان آموزش تعریف کند که به صورت ذیل باشد:

دانشجو از دید آموزش(شماره دانشجویی، نام، نام خانوادگی، رشته، سال ورود، معدل کل)

هنگام تعریف دید خارجی، DBA برای DBMS مشخص می‌کند که مقدار هر یک از ویژگیهای دید خارجی از کدام جداول دید ادراکی و چگونه بدست می‌آیند. پس از آن، وظیفهٔ DBMS است که از روی جداول دید ادراکی دیدهای خارجی را ایجاد کرده و مرتباً به روز رسانی کند

مثلاً اگر در جدول نمرات نمرهٔ جدیدی برای دانشجوی ۷۸۰۱ وارد شود، DBMS وظیفه دارد به صورت اتوماتیک مقدار ویژگی معدل این دانشجو را در دید خارجی اصلاح کند.

در هر برنامهٔ کاربردی که با پایگاه داده‌ها سر و کار دارد از دو دسته زبان استفاده می‌شود:

۱- زبان فرعی داده‌ای یا DSL [1]

۲- زبان میزبان یاHL[2]

۱- زبان فرعی داده‌ای: از دستورات زبان فرعی داده‌ای برای کار با داده‌ها استفاده می‌شود. این دستورات به سه دسته تقسیم می‌شوند:

۱-۱- زبان تعریف داده‌ها یا DDL[3]

از این دستورات برای تعریف ساختار جداول و ایندکسها و ... استفاده می‌شود. به عنوان مثال، از دستور create table برای ایجاد یک جدول و معرفی فیلدهای آن استفاده می‌شود:

```
create table course(c# int,cname char(20), unit int, primary key(c#))
```

این دستور جدول ذیل را ایجاد می‌کند:

course

c#	cname	unit

۱-۲- زبان دستکاری داده‌ها یا DML [4]

از این دستورات برای بازیابی، درج، حذف و اصلاح اطلاعات جداول استفاده می‌شود. به عنوان مثال، از دستور insert برای درج یک سطر جدید در یک جدول استفاده می‌شود:

```
insert into course(c#,cname,unit) values(1500,'ریاضی',3)
```

---

1 - Data SubLanguage

2 - Host Language

3 - Data Definition Language

4 - Data Manipulation Language

c#	cname	unit
1500	ریاضی	3

۳-۱- زبان کنترل داده‌ها یا DCL[1]

از این دستورات برای مدیریت مجوزهای دسترسی به پایگاه داده‌ها استفاده می‌شود. به عنوان مثال، با استفاده از دستور ذیل می‌توان مجوز تغییر ویژگی cname از جدول course را به کاربری که شناسهٔ وی ali است واگذار کرد:

grant update(cname) on course to ali

معروفترین زبان فرعی داده‌ای زبان SQL[2] است. این زبان توسط کلیهٔ DBMS-های کنونی مانند Informix، SQL Sever و Oracle پشتیبانی می‌شود.

۲- زبان میزبان

زبانهای فرعی داده‌ای معمولاً فاقد امکانات لازم برای تعریف متغیرها، کنترل ظاهر برنامه، ایجاد حلقه‌ها، تست کردن شرطها و ... هستند. به همین دلیل تنها با استفاده از زبان فرعی داده‌ای نمی‌توان یک برنامهٔ کاربردی کامل ایجاد کرد. در هر برنامهٔ کاربردی برای کار با داده‌ها از دستورات زبان فرعی داده‌ای و برای انجام عملیاتی که با داده‌ها سر و کار ندارند(مثل طراحی منوها، ایجاد حلقه‌ها، تست کردن شرطها و ...) از دستورات یک زبان برنامه‌نویسی مانند دلفی، ویژوال C و ... استفاده می‌شود. در واقع، دستورات زبان فرعی داده‌ای در دستورات زبان برنامه‌نویسی (زبان میزبان) مهمان می‌شوند.

---

1 - Data Control Language

2 - Structured Query Language

۸-۱) مدلهای پایگاه داده‌ها

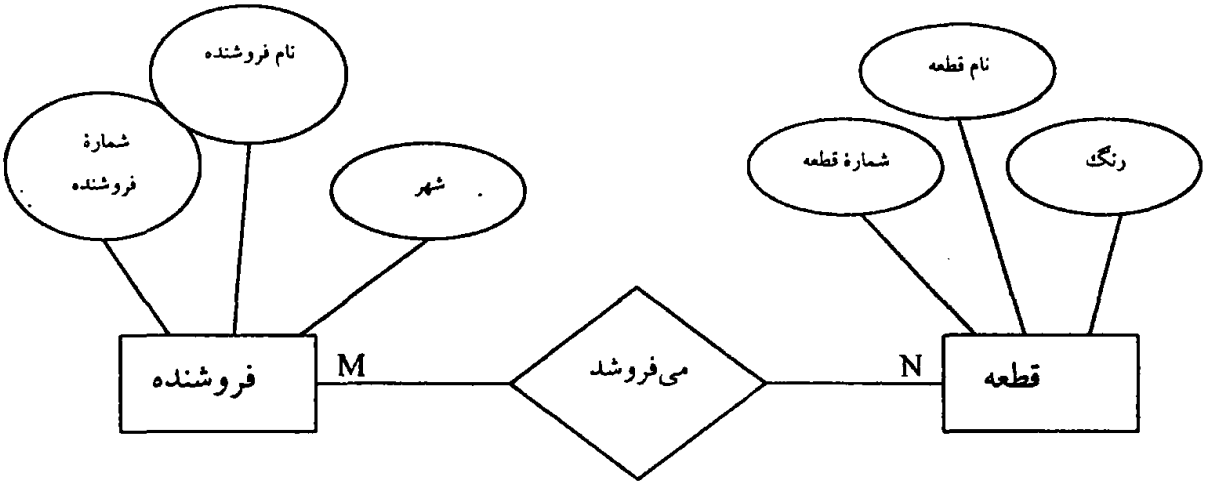
مدل پایگاه داده‌ها مجموعه‌ای از ساختارهای منطقی است که ساختار داده‌ها و روابط میان داده‌ها را نمایش می‌دهد. مدلهای پایگاه داده‌ها را می‌توان به دو گروه اساسی تقسیم کرد: مدلهای انتزاعی[1] و مدلهای پیاده‌سازی[2].

- ❖ مدلهای انتزاعی ساختار منطقی داده‌ها و روابط منطقی میان آنها را نشان می‌دهند. این مدلها تنها مشخص می‌کنند در پایگاه داده‌ها چه چیزی باید ارائه شود و به چگونگی پیاده‌سازی نمی‌پردازند. مدلهای انتزاعی شامل مدل موجودیت-رابطه یا ERM [3] و مدل شی‌گرا یا OOM[4] می‌باشد.
- ❖ مدلهای پیاده‌سازی مشخص می‌کنند آنچه در مدلهای انتزاعی در نظر گرفته شده است چگونه بایستی پیاده‌سازی شود. این مدلها شامل مدل سلسله‌مراتبی[5]، مدل شبکه‌ای[6] و مدل رابطه‌ای[7] می‌باشند.

مدل شی‌گرا از بحث این کتاب خارج است و دربارهٔ مدل ER در فصلهای بعدی به طور مفصل بحث خواهیم کرد. در این مرحله تنها به مدلهای پیاده‌سازی می‌پردازیم. برای درک مدلهای پیاده‌سازی، یک سیستم فروش قطعه توسط فروشندگان مختلف را در نظر بگیرید. موجودیتهای اساسی این سیستم عبارتند از: فروشندگان و قطعات. رابطه‌ای که بین این دو موجودیت وجود دارد، فروش است. طبق قوانین این سیستم، هر فروشنده ممکن است چندین نوع قطعه را بفروشد و هر نوع قطعه ممکن است توسط چندین فروشندهٔ مختلف فروخته شود. بنابراین این رابطه یک رابطه چند به چند یا M:N است. مدل ER این سیستم به شکل ذیل خواهد بود:

---

1 - Conceptual Models
2 - Implementation Models
3 - Entity-Relationship Model
4 - Object-Oriented Model
5 - Hierarchical Database Model
6 - Network Database Model
7 - Relational Database Model



فرض کنید اطلاعاتی که قرار است در پایگاه داده ها ذخیره شوند به قرار ذیل باشند:

- اطلاعات فروشندگان:

شمارهٔ فروشنده = S1	نام فروشنده = تهران قطعه	شهر = تهران
شمارهٔ فروشنده = S2	نام فروشنده = خودیاران	شهر = تهران
شمارهٔ فروشنده = S3	نام فروشنده = یزد قطعه	شهر = یزد
شمارهٔ فروشنده = S4	نام فروشنده = البرز	شهر= اصفهان

- اطلاعات قطعات:

شمارهٔ قطعه = P1	نام قطعه = تیرآهن	رنگ = مشکی
شمارهٔ قطعه = P2	نام قطعه = آرماتور	رنگ = مشکی
شمارهٔ قطعه = P3	نام قطعه = سیمان	رنگ = طوسی
شمارهٔ قطعه = P4	نام قطعه = آلومینیم	رنگ = سفید

- اطلاعات مربوط به فروش:

فروشندهٔ S1 ، ۱۰۰۰۰ کیلوگرم از قطعهٔ P2 فروخته است.

فروشندهٔ S1 ، ۳۰۰۰ کیلوگرم از قطعهٔ P4 فروخته است.

فروشندهٔ S2 ، ۲۰۰۰ کیلوگرم از قطعهٔ P1 فروخته است.

فروشندهٔ S2 ، ۴۰۰۰ کیلوگرم از قطعهٔ P2 فروخته است.

فروشندهٔ S2 ، ۳۰۰۰ کیلوگرم از قطعهٔ P4 فروخته است.

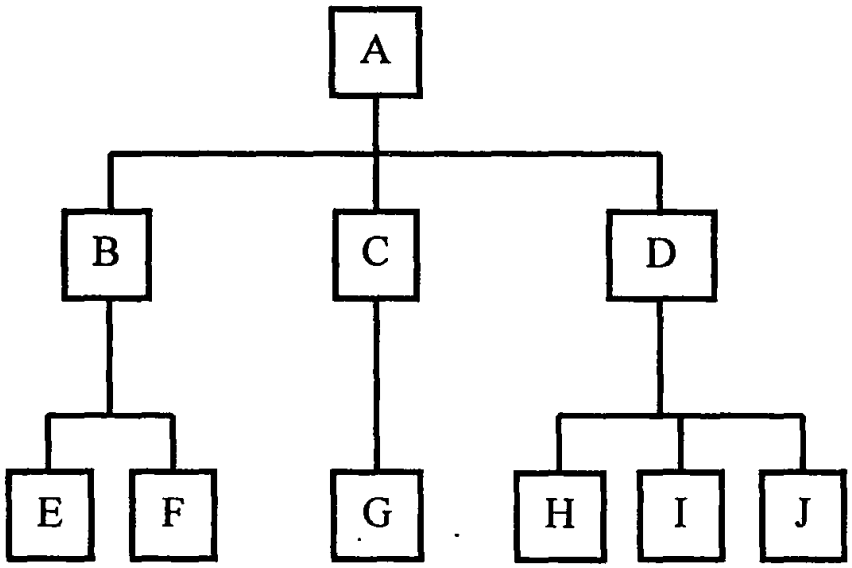
فروشندهٔ S3 ، ۳۰۰۰ کیلوگرم از قطعهٔ P4 فروخته است.

حال مدل ER فروشنده-قطعه را به ترتیب با مدلهای سلسله‌مراتبی، شبکه‌ای و رابطه‌ای پیاده‌سازی کرده، این مدلها را با هم مقایسه می‌کنیم.

۱-۸-۱)مدل سلسله‌مراتبی

۱-۱-۸-۱) اساس مدل سلسله‌مراتبی

مدل سلسله‌مراتبی قدیمی‌ترین مدل پایگاه داده‌هاست. در این مدل، داده‌ها در گره‌های یک درختواره ذخیره می‌شوند. درختواره، یک گراف غیرچرخشی و متصل است که یک ریشه دارد. در درختواره هر گره می‌تواند چند فرزند و تنها یک پدر داشته باشد. به همین دلیل مدل سلسله‌مراتبی تنها برای پیاده‌سازی روابط یک به چند مناسب است و برای پیاده‌سازی روابط چند به چند مناسب نیست. گراف ذیل را در نظر بگیرید:

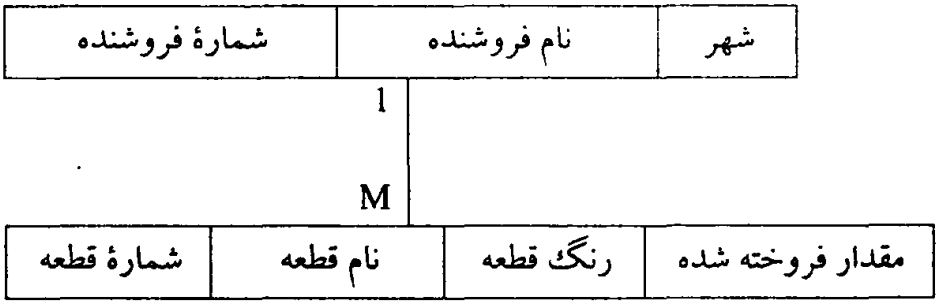


همانطور که مشاهده می‌کنید در این گراف هیچ گرهٔ منفصلی وجود ندارد و کلیهٔ گره‌ها غیر از گرهٔ ریشه تنها یک پدر دارند پس این گراف درختواره است.

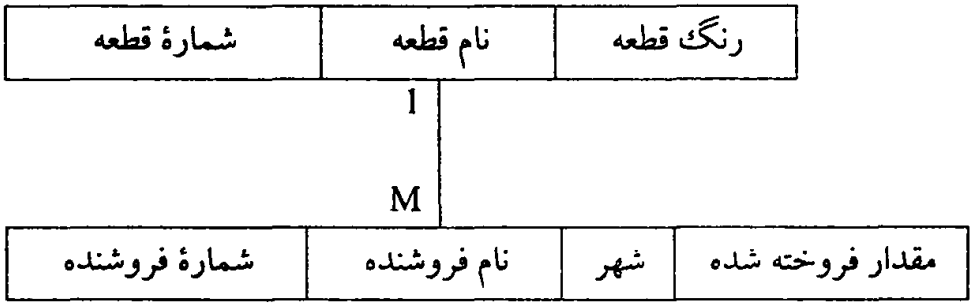
مسیر پیمایش درختواره از بالا به پائین و از چپ به راست است. مثلاً اگر بخواهیم به گرهٔ J برسیم باید مسیر ABEFCGDHIJ را طی کنیم.

از آنجا که مدل سلسله‌مراتبی برای پیاده‌سازی روابط چند به چند مناسب نیست، برای پیاده‌سازی رابطهٔ چند به چند میان فروشندگان و قطعات دو روش پیش رو داریم:

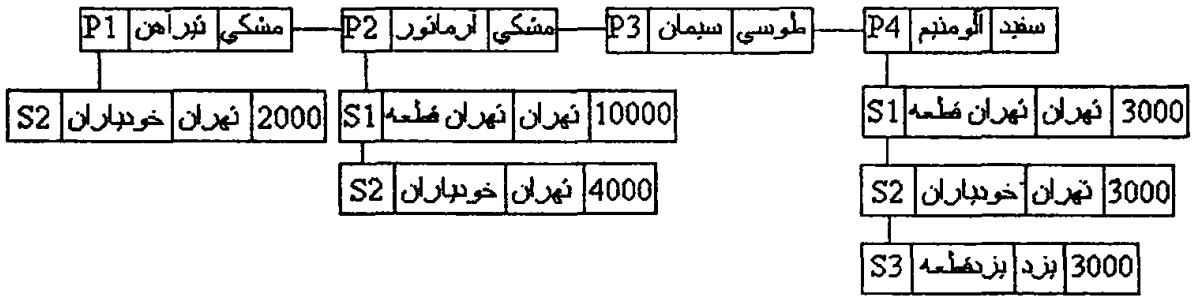
۱- طرف فروشنده را یک و طرف قطعه را چند در نظر بگیریم.



۲- طرف قطعه را یک و طرف فروشنده را چند قرار دهیم.



فرض کنید بخواهیم از روش دوم استفاده کنیم. در این صورت، مدل پایگاه داده‌های کوچک ما به شکل ذیل خواهد بود:



همانطور که مشاهده می‌کنید، اطلاعات فروشندگان هر قطعه در گره‌هایی که زیر گرهٔ قطعهٔ مورد نظر قرار دارند، درج می‌شوند. مقدار فروش نیز در گرهٔ فروشنده ثبت می‌شود. کلیهٔ گره‌های

درختواره از طریق اشاره‌گر به هم متصل هستند. هر گرهٔ قطعه شامل دو اشاره‌گر است: اشاره‌گری که به قطعهٔ بعدی اشاره می‌کند و اشاره‌گری که به اولین فروشندهٔ آن قطعه اشاره می‌کند. هر گرهٔ فروشنده نیز شامل یک اشاره‌گر به گرهٔ فروشندهٔ بعدی می‌باشد.

۲-۱-۸-۱) بررسی عملیات گوناگون در مدل سلسله‌مراتبی

در این قسمت عملیات بازیابی، درج، حذف و اصلاح را در مدل سلسله‌مراتبی بررسی می‌کنیم:

عملیات بازیابی

برای درک بهتر نحوهٔ بازیابی در مدل سلسله‌مراتبی، دو پرس و جوی قرینهٔ ذیل را در نظر بگیرید:

Q1: نام فروشندگانی را بیابید که قطعهٔ P2 را فروخته‌اند.

روش بدست آوردن نتیجهٔ این پرس و جو بسیار ساده است. کافیست در قطعات، قطعهٔ شمارهٔ P2 را یافته و نام کلیهٔ فروشندگان زیر این قطعه را واکشی کنیم.

Q2: نام قطعاتی را بیابید که توسط فروشندهٔ S2 فروخته شده‌اند.

برای بدست آوردن نتیجهٔ این پرس و جو باید کلیهٔ قطعات را پیمایش کرده، برای هر قطعه جستجوی عمقی انجام دهیم، اگر S2 جزء فروشندگانی باشد که زیر قطعهٔ مورد نظر قرار دارند، مشخص می‌شود که قطعهٔ مورد نظر توسط فروشندهٔ S2 فروخته شده است و باید نام آن را در لیست جوابها قرار داد.

مشاهده می‌کنید با اینکه پرس و جوها کاملاً قرینهٔ یکدیگرند، عملیات صورت گرفته برای بدست آوردن نتایج آنها کاملاً با هم متفاوت است و عملیات مورد نیاز برای پرس و جوی دوم بسیار وقت‌گیرتر و پیچیده‌تر از عملیات مورد نیاز برای پرس و جوی اول می‌باشد.

عملیات درج

اگر دقت کرده باشید، اطلاعات فروشندهٔ S4 در این مدل در نظر گرفته نشده است. دلیل این امر نیز روشن است. S4 هنوز قطعه‌ای نفروخته است بنابراین نمی‌توان اطلاعات وی را در گره‌ای زیر یک گرهٔ قطعه درج کرد به عبارت دیگر، گره‌های بدون پدر را نمی‌توان در این مدل درج کرد. بنابراین، مدل سلسله‌مراتبی در عملیات درج ناهنجاری دارد.

عملیات حذف

فرض کنید بخواهیم اطلاعات قطعهٔ P4 را از پایگاه داده‌ها حذف کنیم. با حذف این قطعه کلیهٔ گره‌های زیر این قطعه یعنی گره‌های مربوط به S1 و S2 و S3 نیز به طور خودکار حذف می‌شوند. از آنجا که اطلاعات مربوط به فروشندگان S1 و S2 در گره‌های دیگر نیز وجود دارند، حذف این دو گره مشکلی به وجود نخواهد آورد ولی حذف گرهٔ S3 باعث بروز مشکلاتی خواهد شد چرا که اطلاعات این قطعه تنها در همین گره ثبت شده است. به عبارت ساده‌تر، با حذف اطلاعات مربوط به قطعهٔ P4، به طور ناخواسته اطلاعات مربوط به فروشندهٔ S3 را نیز از دست می‌دهیم. از طرف دیگر، اگر بخواهیم اطلاعات مربوط به فروشندهٔ S1 را از پایگاه داده‌ها حذف کنیم، مجبوریم گره‌های متعددی را حذف کنیم و حذف یک گره اکتفا نمی‌کند چون اطلاعات هر فروشنده در چندین گره ثبت شده است.

عملیات اصلاح

فرض کنید بخواهیم رنگ قطعهٔ P3 را از طوسی به مشکی تغییر دهیم. در این صورت کافیست این تغییر را تنها در یک گره اعمال کنیم چون اطلاعات هر قطعه تنها در یک گره ثبت شده است ولی اگر بخواهیم شهر فروشندهٔ S1 را از تهران به ساری تغییر دهیم، اعمال این تغییر تنها در یک گره اکتفا نخواهد کرد چون ممکن است اطلاعات این فروشنده در گره‌های متعددی ثبت شده باشد و حفظ سازگاری پایگاه داده‌ها ایجاب می‌کند این تغییر در کلیهٔ این گره‌ها اعمال شود. به عبارت دیگر، برای حفظ سازگاری داده‌ها لازم است اصلاح منتشر شونده صورت گیرد.

۳-۱-۸-۱)معایب مدل سلسله‌مراتبی

با توجه به موارد مذکور در بخش قبل، مهمترین معایب مدل سلسله‌مراتبی به شرح ذیل است:

۱- مدل سلسله‌مراتبی برای طراحی و پیاده‌سازی روابط چند به چند مناسب نیست.

۲- تهیهٔ برخی گزارشات در این مدل بسیار وقتگیر است.

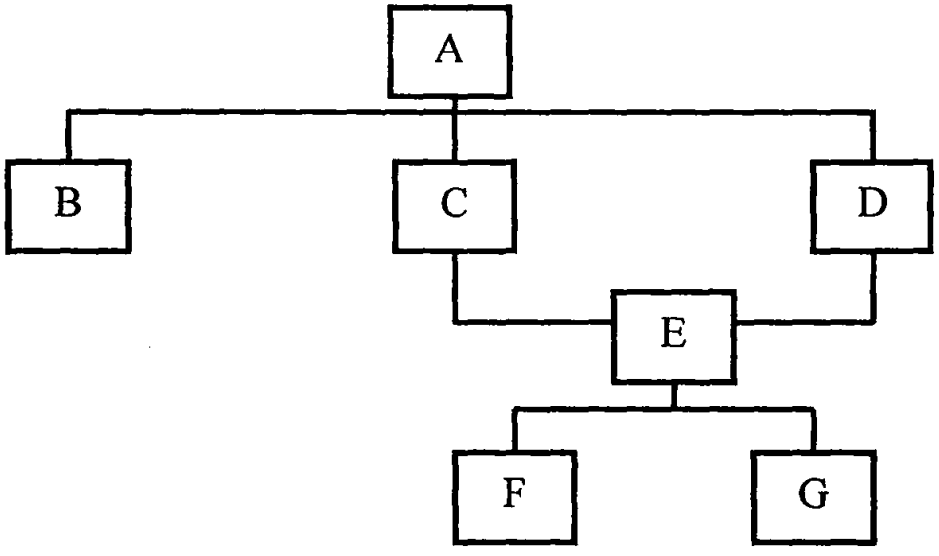
۳- این مدل در عملیات درج، حذف و اصلاح ناهنجاری دارد.

٤- به دلیل استفاده از اشاره‌گرها و وابستگی به آدرسهای فیزیکی ذخیره‌سازی، مدل کاملاً به محیط فیزیکی ذخیره‌سازی وابسته است و با اعمال هرگونه تغییر در ساختار داده‌ها، طراح پایگاه داده‌ها و برنامه‌نویسان به زحمت می‌افتند!!!

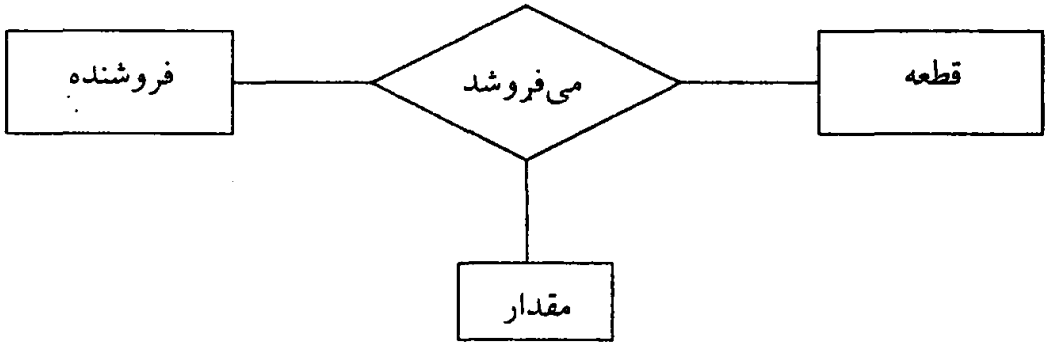
۲-۸-۱) مدل شبکه‌ای

۱-۲-۸-۱) اساس مدل شبکه‌ای

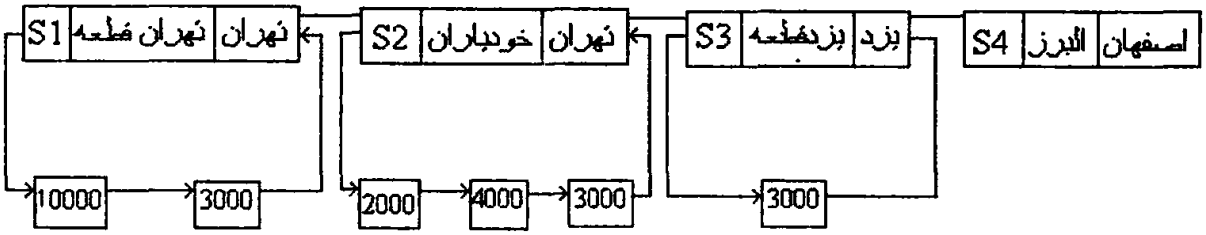
در مدل شبکه‌ای، اطلاعات در گره‌های یک گراف دلخواه ذخیره می‌شوند. در یک گراف دلخواه، هر گره می‌تواند چند پدر و چند فرزند داشته باشد. بنابراین، مدل شبکه‌ای برای پیاده‌سازی روابط چند به چند نیز مناسب است. شکل ذیل یک گراف دلخواه را نشان می‌دهد:



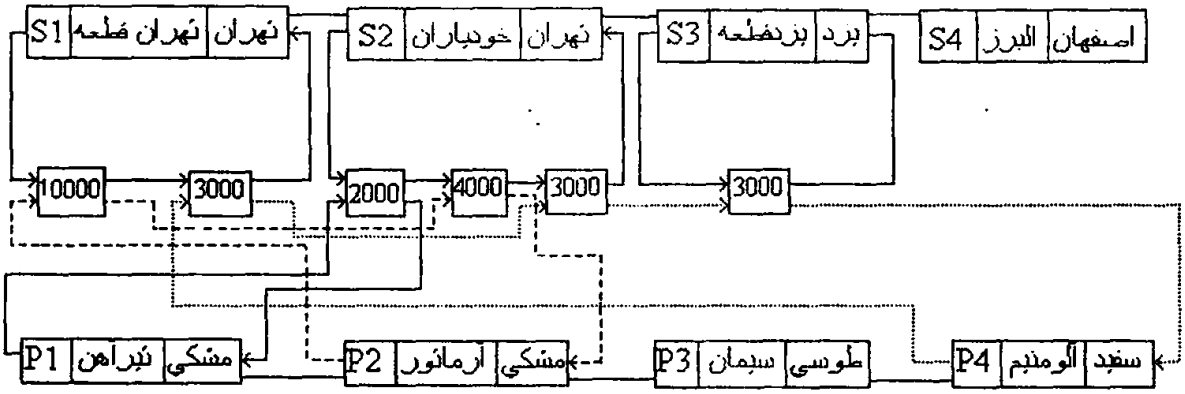
حال مدل شبکه‌ای را برای پایگاه دادهٔ کوچک خودمان رسم می‌کنیم. در این مدل از مقدار فروش برای اتصال گره‌های فروشندگان و قطعات استفاده می‌کنیم:



قدم ۱: برای هر فروشنده، مقدار فروش کلیهٔ قطعات را در نظر گرفته، کلیهٔ این گره‌ها را از طریق اشاره‌گر به هم متصل می‌کنیم. مثلاً فروشندهٔ S1، ۱۰۰۰۰ کیلوگرم از قطعه P2 و ۳۰۰۰ کیلوگرم از قطعهٔ P4 فروخته است. بنابراین برای S1 دو مقدار ۱۰۰۰۰ و ۳۰۰۰ را در نظر گرفته، از S1 یک اشاره‌گر به ۱۰۰۰۰ و از ۱۰۰۰۰ یک اشاره‌گر به ۳۰۰۰ و از ۳۰۰۰ یک اشاره‌گر به خود S1 رسم می‌کنیم و این کار را برای کلیهٔ فروشندگان تکرار می‌کنیم:



قدم ۲: برای هر قطعه، مقدار فروخته شده توسط یک فروشنده را در نظر گرفته، کلیهٔ این گره‌ها را توسط اشاره‌گر بهم متصل می‌کنیم. مثلاً قطعهٔ P2 توسط فروشندهٔ S1 به مقدار ۱۰۰۰۰ کیلوگرم و توسط فروشندهٔ S2 به مقدار ۴۰۰۰ کیلوگرم فروخته شده است، بنابراین از قطعهٔ P2 یک اشاره‌گر به ۱۰۰۰۰-یی که در طرف دیگر به فروشندهٔ S1 وصل است و از ۱۰۰۰۰ یک اشاره‌گر به ۴۰۰۰-یی که در طرف دیگر به فروشندهٔ S2 وصل است و از ۴۰۰۰ یک اشاره‌گر به خود P2 رسم می‌کنیم. این کار را برای کلیهٔ قطعات تکرار می‌کنیم:



۲-۲-۸-۱) بررسی عملیات گوناگون در مدل شبکه‌ای

عملیات بازیابی :

دو پرس و جوی قرینه‌ای را که در مدل سلسله‌مراتبی بررسی کردیم در نظر بگیرید:

Q1 : نام فروشندگانی را بیابید که قطعهٔ P2 را فروخته‌اند.

برای بدست آوردن نتیجهٔ این پرس و جو باید در میان قطعات، قطعهٔ P2 را یافته، اتصالاتی (گره‌های مقادیر) را که به این قطعه متصلند پیمایش کرده و نام فروشندگانی را که در سمت دیگر این اتصالات قرار دارند بدست آوریم.

Q2 : نام قطعاتی را بیابید که توسط S2 فروخته شده‌اند.

برای بدست آوردن نتیجهٔ این پرس و جو باید در میان فروشندگان فروشندهٔ S2 را یافته، اتصالاتی را که به این فروشنده متصلند پیمایش کرده و نام قطعاتی را که در سمت دیگر این اتصالات قرار دارند بدست آوریم.

مشاهده می‌کنید که برای پرس و جوهای قرینه عملیات کاملاً مشابهی باید صورت گیرد.

عملیات درج

همان گونه که مشاهده می‌کنید در مدل شبکه‌ای هیچ اشکالی برای درج اطلاعات قطعهٔ P3 که توسط هیچ یک از فروشندگان فروخته نشده است و یا S4 که هیچ قطعه‌ای را نفروخته است وجود ندارد بنابراین مدل شبکه‌ای در عملیات درج ناهنجاری ندارد.

عملیات حذف

در این مدل می‌توانیم اطلاعات هر قطعه‌ای را حذف کنیم بدون آنکه اطلاعات فروشنده‌ای را از دست بدهیم و بالعکس. بنابراین مدل شبکه‌ای در عملیات حذف ناهنجاری ندارد.

عملیات اصلاح

فرض کنید بخواهیم رنگ قطعهٔ P3 را از طوسی به مشکی تغییر دهیم، برای اعمال این تغییر کافیست تنها اطلاعات یک گره را تغییر دهیم چون اطلاعات هر قطعه تنها در یک گره ثبت شده است. به همین ترتیب، اگر بخواهیم شهر فروشندهٔ S1 را از تهران به ساری تغییر دهیم، کافیست این تغییر را در یک گره اعمال کنیم چون اطلاعات هر فروشنده نیز تنها در یک گره ثبت شده است. بنابراین در مدل شبکه‌ای، در عملیات اصلاح ناهنجاری وجود ندارد و نیازی به اصلاح منتشر شونده نیست.

۳-۲-۸-۱) محاسن مدل شبکه‌ای نسبت به مدل سلسله‌مراتبی

۱-مدل شبکه‌ای برخلاف مدل سلسله‌مراتبی که تنها برای پیاده‌سازی روابط یک به چند مناسب می‌باشد برای پیاده‌سازی روابط چند به چند نیز مناسب است.

۲-در مدل شبکه‌ای، ناهنجاری درج، حذف و اصلاح وجود ندارد.

٤-۲-۸-۱) معایب مدل شبکه‌ای

۱- طراحی و پیاده‌سازی و برنامه‌نویسی در مدل شبکه‌ای بسیار پیچیده است.

۲- مدل شبکه‌ای به دلیل استفاده از اشاره‌گرها کاملاً به محیط فیزیکی ذخیره‌سازی داده‌ها وابسته است و با انجام کوچکترین تغییر در ساختار پایگاه داده‌ها، طراحان و برنامه‌نویسان با مشکل مواجه می‌شوند.

۳-۸-۱) مدل رابطه‌ای

۱-۳-۸-۱) اساس مدل رابطه‌ای

در مدل رابطه‌ای، داده‌ها در داخل جداول ذخیره می‌شوند. در این مدل برای هر یک از موجودیتها و یا روابط میان موجودیتها یک جدول مجزا در نظر گرفته می‌شود. شکل ذیل مدل رابطه‌ای مثال کوچک ما را نشان می‌دهد:

Supplier

S#	Sname	City
S1	تهران قطعه	تهران
S2	خودیاران	تهران
S3	یزد قطعه	یزد
S4	البرز	اصفهان

Part

P#	Pname	Color
P1	تیرآهن	مشکی
P2	آرماتور	مشکی
P3	سیمان	طوسی
P4	آلومینیم	سفید

SP

S #	P #	qty
S1	P2	10000
S1	P4	3000
S2	P1	2000
S2	P2	4000
S2	P4	3000
S3	P4	3000

همانطور که مشاهده می‌کنید، برای هر یک از موجودیتهای فروشنده و قطعه و رابطهٔ میان فروشنده- قطعه (فروش) یک جدول مجزا در نظر گرفته شده است.

۲-۳-۸-۱) بررسی عملیات گوناگون در مدل رابطه‌ای

عملیات بازیابی :

دو پرس و جوی قرینهٔ قسمت قبل را در نظر بگیرید:

Q1: نام فروشندگانی را بیابید که قطعهٔ P2 را فروخته‌اند.

در جدول SP به راحتی می‌توان شمارهٔ فروشندگانی را که قطعهٔ P2 را فروخته‌اند بدست آورد. برای بدست آوردن نام این فروشندگان کافیست به جدول Supplier مراجعه کنیم.

Q2: نام قطعاتی را بیابید که توسط فروشندهٔ S2 فروخته شده‌اند.

در جدول SP به راحتی می‌توان شمارهٔ قطعاتی را که توسط S2 فروخته شده‌اند بدست آورد. برای بدست آوردن نام این قطعات کافیست به جدول Part مراجعه کنیم.

مشاهده می‌کنید که برای بدست آوردن نتایج این دو پرس و جوی قرینه، عملیات مشابه و بسیار ساده‌ای باید انجام شود.

عملیات درج

در جدول Supplier به راحتی می‌توان اطلاعات فروشنده‌ای مانند S4 را که هنوز هیچ قطعه‌ای نفروخته است و در جدول Part اطلاعات P3 را که هنوز توسط فروشنده‌ای فروخته نشده است، وارد کرد. بنابراین، مدل رابطه‌ای در عملیات درج ناهنجاری ندارد.

عملیات حذف

می‌توان اطلاعات فروشنده‌ای مانند S1 را در جدول Supplier حذف کرد. در این صورت باید کلیهٔ فروشهای مربوط به S1 در جدول SP نیز حذف شوند ولی اطلاعات هیچ قطعه‌ای از دست نمی‌رود. به همین ترتیب می‌توان اطلاعات قطعه‌ای مانند P4 را در جدول Part حذف کرد. در این صورت، برای حفظ سازگاری داده‌ها، بایستی کلیهٔ فروشهای مربوط به P4 در جدول SP نیز حذف شود ولی اطلاعات هیچ فروشنده‌ای از دست نمی‌رود. پس مدل رابطه‌ای در عملیات حذف ناهنجاری ندارد.

عملیات اصلاح

برای آنکه رنگ قطعهٔ P3 را از طوسی به مشکی تغییر دهیم، کافیست این تغییر را تنها در یکی از سطرهای جدول Part اعمال کنیم. به همین ترتیب، برای تغییر شهر فروشندهٔ S1 از تهران به ساری کافیست یکی از سطرهای جدول Supplier را تغییر دهیم و نیازی به اصلاح منتشر شونده نیست. پس مدل رابطه‌ای در عملیات اصلاح ناهنجاری ندارد.

۳-۳-۸-۱) محاسن مدل رابطه‌ای نسبت به مدل شبکه‌ای

در مدل رابطه‌ای نیز مانند مدل شبکه‌ای ناهنجاری درج، حذف و اصلاح وجود ندارد (مشروط بر آنکه جداول به درستی طراحی شده باشند). مدل رابطه‌ای برای پیاده‌سازی روابط یک به چند و چند به چند کاملاً مناسب است. علاوه بر این، مدل رابطه‌ای نسبت به مدل شبکه‌ای دو حسن عمده دارد:

۱- طراحی و پیاده‌سازی آن بسیار ساده است.

۲- به علت عدم وابستگی به آدرسهای فیزیکی، طراحان پایگاه داده‌ها و برنامه‌نویسان را از درگیر شدن با جزئیات ذخیره‌سازی فیزیکی داده‌ها معاف می‌کند.

تمرینهای فصل

۱- یک سیستم پایگاه داده‌ها شامل اطلاعات دانشجویان(شماره دانشجویی، نام دانشجو، رشتهٔ تحصیلی دانشجو)، اطلاعات دروس(کد درس، نام درس، تعداد واحد درس) و نمرات دانشجویان در دروس مختلف می‌باشد. با فرض داشتن اطلاعات ذیل، مدل سلسله‌مراتبی، مدل شبکه‌ای و مدل رابطه‌ای را برای این پایگاه داده‌ها رسم کنید.

اطلاعات دانشجویان:

- (۸۰۱۲۱۲ وعلی و کامپیوتر)
- (۸۰۱۰۱۰ و زهرا و حسابداری)
- (۷۷۰۰۰۰ و حسن و کامپیوتر)

اطلاعات دروس:

-(۴۴۰۰ و پایگاه داده‌ها و ۳)
-(۴۴۰۲ و برنامه سازی و ۳)
-(۴۴۰۳ و سیستم عامل و ۲)
-(۴۴۰۴ و شبکه و ۳)
-(۱۲۰۰ و ادبیات و۲)

اطلاعات مربوط به نمرات:

- علی در درس پایگاه داده‌ها نمرهٔ ۱۸ گرفته است.
- زهرا در درس پایگاه داده‌ها نمرهٔ ۱۲ گرفته است.
- حسن در درس پایگاه داده‌ها نمرهٔ ۱۸ گرفته است.
- حسن در درس سیستم عامل نمرهٔ ۱۰ گرفته است.
- زهرا در درس برنامه‌سازی نمرهٔ ۸ گرفته است.
- زهرا در درس سیستم عامل نمرهٔ ۱۰ گرفته است.

۲

پایگاه داده‌های رابطه‌ای[1]

در فصل قبل مدلهای پایگاه داده‌های سلسله‌مراتبی، شبکه‌ای و رابطه‌ای را به طور مختصر مورد بحث قرار دادیم. در سالهای اخیر محبوبیت مدلهای سلسله‌مراتبی و شبکه‌ای به علت پیچیدگیها و معایبی که این مدلها دارند کم شده است و در حال حاضر پایگاه داده‌های رابطه‌ای رایجترین پایگاه داده‌های مورد استفاده هستند. از آنجا که بحث اصلی این کتاب پایگاه داده‌های رابطه‌ای می‌باشد، در این فصل مفاهیم موجود در پایگاه داده‌های رابطه‌ای و قوانین حاکم بر آنها را تشریح می‌کنیم.

۲-۱) مفاهیم پایگاه داده‌های رابطه‌ای

هرپایگاه دادهٔ رابطه‌ای، مجموعه‌ای محدود از جداول یا رابطه‌هاست.

جدول یا رابطه[2]: هر جدول یا رابطه، ساختاری دو بعدی شامل سطرها و ستونهاست. در پایگاه داده‌های رابطه‌ای برای هر نوع موجودیت یا ارتباط میان موجودیتها یک جدول در نظر گرفته می‌شود. مثلاً برای موجودیت دانشجو یک جدول و برای رابطهٔ ثبت نام(رابطهٔ میان درس و

---

1 - Relational Database

2 - Table or Relation

دانشجو) یک جدول در نظر گرفته می‌شود. هر سطر جدول نمایانگر یک موجودیت یا رابطهٔ خاص است مثلاً هر سطر از جدول دانشجو، یک دانشجوی خاص را توصیف می‌کند و یا هر سطر از جدول ثبت نام، ثبت نام یک دانشجوی مشخص در یک درس مشخص در طول یک ترم مشخص را نشان می‌دهد.

ویژگی[1]: هر یک از ستونهای یک جدول نمایانگر یکی از ویژگیهای نوع موجودیت است. مثلاً جدول دانشجو می‌تواند شامل ستونهای st# (شماره دانشجویی)، name (نام دانشجو) و course (رشتهٔ تحصیلی) دانشجو باشد.

student

st#	name	course
۷۸۰۱	آرش راد	کامپیوتر
۷۹۰۲	مینا رضایی	هنر
۸۰۰۱	علی راد	هنر
۸۰۰۲	فرزانه رسولی	هنر

تاپل[2]:هر سطر از جدول را یک تاپل نامند. مثلاً در جدول student، (هنر- مینا رضایی-۷۹۰۲) یک تاپل است.

تذکر: در پایگاه داده‌های رابطه‌ای، مفاهیم رابطه با جدول، مفاهیم تاپل با سطر و مفاهیم ویژگی با ستون کاملاً هم ارز هستند.

بدنه[3]:مجموعهٔ تاپلهای یک جدول را بدنهٔ آن جدول نامند. به عنوان مثال:

Body(student)={<۷۸۰۱,آرش راد,کامپیوتر>,
<۷۹۰۲,مینا رضایی,هنر>,
<۸۰۰۱,علی راد, هنر>,
<۸۰۰۲,فرزانه رسولی,هنر>}

---

1 - Attribute
2 - Tupple
3- Body

مجموعهٔ عنوان[1]: مجموعهٔ ویژگیهای یک جدول را مجموعهٔ عنوان آن نامند.

Header(student) = { st#, name, course }

دامنه[2] **یک ویژگی:** مجموعهٔ مقادیر مجاز برای یک ویژگی را دامنهٔ آن ویژگی نامند. به عبارت دیگر، مقادیر هر ویژگی از دامنهٔ آن ویژگی انتخاب می‌شوند. قوانین حاکم بر سیستم دامنهٔ هر ویژگی را مشخص می‌کنند. مثلاً ممکن است طبق قوانین یک دانشگاه، دامنهٔ ویژگی st# اعداد صحیح بزرگتر از ۷۷۰۰۰۰ و یا دامنهٔ ویژگی name رشته‌هایی با حداکثر طول ۲۰ کاراکتر باشد.

درجهٔ یک رابطه[3]: تعداد ویژگیهای یک جدول را درجهٔ آن جدول یا رابطه نامند مثلاً درجهٔ جدول Student ۳ است.

کاردینالیتی یک رابطه[4]: تعداد تاپلهای یک جدول در هر لحظه از حیات آن را کاردینالی آن جدول یا رابطه نامند. مثلاً کاردینالی جدول Student، ۴ است. کاردینالیتی یک رابطه، مرتباً در حال تغییر است چون دائماً تاپلهایی از جدول حذف و یا تاپلهایی در آن درج می‌شوند.

وابستگی تابعی[5]: در یک جدول، ویژگی B به ویژگی A وابستگی تابعی دارد اگر به ازاء هر مقدار برای ویژگی A حداکثر یک مقدار برای ویژگی B وجود داشته باشد. این مفهوم را بصورت $A \longrightarrow B$ نشان می‌دهیم. در این صورت، A تعیین کنندهٔ B نامیده می‌شود.

مثال ۱: روابط ذیل در جدول Student صادق می‌باشند :

---

1 - Header
2 - Domain
3 - Relation Degree
4 - Relation Cardinality
5 - Functional Dependence

$st\# \longrightarrow name, course$

این رابطه صحیح است چون هر شماره دانشجویی ما را تنها به یک نام و تنها به یک رشتهٔ تحصیلی می‌رساند (البته با این فرض که طبق قوانین دانشگاه، هر دانشجو تنها در یک رشته تحصیل کند) مثلاً این امکان وجود ندارد که شماره دانشجویی ۷۰۱۱۱۱ ما را به دو نام برساند چون هر شماره دانشجویی مربوط به یک دانشجو است نه بیشتر.

$st\#+name \longrightarrow course$

این رابطه صحیح است چون st# به تنهایی ما را تنها به یک رشتهٔ تحصیلی می‌رساند و آوردن ویژگی name در کنار آن نیز قضیه را نقض نمی‌کند. به عنوان مثال اگر شما برای اطلاع از نمرات خود تنها شماره دانشجویی خود را در اختیار کارمند قسمت آموزش دانشگاه قرار دهید، این شماره دانشجویی، کارمند قسمت آموزش را تنها به اطلاعات شما می‌رساند. حال اگر هم شماره دانشجویی و هم نام خود را در اختیار وی قرار دهید، مجدداً این شماره دانشجویی و نام، کارمند قسمت آموزش را تنها به اطلاعات شما می‌رساند.

$st\#+name + course \longrightarrow course, name$

صحت این رابطه بدیهی است! در واقع مثل این است که دانشجویی بگوید "شماره دانشجویی من ۷۸۰۱، نام من علی و رشتهٔ تحصیلی من کامپیوتر است. نام و رشتهٔ تحصیلی من چیست؟" مسلماً این اطلاعات شنونده را تنها به یک نام و یک رشتهٔ تحصیلی می‌رساند.

روابط ذیل در جدول student صحت ندارند:

$name + course \longrightarrow st\#$

این رابطه درست نیست چون با داشتن یک نام و رشتهٔ تحصیلی لزوماً به یک شماره دانشجویی واحد نمی‌رسیم. مثلاً ممکن است سه دانشجو با نام علی راد در رشتهٔ هنر تحصیل کنند پس با داشتن نام علی راد و رشتهٔ هنر به سه شماره دانشجویی مختلف می‌رسیم.

$name \longrightarrow course$

این رابطه درست نیست چون با داشتن یک نام لزوماً تنها به یک رشتهٔ تحصیلی نمی‌رسیم. مثلاً ممکن است در دانشگاه سه دانشجو با نام علی راد وجود داشته باشند که در سه رشتهٔ مختلف تحصیل کنند پس با داشتن نام علی راد به سه رشتهٔ مختلف می‌رسیم.

مثال ۲: فرض کنید جدول employee که مربوط به اطلاعات کارمندان یک شرکت است، شامل ویژگیهای ذیل باشد:

Employee (emp#, name, family,Id, birthDate, birthPlace, address, SSN)

emp#: شمارهٔ کارمندی (هر کارمند یک شمارهٔ منحصر بفرد دارد.)

name: نام کارمند

family: نام خانوادگی کارمند

Id: شماره شناسنامه

birthDate: تاریخ تولد

birthPlace: محل تولد

address: آدرس محل سکونت

SSN: شمارهٔ ملی کارمند

در این جدول، روابط ذیل صحیح هستند:

$emp\# \longrightarrow name$

این رابطه صحیح است چون با داشتن یک شمارهٔ کارمندی تنها به یک نام می‌رسیم.

$emp\# + family \longrightarrow Id$

این رابطه صحیح است چون با داشتن یک شمارهٔ کارمندی و یک نام خانوادگی تنها به یک شماره شناسنامه می‌رسیم. در واقع، شمارهٔ کارمندی به تنهایی ما را به یک شماره شناسنامهٔ واحد می‌رساند و وجود نام خانوادگی هم تغییری در این مسئله بوجود نمی‌آورد.

$SSN \longrightarrow emp\# , name, family, Id, ...$

این رابطه صحیح است چون هر شمارهٔ ملی نه تنها یک کارمند واحد در سازمان بلکه تنها یک شخص را در کل کشور مشخص می‌کند. بنابراین با داشتن یک شمارهٔ ملی تنها به یک شمارهٔ کارمندی ، تنها به یک نام، تنها به یک نام خانوادگی، تنها به یک شماره شناسنامه و ... می‌رسیم.

$Id + birthDate + birthPlace \longrightarrow emp\#\ ,\ name,\ family,...$

این رابطه صحیح است چون ترکیب شماره شناسنامه، تاریخ تولد و محل تولد تنها یک شخص را در کل کشور و مسلماً تنها یکی از کارمندان سازمان را مشخص می‌کند. بنابراین با داشتن ترکیبی از یک شماره شناسنامه، تاریخ تولد و محل تولد تنها به یک شماره کارمندی، تنها به یک نام، تنها به یک نام خانوادگی و ... می‌رسیم.

در جدول employee، روابط ذیل نادرست می‌باشند:

$name + family \longrightarrow Id$

این رابطه صحیح نیست چون با در دست داشتن یک نام و نام خانوادگی، لزوماً تنها به یک شماره شناسنامه نمی‌رسیم. مثلاً ممکن است در شرکت چندین کارمند وجود داشته باشند که نام آنها علی و نام خانوادگی آنها راد باشد. بنابراین، با در دست داشتن این نام و نام خانوادگی به چندین شماره شناسنامهٔ مختلف می‌رسیم.

$Id + birthPlace \longrightarrow name$

این رابطه صحیح نیست چون ممکن است چندین کارمند در سازمان وجود داشته باشند که محل تولد و شماره شناسنامهٔ آنها یکسان باشد. بنابراین با در دست داشتن شماره شناسنامه و محل تولد لزوماً به یک نام واحد نمی‌رسیم. مثلاً با در دست داشتن شماره شناسنامهٔ ۷۸۱ و محل تولد تهران ممکن است به چند کارمند با نامهای مختلف برسیم.

$Id \longrightarrow name,\ family$

این رابطه صحیح نیست چون ممکن است در سازمان کارمندان زیادی وجود داشته باشند که شماره شناسنامهٔ آنها یکسان باشد مثلاً با در دست داشتن شماره شناسنامهٔ ۷۸۱ ممکن است به دهها کارمند با نامها و نامهای خانوادگی مختلف برسیم.

سوپر کلید[1]:

سوپر کلید مجموعه‌ای از یک یا چند ویژگی است که سایر ویژگیهای جدول به آن وابستگی تابعی دارند.

مثال ۱: سوپر کلیدهای جدول student عبارتند از :

st #
st # + name
st # + course
st # + name + course

مثال ۲: چند نمونه از سوپر کلیدهای جدول employee عبارتند از :

emp#
emp#+name
emp#+ family
emp#+address
emp#+name+family
emp#+name+address
emp#+name+family+address+...

Id+birthDate+birthPlace
Id+birthDate+birthPlace+emp#
Id+birthDate+birthPlace+name
Id+birthDate+birthPlace+family
Id+birthDate+birthPlace+emp#+name+family+...

SSN
SSN+emp#
SSN+name
SSN+family
SSN+emp#+name+family+...

1 - Super Key

وابستگی تابعی کامل[1]:

در یک جدول، ویژگی B به ویژگی A وابستگی تابعی کامل دارد $(A \xrightarrow{FFD} B)$ اگر ویژگی B به کل ویژگی A وابستگی تابعی داشته باشد. به عبارت دیگر، ویژگی B به ویژگی A وابستگی تابعی کامل دارد اگر اولاً ویژگی B به ویژگی A وابستگی تابعی داشته باشد و ثانیاً به هیچ جزئی از آن وابستگی تابعی نداشته باشد.

مثال ۱ :

رابطهٔ ذیل در جدول student درست است:

$st\# \xrightarrow{FFD} name$

چرا که ویژگی name به ویژگی st# وابستگی تابعی دارد و به هیچ جزئی از آن وابستگی تابعی ندارد (چون اصولاً st# یک جزء است این مسئله بدیهی است)

روابط ذیل در جدول student صحت ندارند:

$name+course \xrightarrow{FFD} st\#$

این رابطه درست نیست چون st# به name+course وابستگی تابعی ندارد بنابراین واضح است که وابستگی تابعی کامل نیز ندارد.

$name \xrightarrow{FFD}$ course

این رابطه درست نیست چون course به name وابستگی تابعی ندارد بنابراین وابستگی تابعی کامل نیز ندارد.

$st\#+name \xrightarrow{FFD} course$

این رابطه درست نیست. در واقع، ویژگی course به $st\#+name$ وابستگی تابعی دارد ولی این وابستگی تابعی، کامل نیست چرا که:

$st\# \longrightarrow course$

---

1 - Full Functional Dependence

پس ویژگی course به #st که جزئی از ویژگی st#+name است وابستگی تابعی دارد. بنابراین وابستگی موجود، وابستگی جزئی[1] است و کامل نیست.

مثال ۲:

در جدول employee روابط ذیل صادقند:

$$emp\# \xrightarrow{FFD} name,\ family, Id, ...$$

این رابطه صحیح است چون ویژگیهای name و family و Id و ... به #emp وابستگی تابعی دارند و از آنجا که #emp تنها یک جزء دارد، بنابراین وابستگی جزئی نمی‌تواند وجود داشته باشد و وابستگی، کامل است.

$$SSN \xrightarrow{FFD} name,\ family, emp\#, ...$$

این رابطه صحیح است چون ویژگیهای #emp و name و family و ... به SSN وابستگی تابعی دارند و چون SSN تنها یک جزء دارد، بدون شک این وابستگی از نوع کامل است و نیازی به بررسی ندارد.

$$Id + birthDate + birthPlace \xrightarrow{FFD} emp\#,\ name,\ family, ...$$

این رابطه صحیح است چون ویژگیهای #emp و name و family و ... به ترکیب Id+birthDate+birthPlace وابستگی تابعی دارند ولی به هیچ جزئی از آن (مثل Id یا Id+birthDate یا Id+birthPlace یا birthDate+birthPlace) وابستگی تابعی ندارند پس این وابستگی از نوع کامل است.

در جدول employee ، روابط ذیل صادق نیستند:

$$emp\#+name \xrightarrow{FFD} family$$

این رابطه درست نیست چرا که:

$$emp\# \longrightarrow family$$

---

1 - Partial Dependence

به عبارت دیگر، family به #emp که قسمتی از emp#+name است وابستگی تابعی دارد. پس این وابستگی تابعی، وابستگی تابعی جزئی است و کامل نیست.

$$Id + birthPlace \xrightarrow{FFD} name$$

این رابطه درست نیست چرا که اصولاً name به Id+birthPlace وابستگی تابعی ندارد پس مسلماً وابستگی تابعی کامل نیز ندارد.

$$SSN + Id \xrightarrow{FFD} name$$

این رابطه درست نیست چرا که:

$$SSN \longrightarrow name$$

به عبارت دیگر، name به جزئی از SSN+Id وابستگی تابعی دارد، پس این وابستگی جزئی است و کامل نیست.

$$Id + birthDate + birthPlace + name \xrightarrow{FFD} family$$

این رابطه درست نیست چرا که :

$$Id + birthDate + birthPlace \longrightarrow family$$

به عبارت دیگر، family به جزئی ازId+birthDate+birthPlace+name وابستگی تابعی دارد پس این وابستگی از نوع جزئی است و کامل نیست.

کلید کاندیدا[1] :

کلید کاندیدا سوپر کلیدی است که قابل خلاصه شدن نباشد.

و یا به عبارتی دیگر:

سوپر کلیدی است که هیچ جزئی از آن سوپر کلید نباشد.

و یا به عبارتی دیگر:

1 - Candidate key

مجموعه‌ای از یک یا چند ویژگی است که سایر ویژگیهای جدول به آن وابستگی تابعی کامل دارند.

مثال ۱: در جدول student، st# یک کلید کاندیدا است ولی st#+name یک کلید کاندیدا نیست چون st# که جزئی از آن است، به تنهایی یک کلید کاندیدا می‌باشد.

مثال ۲ : کلیدهای کاندیدا برای جدول employee عبارتند از :

کلید کاندیدای ۱: emp#

کلید کاندیدای ۲: SSN

کلید کاندیدای ۳: Id+birthDate+birthPlace

کلید اصلی[1]:

طراح پایگاه داده‌ها با توجه به شرایط حاکم بر سیستم یکی از کلیدهای کاندیدا را به عنوان کلید اصلی انتخاب می‌کند. کلید اصلی، مجموعه‌ای از یک یا چند ویژگی است که برای شناسائی و تمایز میان موجودیتهای یک دسته مورد استفاده قرارمی‌گیرد مثلاً شمارهٔ دانشجویی، وسیلهٔ شناسایی و تمایز میان دانشجوهای مختلف است.

مثال ۱: در جدول student تنها یک کلید کاندیدا وجود دارد پس طراح برای تعیین کلید اصلی تنها یک انتخاب پیش رو دارد و st# را به عنوان کلید اصلی تعیین می‌کند.

مثال ۲: در جدول employee سه کلید کاندیدا وجود دارد. کلید کاندیدای سوم یعنی Id+birthDate+birthPlace بسیار طولانی است و برای شناسایی کارمندان یک شرکت مناسب نیست. کلید کاندیدای دوم یعنی SSN در حال حاضر نمی‌تواند وسیلهٔ مناسبی برای شناسایی کارمندان یک شرکت باشد چون ممکن است بسیاری از آنها هنوز برای دریافت شمارهٔ ملی اقدام

---

1 - Primary key

نکرده باشند. مناسب ترین کلید اصلی در این حالت emp# است چرا که خود شرکت می‌تواند به هر کارمند یک شمارهٔ کارمندی مناسب و منحصر بفرد اختصاص دهد.

کلید ثانویه[1]:

فرض کنید بخواهید از وضعیت نمرات خود آگاه شوید ولی شماره دانشجویی خود را فراموش کرده باشید. در این صورت می‌توانید نام+نام خانوادگی و یا نام+نام خانوادگی+رشتهٔ تحصیلی و یا نام+نام خانوادگی+سال ورود به دانشگاه و یا ... را در اختیار کارمند قسمت آموزش قرار دهید تا براساس مقادیر این ویژگیها، اطلاعات شما را جستجو کند. دلیل اینکه کارمند قسمت آموزش می‌تواند روی مقادیر این ویژگیها عمل جستجوی سریع انجام دهد، این است که طراح پایگاه داده‌ها یک کلید ثانویه براساس ترکیبی از این ویژگیها روی جدول مربوط به اطلاعات دانشجویان تعریف کرده است. به عبارت دیگر، کلید ثانویه مجموعه‌ای از یک یا چند ویژگی است که در صورت عدم دسترسی به مقدار کلید اصلی، از مقدار آن برای تسریع جستجوی اطلاعات یک موجودیت خاص استفاده می‌شود. اگرچه ممکن است مقادیر این ویژگی یک موجودیت منحصر بفرد را مشخص نکنند، ولی در یافتن آن کمک می‌کنند به عنوان مثال هنگامیکه نام+نام خانوادگی خود را در اختیار کارمند آموزش قرار می‌دهید، ممکن است کارمند آموزش به اطلاعات شما و چندین دانشجوی دیگر که اتفاقاً همنام شما هستند برسد ولی به هر حال این امکان به کارمند آموزش کمک می‌کند تا با یک یا چند پرسش سادهٔ دیگر به اطلاعات شخص شما برسد.

مثال ۱: در جدول student ، name یا name+course می‌توانند کلید ثانویه باشند.

مثال ۲: در جدول employee، موارد ذیل می‌توانند به عنوان کلید ثانویه معرفی شوند:

name + family
SSN
Id + birthDate + birthPlace

کلید خارجی[1]:

---

1 - Secondary key

اگر ویژگی A بین جدول۱ و جدول۲ مشترک و در جدول۱ کلید اصلی باشد آنگاه ویژگی A در جدول۲، کلید خارجی نسبت به جدول۱ خواهد بود.

مثال ۱: فرض کنید پایگاه داده‌های یک دانشگاه شامل جداول ذیل باشد:

city (city# , cityName)

city#: کد شهر (به هر شهر یک کد منحصر به فرد نسبت داده شده است)

cityName: نام شهر

st (st# , sname , city# , field)

st#: شماره دانشجویی

sname: نام دانشجو

city#: کد شهری که دانشجو در آن ساکن است.

field: رشتهٔ تحصیلی دانشجو (در این دانشگاه هر دانشجو، تنها در یک رشته تحصیل می‌کند.)

course (crs# , cname , unit)

crs#: شمارهٔ درس (به هر درس یک شمارهٔ منحصر به فرد داده شده است)

cname: نام درس

unit: تعداد واحد درس

enroll (st# , crs# , year , term , grade)

st#: شماره دانشجویی

crs#: شمارهٔ درس اخذ شده توسط دانشجو

year: سال اخذ درس توسط دانشجو

term: نیمسال اخذ درس توسط دانشجو (نیمسال اول یا دوم یا تابستانه)

grade: نمرهٔ اخذ شده

برای درک بهتر مسئله فرض کنید اطلاعات وارد شده در جداول به شرح ذیل باشند :

---

1 - Foreign key

city

city#	cityName
۰۰۱	تهران
۰۰۲	اصفهان
۰۰۳	شیراز
۰۰۴	رشت

st

st#	sname	city#	Field
۷۸۰۱	آرش راد	۰۰۱	هنر
۷۸۰۹	علی رضایی	۰۰۱	کامپیوتر
۸۰۰۱	عسل رسولی	۰۰۲	کامپیوتر

course

crs#	cname	unit
۱۴۰۰	پایگاه داده‌ها	۳
۱۴۰۵	مهندسی اینترنت	۳
۱۲۰۰	ادبیات	۳

enroll

st#	crs#	year	term	grade
۷۸۰۱	۱۴۰۰	۸۰	۱	۹.۵
۷۸۰۱	۱۴۰۰	۸۰	۲	۱۲
۷۸۰۱	۱۲۰۰	۸۰	۱	۲۰
۷۸۰۹	۱۴۰۰	۸۰	۱	۱۵
۷۸۰۹	۱۴۰۵	۸۱	۲	۱۸
۸۰۰۱	۱۴۰۰	۸۱	۱	۱۶

کلیدهای اصلی:

در جدول city ، city#

و در جدول st ، st#

و در جدول course ، crs#

و در جدول enroll ، st# + crs# + year + term

کلید اصلی می‌باشد.

کلیدهای خارجی:

از آنجا که ویژگی city# بین دو جدول city و st مشترک و در جدول city کلید اصلی است، بنابراین ویژگی city# در جدول st کلید خارجی نسبت به جدول city است.

از آنجا که ویژگی #st بین دو جدول st و enroll مشترک و در جدول st کلید اصلی است، پس در جدول enroll کلید خارجی نسبت به جدول st است.

از آنجا که ویژگی #crs بین دو جدول course و enroll مشترک و در جدول course کلید اصلی است، پس در جدول enroll کلید خارجی نسبت به جدول course است.

۲-۲) قوانین حاکم بر پایگاه داده‌های رابطه‌ای: قوانین جامعیت داده ای[1]

برای حفظ جامعیت داده‌ای در پایگاه داده‌های رابطه‌ای دو قانون تعریف شده‌اند:

قانون ۱: قانون جامعیت موجودیت[2]

طبق این قانون هیچ جزء از کلید اصلی در هیچ یک از تاپلها نمی‌تواند تهی[3] باشد. همچنین، مقدار کلید اصلی در تاپلها نمی‌تواند تکراری باشد. دلیل ارائه این قانون آن است که از کلید اصلی برای شناسائی موجودیتهای سیستم استفاده می‌شود بنابراین هر موجودیت برای آنکه در سیستم قابل تشخیص باشد، باید برای کلید اصلی یک مقدار منحصر بفرد داشته باشد. به عنوان مثال، در هیچ یک از تاپلهای جدول enroll که در مثال قبل ارائه شد، هیچ یک از مقادیر #st یا #crs یا year یا term نمی‌توانند تهی باشند چون این ویژگیها جزئی از کلید اصلی هستند. علاوه بر این، نمی‌توان تاپلی را در جدول enroll درج کرد که هم #st، هم #crs، هم year و هم term آن مشابه یک تاپل دیگر باشد.

قانون ۲: قانون جامعیت ارجاعی[4]

طبق این قانون، مقدار کلید خارجی در یک جدول می‌تواند تهی باشد(مشروط بر آنکه جزئی از کلید اصلی نباشد) ولی اگر تهی نباشد، بایستی با یکی از مقادیر کلید اصلی در جدول اصلی مساوی باشد.

---

1 - Data Integrity Rules
2 - Entity Integrity Rule
3 - Null
4 - Referential Integrity Rule

مثال ۱: در جدول st می‌توان دانشجویی را درج کرد که شهر سکونت وی مشخص نباشد (تهی باشد) ولی نمی‌توان دانشجویی را درج کرد که شهر سکونت وی ۰۷ باشد چرا که در جدول اصلی یعنی جدول city شهری با این کد وجود ندارد.

مثال ۲: در جدول enroll نمی‌توان تاپلی را درج کرد که شمارهٔ درس در آن ۱۶۰۰ باشد چون در جدول course درسی با این شماره وجود ندارد.

تمرینهای حل شده :

تمرین 1: فرض کنید سیستم پایگاه داده ها در یک دانشگاه شامل جداول ذیل باشد:

field(field# , fieldName)

در این جدول اطلاعات مربوط به رشته‌های تحصیلی ذخیره می‌شود:

#field: کد رشتهٔ تحصیلی (برای هر رشتهٔ تحصیلی یک کد منحصر بفرد در نظر گرفته شده است)

fieldName: نام رشتهٔ تحصیلی

type(type# , typeName , fee)

در این جدول اطلاعات مربوط به نوع دروس ذخیره می‌شود:

#type: کد نوع درس (به هر نوع درس یک کد منحصر بفرد داده شده است)

typeName: نوع درس

fee: قیمت هر واحد

student(st# , sname , startYear , field#)

در این جدول اطلاعات مربوط به دانشجویان ذخیره می‌شود:

#st: شماره دانشجویی

sname: نام دانشجو

startYear: سال ورود به دانشگاه

#field: کد رشتهٔ تحصیلی دانشجو

course(crs# , cname , unit , type#)

در این جدول اطلاعات دروس ذخیره می‌شود:

#crs: شمارهٔ درس

cname: نام درس

unit: تعداد واحد درس

#type: کد نوع درس(نظری، عملی،...)

CF(crs# , field# , kind)

در این جدول مشخص می‌شود هر درس مربوط به کدام رشته‌های تحصیلی است:

#crs: شمارهٔ درس

#field: کد رشتهٔ تحصیلی

kind: این ویژگی مشخص می‌کند درس مورد نظر برای رشتهٔ مورد نظر چه حالتی دارد('P' برای پیش نیاز، 'A' برای پایه، 'T' برای تخصصی و 'O' برای عمومی و 'E' برای اختیاری)

grades(st# , crs# , term , grade)

در این جدول اطلاعات مربوط به نمرات دانشجویان ذخیره می‌شود:

#st: شماره دانشجویی

#crs: شمارهٔ درس

term: نیمسال اخذ درس توسط دانشجو (مثلاً ۸۳۱ به معنای ترم اولِ سال ۸۳ است)

grade: نمرهٔ اخذ شده

pre(crs# , pre#)

در این جدول پیش نیازهای هر درس مشخص می‌شوند:

#crs: شمارهٔ درس

#pre: شمارهٔ درس پیش نیاز

prof (prof# , pname , degree)

در این جدول اطلاعات مربوط به اساتید ذخیره می‌شود:

#prof:شمارهٔ استاد (به هر استاد یک شمارهٔ منحصر بفرد داده شده است)

pname: نام استاد

degree: آخرین مدرک تحصیلی استاد('D' برای دکترا، 'F' برای فوق لیسانس و 'L' برای لیسانس)

PC(prof# , crs# , term)

در این جدول مشخص می‌شود هر استاد در هر نیمسال چه دروسی را تدریس کرده است:

#prof: شمارهٔ استاد

#crs: شمارهٔ درس

term: نیمسال تحصیلی

tuition(field# , startYear , constTuition)

در این جدول بر اساس سال ورود به دانشگاه، شهریهٔ ثابت هر رشتهٔ تحصیلی مشخص می‌شود:

#field: کد رشتهٔ تحصیلی

startYear: سال ورود به دانشگاه

constTuition: شهریهٔ ثابت

برای درک بهتر مسئله به نمونه‌هایی از اطلاعات ذخیره شده در پایگاه داده‌ها توجه کنید:

field

field#	fieldName
1	مهندسی کامپیوتر
2	مهندسی الکترونیک
3	ریاضی محض

type

type#	typeName	fee
1	نظری	5000
2	عملی	20000
3	آزمایشگاه	30000

tuition

field#	startYear	constTuition
1	80	75000
1	81	80000
1	82	90000
2	81	80000
2	82	90000
3	82	75000

st

st#	sname	startYear	field#
8001	آرش راد	80	1
8002	عسل شاملو	80	3
8003	سیاوش آزاد	80	1
8101	ساغر راد	81	1
8102	علی نیکی	81	1
8111	فرامرز نیکی	81	1
8112	علی رضایی	81	2

course

crs#	cname	unit	type#
1100	ادبیات	2	1
1105	تربیت بدنی	1	2
1400	برنامه سازی ۱	3	1
1402	برنامه سازی ۲	3	1
1403	ذخیره و بازیابی	3	1
1407	پایگاه داده‌ها	3	1
1500	ریاضی ۱	3	1
1600	زبان پیش نیاز	2	1

grades

st#	crs#	term	grade
8001	1400	811	9
8001	1400	812	13
8001	1401	811	13
8101	1400	821	19
8101	1500	821	14
8111	1400	811	12
8111	1401	811	20

prof

Prof#	pname	degree
101	فرانک شایسته	L
102	علی پیامی	F
106	آزاده نیکوکار	F
107	سیامک فرزانه	D
108	علی نادرنژاد	F

PC

Prof#	crs#	term
101	1400	801
101	1400	802
101	1401	801
102	1400	801
108	1500	811

CF

crs#	field#	kind
1100	1	O
1100	2	O
1100	3	O
1105	1	O
1105	2	O
1105	3	O
1400	1	T
1400	2	E
1400	3	E
1402	1	T
1403	1	T
1407	1	T
1500	1	P
1500	2	P
1500	3	P

pre

crs#	pre#
1400	1500
1402	1400
1407	1402
1407	1403

۱-۱) صحت روابط ذیل را در جدول field بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$field\# \longrightarrow fieldName$

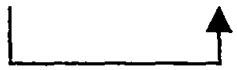
این رابطه صحیح است چون با داشتن یک کد رشته تنها به یک نام رشته می‌رسیم.

$field\# \xrightarrow{FFD} fieldName$

این رابطه صحیح است چون با داشتن یک کد رشته تنها به یک نام رشته می‌رسیم و #field قابل خلاصه شدن نیست.

نمودار وابستگی:

field(<u>field#</u> , fieldName)



تذکر: در نمودارهای وابستگی، برای مشخص کردن کلید اصلی از خط زیر استفاده می‌شود.

۱-۲) صحت روابط ذیل را در جدول type بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$type\# \longrightarrow typeName, fee$

این رابطه صحیح است چون با داشتن یک کد نوع درس تنها به یک نام نوع درس و تنها به یک قیمت واحد می‌رسیم.

$type\#+typeName \xrightarrow{FFD} fee$

این رابطه صحیح نیست چون $fee \longrightarrow type\#$. پس طرف چپ قابل خلاصه شدن است.

$type\# \xrightarrow{FFD} typeName, fee$

این رابطه صحیح است چون $type\# \longrightarrow typeName, fee$ و از طرف دیگر، #type قابل خلاصه شدن نیست.

نمودار وابستگی:

type(type# , typeName , fee)

۳-۱) صحت روابط ذیل را در جدول tuition بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$field\# \longrightarrow constTuition$

این رابطه صحیح نیست چون با داشتن یک کد رشته لزوماً به یک شهریهٔ ثابت واحد نمی‌رسیم چون نرخ شهریهٔ هر رشته برای ورودیهای مختلف متفاوت است مثلاً برای کد رشتهٔ کامپیوتر ممکن است چندین نرخ شهریه وجود داشته باشد.

$startYear \longrightarrow constTuition$

این رابطه صحیح نیست چون با داشتن یک سال ورود لزوماً به یک شهریهٔ ثابت واحد نمی‌رسیم چون نرخ شهریه برای ورودیهای هر سال در رشته‌های مختلف متفاوت است مثلاً برای ورودیهای سال ۸۳ ممکن است چندین نرخ شهریه وجود داشته باشد.

$field\#+startYear \longrightarrow constTuition$

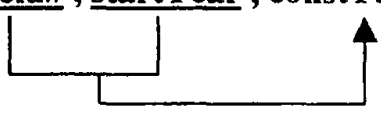
این رابطه صحیح است چون با داشتن یک کد رشته و سال ورود لزوماً به یک شهریهٔ ثابت واحد می‌رسیم.

$field\#+startYear \xrightarrow{FFD} constTuition$

این رابطه صحیح است چون $field\#+startYear \longrightarrow constTuition$ و از طرف دیگر، field#+startYear قابل خلاصه شدن نیست.

نمودار وابستگی:

tuition(field# , startYear , constTuition)



٤-۱) صحت روابط ذیل را در جدول st بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$sname + startYear \longrightarrow field\#$

این رابطه صحیح نیست چون با داشتن یک نام دانشجو و سال ورود لزوماً به یک کد رشتهٔ واحد نمی‌رسیم. مثلاً ممکن است دو دانشجوی ورودی ۸۰ با نام علی راد در دانشگاه وجود داشته باشند که یکی در رشتهٔ هنر و دیگری در رشتهٔ ریاضی تحصیل کند.

$st\# \longrightarrow sname, startYear, field\#$

این رابطه صحیح است چون با داشتن یک شماره دانشجویی تنها به یک نام، تنها به یک سال ورود و تنها به یک کد رشته می‌رسیم.

$st\# \xrightarrow{FFD} sname, startYear, field\#$

این رابطه صحیح است چون $st\# \longrightarrow sname, startYear, field\#$ و از طرف دیگر st# قابل خلاصه شدن نیست.

نمودار وابستگی:

student(st# , sname , startYear , field#)

۵-۱) صحت روابط ذیل را در جدول course بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$crs\# \longrightarrow cname, unit, type\#$

این رابطه صحیح است چون با داشتن یک شمارهٔ درس تنها به یک نام درس، تنها به یک تعداد واحد و تنها به یک کد نوع درس می‌رسیم.

$crs\# + cname \longrightarrow unit, type\#$

این رابطه صحیح است چون با داشتن یک شمارهٔ درس و نام درس تنها به یک تعداد واحد و تنها به یک کد نوع درس می‌رسیم.

$crs\# \xrightarrow{FFD} cname,unit,type\#$

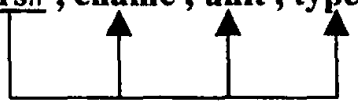
این رابطه صحیح اســـت چون $crs\# \longrightarrow cname,unit,type\#$ و از طرف دیگر crs# قابل خلاصه شدن نیست.

$crs\#+cname \xrightarrow{FFD} unit,type\#$

این رابطه صحیح نیست چون $crs\# \longrightarrow unit,type\#$. پس سمت چپ قابل خلاصه شدن است.

نمودار وابستگی:

course(crs# , cname , unit , type#)



۶-۱) صحت روابط ذیل را در جدول CF بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$crs\# \longrightarrow field\#$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ درس لزوماً تنها به یک کد رشته نمی‌رسیم. مثلاً ممکن است درس ریاضی ۱ هم مربوط به رشتهٔ ریاضی و هم مربوط به رشتهٔ کامپیوتر باشد.

$crs\# \longrightarrow kind$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ درس لزوماً تنها به یک حالت درس نمی‌رسیم. مثلاً ممکن است درس برنامه سازی ۲ برای رشتهٔ کامپیوتر تخصصی و برای رشتهٔ ریاضی اختیاری باشد.

$field\# \longrightarrow kind$

این رابطه صحیح نیست چون با داشتن یک کد رشته لزوماً تنها به یک حالت درس نمی‌رسیم چون مسلماً درسهای زیادی در یک رشته وجود دارند که هر کدام از آنها حالت خاص خود را

دارد. مثلاً در رشتهٔ کامپیوتر هم دروس پایه و هم دروس تخصصی و هم دروس اختیاری وجود دارند.

$field\# \xrightarrow{FFD} kind$

این رابطه صحیح نیست چون رابطهٔ $field\# \longrightarrow kind$ صحیح نیست.

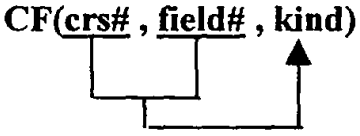
$crs\#+field\# \longrightarrow kind$

این رابطه صحیح است چون هر درس برای هر رشته تنها یک حالت می‌تواند داشته باشد مثلاً درس برنامه سازی ۲ برای رشتهٔ کامپیوتر تنها تخصصی است.

$crs\#+field\# \xrightarrow{FFD} kind$

این رابطه صحیح است چون اولاً $crs\#+field\# \longrightarrow kind$ و دوماً crs#+field# قابل خلاصه شدن نیست.

نمودار وابستگی:



۷-۱) صحت روابط ذیل را در جدول grades بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$st\# \longrightarrow grade$

این رابطه صحیح نیست چون یک دانشجو ممکن است نمرات زیادی داشته باشد پس داشتن یک شماره دانشجویی لزوماً ما را به یک نمرهٔ واحد نمی‌رساند.

$st\#+crs\# \longrightarrow grade$

این رابطه صحیح نیست چون ممکن است یک دانشجو در یک درس چندین نمره داشته باشد. مثلاً ممکن است دانشجوی ۷۸۰۱ بار اول در درس ۱۴۰۰ نمرهٔ ۸ و بار دوم نمرهٔ ۱۱ گرفته باشد. پس با داشتن یک شماره دانشجویی و یک شمارهٔ درس لزوماً به یک نمرهٔ واحد نمی‌رسیم.

$st\#+crs\# \longrightarrow term$

این رابطه صحیح نیست چون ممکن است یک دانشجو یک درس را در چند ترم گرفته باشد مثلاً ممکن است دانشجوی ۷۸۰۱ درس ۱۴۰۰ را هم در ترم ۸۲۱ و هم در ترم ۸۲۲ گرفته باشد. پس با داشتن یک شماره دانشجویی و یک شمارهٔ درس لزوماً به یک ترم واحد نمی‌رسیم.

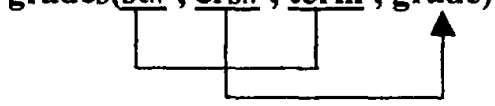
$st\#+crs\#+term \longrightarrow grade$

این رابطه صحیح است چون با داشتن یک شماره دانشجویی، یک شمارهٔ درس و یک ترم تنها به یک نمره می‌رسیم چون در هر درس در یک ترم برای هر دانشجو تنها یک نمره ثبت می‌شود.

$st\#+crs\#+term \xrightarrow{FFD} grade$

این رابطه صحیح است چون اولاً $st\#+crs\#+term \longrightarrow grade$ و دوماً طرف چپ قابل خلاصه شدن نیست.

نمودار وابستگی:



۸-۱) صحت روابط ذیل را در جدول pre بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$crs\# \longrightarrow preCrs\#$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ درس لزوماً به یک شمارهٔ درس پیش نیاز نمی‌رسیم مثلاً ممکن است درس پایگاه داده‌ها بیش از یک پیش نیاز داشته باشد.

$preCrs\# \longrightarrow crs\#$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ درس پیش نیاز لزوماً به یک شمارهٔ درس واحد نمی‌رسیم مثلاً ممکن است درس برنامه سازی ۱ پیش نیاز چند درس مختلف باشد.

$crs\#+preCrs\# \longrightarrow crs\#, preCrs\#$

صحت این رابطه بدیهی است.

$crs\#+preCrs\# \xrightarrow{FFD} crs\#, preCrs\#$

این رابطه صحیح است چون اولاً $crs\#+preCrs\# \longrightarrow crs\#, preCrs\#$ و دوماً سمت چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی:

pre(crs# , pre#)

۹-۱) صحت روابط ذیل را در جدول prof بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$prof\# \longrightarrow pname, \deg ree$

این رابطه صحیح است چون با داشتن یک شمارهٔ استاد تنها به یک نام استاد و تنها به یک آخرین مدرک تحصیلی می‌رسیم.

$prof\# \xrightarrow{FFD} pname, \deg ree$

این رابطه صحیح است چون اولاً $prof\# \longrightarrow pname, \deg ree$ و دوماً سمت چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی:

prof (prof# , pname , degree)

۱۰-۱) صحت روابط ذیل را در جدول PC بررسی کرده، نمودار وابستگی این جدول را رسم کنید:

$prof\# \longrightarrow term$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ استاد لزوماً به یک ترم واحد نمی‌رسیم چون ممکن است یک استاد چندین ترم در دانشگاه تدریس کرده باشد.

$prof\#+crs\# \longrightarrow term$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ استاد و یک شمارهٔ درس لزوماً به یک ترم واحد نمی‌رسیم چون ممکن است یک استاد یک درس را چندین ترم در دانشگاه تدریس کرده باشد.

$prof\#+term \longrightarrow crs\#$

این رابطه صحیح نیست چون با داشتن یک شمارهٔ استاد و یک ترم لزوماً به یک شمارهٔ درس واحد نمی‌رسیم چون ممکن است یک استاد در طول یک ترم چندین درس را تدریس کند.

$prof\#+crs\#+term \longrightarrow prof\#, crs\#, term$

صحت این رابطه بدیهی است.

$prof\#+crs\#+term \xrightarrow{FFD} prof\#, crs\#, term$

این رابطه صحیح است چون اولاً $prof\#+crs\#+term \longrightarrow prof\#, crs\#, term$ و دوماً سمت چپ رابطه قابل خلاصه شدن نیست.

نمودار وابستگی:

PC(<u>prof#</u> , <u>crs#</u> , <u>term</u>)

۱۱-۱) کلیدهای خارجی کلیهٔ جداول را مشخص کنید:

جدول field:

کلید خارجی: ندارد

جدول type:

کلید خارجی: ندارد

جدول tuition:

کلید خارجی: #field نسبت به جدول field

جدول st:

کلید خارجی: #field نسبت به جدول field

جدول course:

کلید خارجی: #type نسبت به جدول type

جدول CF:

کلیدهای خارجی: #crs نسبت به جدول course و #field نسبت به جدول field

جدول grades:

کلیدهای خارجی: #st نسبت به جدول st و #crs نسبت به جدول course

جدول pre:

کلیدهای خارجی: #crs نسبت به جدول course و #pre نسبت به جدول course(تشابه اسمی کلید اصلی و خارجی اهمیتی ندارد)

جدول prof:

کلید خارجی: ندارد

جدول PC:

کلیدهای خارجی: #prof نسبت به جدول prof و #crs نسبت به جدول course

تمرین ۲: سیستم حقوق و مدیریت پروژه در یک سازمان را در نظر بگیرید. در این سازمان کارمندان به دو دستهٔ رسمی و قراردادی تقسیم می‌شوند. به کارمندان رسمی‌حقوق ثابت تعلق می‌گیرد ولی حقوق کارمندان قراردادی بر اساس مبلغ ساعتی توافق شده در آخرین قرارداد منعقد شده با هر یک از آنها محاسبه می‌شود. در هر لحظه چندین پروژه در سازمان در حال اجرا است. روی هر پروژه چندین کارمند کار می‌کنند و ممکن است هر کارمند روی چند پروژه کار کند.

در هر پروژه یکی از کارمندان مدیر است. مدیر، پروژه را به تعدادی وظایف[1] کوچکتر تقسیم کرده و هر یک را به تعدادی از کارمندانی که روی پروژه کار می‌کنند محول می‌کند. هر کارمند موظف است هر روز برای هر یک از وظایفی که در حال کار روی آنهاست یک گزارش پیشرفت کار جداگانه پر کند و مشخص کند برای انجام قسمت انجام شده از وظیفهٔ مورد نظر چه مقدار زمان مصرف کرده است. حقوق کارمندان قراردادی بر اساس کل ساعاتی که در گزارشات پیشرفت کار ذکر کرده‌اند محاسبه می‌شود. فرض کنید پایگاه داده‌های این سیستم شامل جداول ذیل باشد:

emp (emp# , ename , tel , address)

در این جدول مشخصات کلی همهٔ کارمندان ذخیره می‌شود:

#emp: شمارهٔ کارمند

ename: نام کارمند

tel: شماره تلفن

address: آدرس

contract(con# , emp# , description , startDate , endDate , payPerHour)

در این جدول مشخصات قراردادهای بسته شده با کارمندان قراردادی ذخیره می‌شود:

#con: شمارهٔ قرارداد

#emp: شمارهٔ کارمند

description: شرح قرارداد

startDate: تاریخ شروع قرارداد

endDate: تاریخ خاتمهٔ قرارداد

payPerHour: حقوق پرداختی توافق شده برای هر ساعت

offEmp (emp# , startDate , pay)

در این جدول مشخصات خاص کارمندان رسمی ذخیره می‌شود:

#emp: شمارهٔ کارمند

---

1 - Task

startDate: تاریخ استخدام

pay: حقوق ثابت

profession (p# , pname)

#p: شمارهٔ تخصص

pname: نام تخصص

Emp_profession (emp# , p# , degree)

در این جدول مشخص می‌شود هر کارمند چه تخصصی دارد(ممکن است یک کارمند چند تخصص داشته باشد):

#emp: شمارهٔ کارمند

#p: شمارهٔ تخصص

degree: سطح مهارت کارمند مورد نظر در تخصص مورد نظر

project (proje# , pname , headEmp# , status)

در این جدول اطلاعات پروژه‌ها ذخیره می‌شود:

#proje: شمارهٔ پروژه

pname: نام پروژه

#headEmp: شمارهٔ مدیر پروژه

status: وضعیت انجام پروژه(در حال اجرا، در حال پشتیبانی،...)

task (proje# , task# , tname , startDate , endDate)

در این جدول اطلاعات وظایف ذخیره می‌شود:

#proje: شمارهٔ پروژه

#task: شمارهٔ وظیفه

tname: عنوان وظیفه

startDate: تاریخ شروع وظیفه

endDate: تاریخ خاتمهٔ وظیفه

Emp_Task (proje# , task# , emp#)

در این جدول مشخص می‌شود کدام کارمندان روی کدام وظایف کار می‌کنند:

#proje: شمارهٔ پروژه

task#: شمارهٔ وظیفه

emp#: شمارهٔ کارمندی که در وظیفه مشارکت دارد

report (report# , emp# , proje# , task# , rdate , description , hours)

در این جدول اطلاعات وارد شده در گزارشات پیشرفت کار ذخیره می‌شود:

report#: شمارهٔ گزارش

emp#: شمارهٔ کارمند گزارش دهنده

proje#: شمارهٔ پروژه

task#: شمارهٔ وظیفه‌ای که کارمند در بارهٔ آن گزارش می‌دهد

rdate: تاریخ گزارش

description: شرح گزارش

hours: تعداد ساعات صرف شده برای انجام کار مذکور در گزارش

کلیدهای اصلی و خارجی کلیهٔ جداول را مشخص کنید:

جدول emp:

کلید اصلی: emp#

کلید خارجی: ندارد

جدول contract:

کلید اصلی: con#

کلید خارجی: emp# نسبت به جدول emp

جدول offEmp:

کلید اصلی: emp#

کلید خارجی: emp# نسبت به جدول emp

جدول profession:

کلید اصلی: p#

کلید خارجی: ندارد

جدول Emp_profession:

کلید اصلی: #p+#emp

کلید خارجی: #emp نسبت به جدول emp و #p نسبت به جدول profession

جدول project:

کلید اصلی: #proje

کلید خارجی: #headEmp نسبت به جدول emp

جدول task :

کلید اصلی: #task+#proje

کلید خارجی: #proje نسبت به جدول project

جدول Emp_task:

کلید اصلی: #emp+#task+#proje

کلید خارجی: #task+#proje نسبت به جدول task و #emp نسبت به جدول emp

جدول report:

کلید اصلی: #report

کلید خارجی: #task+#proje نسبت به جدول task و #emp نسبت به جدول emp

تمرینهای فصل:

۱- کلیهٔ وابستگیهای تابعی و وابستگیهای تابعی کامل در هر یک از جداول نمونه سوال ۲ را مشخص کنید.

۲- فرض کنید پایگاه داده‌های یک وب سایت اطلاع رسانی در مورد بازیهای جام جهانی فوتبال شامل جداول ذیل باشد:

worldcup (cup# , startDate , endDate , place , team1# , team2# , team3#)

در این جدول، اطلاعات مربوط به جامهای مختلف ذخیره می‌شود:

#cup: شمارهٔ جام جهانی(سال برگزاری جام جهانی)

startDate: تاریخ شروع

endDate: تاریخ خاتمه

place: محل برگزاری

#team1: تیم قهرمان اول

#team2: تیم قهرمان دوم

#team3: تیم قهرمان سوم

team (team# , country , history)

در این جدول، اطلاعات مربوط به تیمهای کشورهای مختلف ذخیره می‌شود:

#team: شمارهٔ تیم(به تیم هر کشور یک شمارهٔ منحصر بفرد داده شده است)

country: کشور

history: تاریخچهٔ تیم

cup_team (cup# , team# , color)

در این جدول مشخص می‌شود هر تیم در چه جامهایی شرکت داشته است:

#cup: شمارهٔ جام

#team: شمارهٔ تیم

color: رنگ پیراهن

person (id , name , biography)

در این جدول اطلاعات مربوط به بازیکنان، مربیان و داوران ذخیره می‌شود. (ممکن است یک شخص در جامهای مختلف با سمتهای مختلف حضور یابد مثلاً در یک جام بازیکن و در جام دیگر مربی باشد):

id: شناسهٔ شخص (به هر شخص، فارغ از سمت یا سمتهایی که در جامهای مختلف داشته است یک شناسهٔ منحصر بفرد اختصاص یافته است)

name: نام شخص

biography: زندگینامه

team_player (cup#, team# , id , role , number)

در این جدول، بازیکنان تیمها در هریک از جامها مشخص می‌شوند (در هر جام، هر تیم حداقل ۲۰ بازیکن دارد ولی هر بازیکن تنها در یک تیم بازی می‌کند) :

cup#: شمارهٔ جام

team#: شمارهٔ تیم

id: شناسهٔ شخص(بازیکن)

role: نقش بازیکن(دفاع ، ذخیره، حمله و ...)

number: شمارهٔ پیراهن

team_coatch (cup# , team# , id , role)

در این جدول، مربیان هر تیم در هریک از جامها مشخص می‌شوند(در هر جام، هر تیم حداقل یک مربی دارد ولی هر مربی تنها مربی گری یک تیم را بر عهده دارد) :

cup#: شمارهٔ جام

team#: شمارهٔ تیم

id: شناسهٔ شخص(مربی)

role: نقش مربی (سرمربی، کمک مربی و ...)

play (cup# , play# , pdate , team1# , team2# , result)

در این جدول، اطلاعات کلیهٔ بازیهای جامهای مختلف ذخیره می‌شود:

cup#: شمارهٔ جام

play#: شمارهٔ بازی

pdate: تاریخ بازی

team1#: شمارهٔ تیم ۱

team2#: شمارهٔ تیم ۲

result: نتیجهٔ بازی

play_referee (cup# , play# , id , role)

در این جدول، داوران بازیها مشخص می‌شوند (هر بازی چند داور دارد):

cup#: شمارهٔ جام

play#: شمارهٔ بازی

id: شناسهٔ شخص(داور)

role: نقش داور (داور یا کمک داور یا ...)

goals (cup# , play#, goal# , id , time)

در این جدول، اطلاعات گلهای زده شده در بازیهای مختلف ذخیره می‌شود:

cup#: شمارهٔ جام

play#: شمارهٔ بازی

goal#: شمارهٔ گل زده شده (به هریک از گلهای زده شده در بازی یک شماره داده شده است)

id: شناسهٔ بازیکنی که گل را به ثمر رسانده است

time: دقیقه‌ای که گل زده شده است

penalty (cup# , play# , penalty# , id , type , time , punishment)

در این جدول، اطلاعات خطاهای صورت گرفته در بازیهای مختلف ذخیره می‌شود:

cup#: شمارهٔ جام

play#: شمارهٔ بازی

penalty#: شمارهٔ خطا(به هر یک از خطاهای رخ داده در بازی یک شماره داده شده است)

id: شناسهٔ بازیکنی که خطا را مرتکب شده است

type: نوع خطا(برخورد بدنی، تماس دست با توپ و ...)

time: دقیقه‌ای که در آن خطا صورت گرفته است

punishment: مجازات (کارت قرمز، کارت زرد، ...)

۲-۱) برای هر جدول کلیهٔ وابستگیهای تابعی و وابستگیهای تابعی کامل را مشخص کنید.

۲-۲) کلید اصلی هر جدول را مشخص کنید.

۲-۳) برای هر جدول کلیدهای خارجی را مشخص کنید(در صورت وجود).

۲-۴) برای هر جدول کلیدهای ثانویه را مشخص کنید(در صورت وجود).

۳- فرض کنید پایگاه داده‌های یک سایت اینترنتی فروش اسباب بازی از جداول ذیل تشکیل شده باشد:

company (company# , cname)

در این جدول اطلاعات شرکتهای سازندهٔ اسباب بازی ذخیره می‌شود:

company#: شمارهٔ شرکت سازنده

cname: نام شرکت سازنده

toy (toy# , name , company# , fee , count)

در این جدول اطلاعات مربوط به اسباب بازیهای مختلف ذخیره می‌شود:

toy#: کد اسباب بازی(به هر اسباب بازی یک شمارهٔ منحصر بفرد داده شده است)

name: نام اسباب بازی

company#: شمارهٔ شرکت سازنده

fee: قیمت واحد

count: تعداد موجود در انبار

age (age# , startAge , endAge)

در این جدول به هر گروه سنی یک کد اختصاص یافته است مثلاً گروه سنی ب ، سنین بین ۳ تا ۵ سال را شامل می‌شود:

age#: کد گروه سنی

startAge: سن شروع

endAge: سن پایان

toy_age (toy# , age#)

در این جدول مشخص می‌شود هر اسباب بازی مخصوص چه گروه یا گروههای سنی است:

#toy: کد اسباب بازی

#age: کد گروه سنی

toy_type (toy# , type)

در این جدول نوع یا انواع هر اسباب بازی مشخص می‌شود. ممکن است یک اسباب بازی جزء چند نوع اسباب بازی باشد مثلاً هم ورزشی و هم فکری باشد:

#toy: کد اسباب بازی

type: نوع اسباب بازی (ورزشی، فکری...)

factor (fac# , fdate , address , tel , cc# , ccType)

در این جدول اطلاعات سربرگ فاکتورهای صادر شده برای مشتریان ذخیره می‌شود:

#fac: شمارهٔ فاکتور

fdate: تاریخ فاکتور

address: آدرس مشتری

tel: شماره تلفن مشتری

#cc: شمارهٔ کارت اعتباری

ccType: نوع کارت اعتباری

fac_toy (fac# , toy# , qty)

در این جدول ریز فاکتورهای صادر شده برای مشتریان ذخیره می‌شود:

#fac: شمارهٔ فاکتور

#toy: کد اسباب بازی

qty: تعداد خرید

۱-۳) برای هر جدول کلیهٔ وابستگیهای تابعی و وابستگیهای تابعی کامل را مشخص کنید.

۲-۳) کلید اصلی هر جدول را مشخص کنید.

۳-۳) برای هر جدول کلیدهای خارجی را مشخص کنید(در صورت وجود).

۴-۳) برای هر جدول کلیدهای ثانویه را مشخص کنید(در صورت وجود).

۴- فرض کنید پایگاه داده‌های یک وب سایت اطلاع رسانی در مورد سینما از جداول ذیل تشکیل شده باشد:

film (film# , fname , year , subject)

در این جدول اطلاعات فیلمها ذخیره می‌شود:

film# : شمارهٔ فیلم

fname: نام فیلم

year: سال ساخت

subject: موضوع فیلم (اجتماعی، خانوادگی...)

people(id , name , biography)

در این جدول مشخصات کلیهٔ کارگردانان، بازیگران، نویسندگان و تهیه‌کنندگان نگهداری می‌شود. به هر شخص یک شناسهٔ منحصر بفرد داده شده است. یک شخص ممکن است مثلاً هم کارگردان و هم بازیگر و هم نویسنده باشد ولی اطلاعات وی تنها یکبار ثبت می‌شود.

id: شناسهٔ شخص

name: نام

biography: زندگینامهٔ شخص

Film_Director (film# , id)

در این جدول مشخص می‌شود چه کسانی فیلم را کارگردانی کرده‌اند:

film#: شمارهٔ فیلم

id: شمارهٔ شخصی که فیلم را کارگردانی کرده است (توجه کنید یک فیلم ممکن است بیش از یک کارگردان داشته باشد.)

Film_Writer (film# , id)

در این جدول مشخص می‌شود چه کسانی فیلمنامهٔ فیلم را نوشته‌اند:

film#: شمارهٔ فیلم

id: شمارهٔ شخصی که فیلمنامهٔ فیلم را نوشته است (توجه کنید یک فیلم ممکن است بیش از یک نویسنده داشته باشد.)

Film_Actor (film# , id , role)

در این جدول مشخص می‌شود چه کسانی در فیلم بازی کرده‌اند:

#film: شمارهٔ فیلم

id: شمارهٔ شخصی که در فیلم بازی کرده است (توجه کنید یک فیلم بیش از یک بازیگر دارد.)

role: نوع نقش (نقش اول یا دوم...)

Film_Producer (film# , id)

در این جدول مشخص می‌شود چه کسانی فیلم را تهیه کرده‌اند:

#film: شمارهٔ فیلم

id: شمارهٔ شخصی که فیلم را تهیه کرده است (توجه کنید یک فیلم ممکن است بیش از یک تهیه کننده داشته باشد.)

award (award# , aname)

در این جدول اطلاعات مربوط به انواع جایزه‌ها ذخیره می‌شود:

#award: کد جایزه (به هر نوع جایزه مانند بهترین کارگردانی، بهترین بازیگر نقش اول زن و ... یک کد اختصاص یافته است.)

aname: نام جایزه

festival (fes# , fname , year , place)

در این جدول اطلاعات مربوط به جشنواره‌های مختلف ذخیره می‌شود:

#fes: شمارهٔ جشنواره

fname: نام جشنواره

year: سال برگزاری جشنواره

place: محل برگزاری جشنواره

Fes_Film (fes# , film#)

در این جدول مشخص می‌شود در هر جشنواره چه فیلمهایی شرکت کرده‌اند. در هر جشنواره فیلمهای متعددی شرکت می‌کنند و هر فیلم ممکن است در جشنواره‌های مختلف شرکت کند.

fes#: شمارهٔ جشنواره

film#: شمارهٔ فیلم

grant (fes# , film# , award# , id , kind , result)

در این جدول اطلاعات مربوط به مقامهایی که فیلمهای مختلف در جشنواره‌های مختلف کسب کرده‌اند ذخیره می‌شود. اطلاعات ذخیره شده باید نمایانگر اطلاعاتی به فرم ذیل باشد:

در جشنوارهٔ جشن خانهٔ سینما-دورهٔ چهارم، سلیمه رنگزن برای بازی در فیلم عروس آتش برندهٔ جایزهٔ بهترین بازیگر نقش دوم زن شد و دیپلم افتخار گرفت.

fes#: شمارهٔ جشنواره

film#: شمارهٔ فیلم

award#: کد نوع جایزه

id: شمارهٔ شخصی که جایزه به وی اختصاص یافته است

kind: نوع جایزه (دیپلم افتخار، سیمرغ بلورین...)

result: نتیجه(برنده یا کاندیدا)

۴-۱) برای هر جدول کلیهٔ وابستگیهای تابعی و وابستگیهای تابعی کامل را مشخص کنید.

۴-۲) کلید اصلی هر جدول را مشخص کنید.

۴-۳) برای هر جدول کلیدهای خارجی را مشخص کنید(در صورت وجود).

۴-۴) برای هر جدول کلیدهای ثانویه را مشخص کنید(در صورت وجود).

۳

جبر رابطه‌ای

جبر رابطه‌ای مجموعه‌ای از عملگرها و قوانینی است که برای پردازش رابطه‌ها یا جداول بکار می‌روند. این عملگرها عبارتند از :

۱- عملگرهای یکتایی[1]: ورودی این عملگرها تنها یک جدول است. این عملگرها عبارتند از: عملگر انتخاب (ó یا Select)، عملگر پرتو (∏ یا Project)، عملگر بسط (Extend)، عملگر گروه‌بندی یا خلاصه‌سازی(Summarize)

۲- عملگرهای دوتایی[2] که روی دو جدول سازگار اعمال می‌شوند: تنها دو جدول سازگار می‌توانند ورودی این عملگرها باشند. این عملگرها عبارتند از : اجتماع(U یا Union)، اشتراک (∩ یا Intersect) و تفاضل (- یا Subtract)

۳- عملگرهای دوتایی که روی دو جدول دلخواه اعمال می‌شوند: هر دو جدول دلخواه می‌توانند ورودی این عملگرها باشند. این عملگرها عبارتند از ضرب (× یا Product) و انواع مختلف پیوند (join)

---

1 - Unary

2 - Binary

٤- عملگر تقسیم(÷ یا divide): این عملگر روی دو جدول با شرایط خاص اعمال می‌شود.

٥- عملگر جایگزینی(←)

تذکر: در جبر رابطه‌ای ورودی و خروجی کلیهٔ عملگرها از نوع جدول می‌باشد.

در این فصل با ذکر مثالهای متعدد عملگرهای جبر رابطه‌ای را مورد بررسی قرار می‌دهیم. در اغلب مثالها از جداول یک پایگاه داده‌ها شامل چهار جدول ذیل استفاده شده است:

S (s# , sname , city)

در این جدول اطلاعات تولید کنندگان ذخیره می‌شود:

#s: کد تولید کننده(هر تولید کننده یک کد منحصر بفرد دارد)

sname: نام تولید کننده

city: شهری که تولید کننده در آن قرار دارد.

P (p# , pname , color)

در این جدول اطلاعات محصولات ذخیره می‌شود:

#p: کد محصول (هر محصول یک کد منحصر بفرد دارد)

pname: نام محصول

color: رنگ محصول

J (j# , jname , city)

در این جدول اطلاعات پروژه‌ها ذخیره می‌شود:

#j: کد پروژه (هر پروژه یک کد منحصر بفرد دارد.)

jname: نام پروژه

city: شهری که پروژه در آن اجرا می‌شود

SPJ (s# , p# , j# , qty)

در این جدول مشخص می‌شود هر تولید کننده چند کیلوگرم از هر محصول را برای هر پروژه تولید کرده است:

#s: کد تولید کننده

#p: کد محصول

#j: کد پروژه

qty: مقدار فروش

S

s#	sname	city
S1	تهران مصالح	تهران
S2	یزد مصالح	یزد
S3	البرز	اصفهان
S4	ایران مصالح	تهران

P

p#	pname	color
P1	تیرآهن	مشکی
P2	آرماتور	مشکی
P3	سیمان	طوسی
P4	آلومنیم	سفید

J

j#	jname	city
J1	مرمت آثار باستانی	شیراز
J2	مرمت آثار باستانی	اصفهان
J3	فرودگاه	تهران

SPJ

s#	p#	J#	qty
S1	P1	J1	12000
S1	P2	J1	20000
S1	P3	J1	3000
S1	P4	J1	8000
S1	P1	J2	10000
S1	P1	J3	9000
S2	P1	J1	2000
S2	P1	J3	5000
S2	P3	J1	8000
S3	P1	J1	9000
S3	P2	J3	9000
S3	P1	J3	4000

۱-۳) عملگرهای یکتایی

۱-۱-۳) عملگر انتخاب: از این عملگر برای گزینش افقی در یک جدول استفاده می‌شود. به عبارت دیگر، با استفاده از این عملگر می‌توان تاپلهایی از یک جدول را که دارای شرط بخصوصی هستند انتخاب کرد.

جدول R را در نظر بگیرید:

R

A	B	C	D
a	b	c	10
e	f	a	20
g	h	a	10
l	m	b	30

مثال ۱: تاپلهایی از جدول R را انتخاب کنید که در آنها D>15 است.

این دستور را می‌توان به یکی از صورتهای ذیل نوشت :

ó (R)
D>15

یا:

select R where D>15

یا:

R where D>15

نتیجه:

A	B	C	D
e	f	a	20
l	m	b	30

در شرط انتخاب می‌توان از and و or و not استفاده کرد.

مثال ۲: تاپلهایی از جدول R را انتخاب کنید که در آنها D > 15 و C = 'a' است.

ó (R)
D>15 and C='a'

یا:

select R where D>15 and C='a'

یا:

R where D>15 and C='a'

نتیجه:

A	B	C	D
e	f	a	20

مثال ۳: مشخصات تولید کنندگان تهرانی را بیابید.

کافیست از جدول S تاپلهایی را انتخاب کنیم که مربوط به شهر تهران هستند:

ó (S)
city='تهران'

یا:

select S where city='تهران'

یا:

S where city='تهران'

نتیجه:

s#	sname	city
S1	تهران مصالح	تهران
S4	ایران مصالح	تهران

مثال ٤: مشخصات فروشهای مربوط به محصولات 'P2' و 'P3' و 'P4' را بیابید.

ó (SPJ)
p#='P2' or p#='P3' or p#='P4'

یا:

select SPJ where p#='P2' or p#='P3' or p#='P4'

یا:

SPJ where p#='P2' or p#='P3' or p#='P4'

نتیجه:

s#	p#	j#	qty
S1	P2	J1	20000
S1	P3	J1	3000
S1	P4	J1	8000
S2	P3	J1	8000
S3	P2	J3	9000

مثال ۵: مشخصات فروشهایی را بیابید که مربوط به محصولات 'P2' یا 'P3' یا 'P4' نیستند.

ó (SPJ)
p#<>'P2' and p#<>'P3' and p#<>'P4'

یا:

select SPJ where p#<>'P2' and p#<>'P3' and p#<>'P4'

یا:

SPJ where p#<>'P2' and p#<>'P3' and p#<>'P4'

نتیجه:

s#	p#	j#	qty
S1	P1	J1	12000
S1	P1	J2	10000
S1	P1	J3	9000
S2	P1	J1	2000
S2	P1	J3	5000
S3	P1	J1	9000
S3	P1	J3	4000

۲-۱-۳) عملگر پرتو:

از این عملگر برای گزینش عمودی در یک جدول استفاده می‌شود. به عبارت دیگر، با استفاده از این عملگر می‌توان ستونهای موردنظر از یک جدول را انتخاب کرد.

مثال ۱: ستونهای A و D از جدول R را انتخاب کنید.

این دستور را می‌توان به یکی از صورتهای ذیل نوشت :

∏ (R)
A,D

یا:

R[A,D]

یا:

Project R over A,D

نتیجه:

A	D
a	10
e	20
g	10
l	30

مثال ۲: نام کلیهٔ شهرهایی را بیابید که تولید کننده‌ای در آنها ساکن است.

کافیست ستون city از جدول S را انتخاب کنیم:

$$\prod_{city}(S)$$

یا:

S[city]

یا:

project S over city

نتیجه:

city
تهران
یزد
اصفهان

۳-۱-۳) عملگر بسط

از عملگر بسط برای اضافه کردن یک یا چند ستون به یک جدول استفاده می‌شود. مقادیر این ستونها بر اساس مقادیر ستونهای دیگر محاسبه می‌شود. قالب کلی استفاده از این عملگر به شکل ذیل است:

extend نام جدول add(عبارت محاسباتی) as نام ستون جدید

مثال: لیستی از کلیهٔ ستونهای جدول SPJ و یک ستون جدید تهیه کنید که میزان فروش بر حسب تن را نشان دهد(qty بر حسب کیلوگرم است).

extend SPJ add(qty/1000) as tonQty

نتیجه:

s#	p#	j#	qty	tonQty
S1	P1	J1	12000	12
S1	P2	J1	20000	20
S1	P3	J1	3000	3
S1	P4	J1	8000	8
S1	P1	J2	10000	10
S1	P1	J3	9000	9
S2	P1	J1	2000	2
S2	P1	J3	5000	5
S2	P3	J1	8000	8
S3	P1	J1	9000	9
S3	P2	J3	9000	9
S3	P1	J3	4000	4

٤-١-٣) عملگر خلاصه‌سازی یا گروه‌بندی

با استفاده از این عملگر می‌توان تاپلهای یک جدول را بر اساس مقادیر یک یا چند ستون گروه‌بندی کرد و روی هر گروه بصورت مجزا محاسبه‌ای را انجام داد. از تابع sum برای محاسبهٔ مجموع، از تابع avg برای محاسبهٔ میانگین، از تابع max برای محاسبهٔ ماکزیمم و از تابع min برای محاسبهٔ مینیمم مقادیر یک ستون از هر گروه استفاده می‌شود. همچنین، با استفاده از تابع count می‌توان تعداد تاپلهای هر گروه را محاسبه کرد.

مثال ۱: لیستی از کد محصولات و مقدار کل تولید هر یک از آنها تهیه کنید.

برای بدست آوردن این لیست کافیست در جدول SPJ مجموع ستون qty را برای هر #p بطور مجزا محاسبه کنیم. در واقع، ابتدا تاپلهای SPJ را بر اساس #p گروه‌بندی کرده و سپس sum(qty) را برای هر گروه حساب می‌کنیم:

summarize SPJ by(p#) add sum(qty) as total

p#	total
P1	51000
P2	29000
P3	11000
P4	8000

مثال ۲: لیستی از کد محصولات، کد پروژه‌ها و میزان کل فروش محصول مورد نظر برای پروژهٔ مورد نظر تهیه کنید.

در اینجا باید میزان کل فروش را برای هر جفت کد محصول/کد پروژه بطور جداگانه محاسبه کنیم. پس لازم است لیست را بر اساس #p و #j گروه‌بندی کنیم:

summarize SPJ by(p#,j#) add sum(qty) as total

p#	j#	total
P1	J1	23000
P2	J1	20000
P3	J1	11000
P4	J1	8000
P1	J2	10000
P1	J3	18000
P2	J3	9000

۲-۳) عملگرهای اجتماع، اشتراک و تفاضل

عملگرهای اجتماع و اشتراک و تفاضل تنها روی جداول سازگار قابل اعمال هستند. جداول سازگار جداولی هستند که تعداد ویژگیها و دامنهٔ ویژگیهای متناظر آنها یکسان باشد. جداول T1 و T2 را در نظر بگیرید:

T1

A	B	C	D
a	b	e	10
f	o	p	20
l	m	x	10
p	q	r	10

T2

A	B	C	D
a	b	e	10
z	w	y	20

مجموعهٔ عنوان دو جدول یکسان است. با این فرض که دامنهٔ ویژگیها نیز نظیر به نظیر یکسان باشند، دو جدول سازگارند.

۱-۲-۳) عملگر اجتماع: اجتماع دو جدول سازگار جدولی خواهد بود که مجموعهٔ عنوان آن با مجموعهٔ عنوان دو جدول یکسان است و بدنهٔ آن شامل تاپلهای هر دو جدول می‌باشد(تاپلهای تکراری حذف می‌شوند). برای عملگر اجتماع از نماد $\cup$ یا union استفاده می‌شود.

مثال ۱: $T1 \cup T2$ را بدست آورید.

$T1 \cup T2$

A	B	C	D
A	b	e	10
F	o	p	20
L	m	x	10
P	q	r	10
Z	w	y	20

مثال ۲: نام شهرهایی را بیابید که تولید کننده‌ای در آنها زندگی می‌کند یا پروژه‌ای در آنها در حال اجراست.

کافیست اجتماع ستون city از جدول S و ستون city از جدول J را بدست آوریم:

$S[city] \cup J[city]$

یا:

S[city] union J[city]

نتیجه:

city
تهران
یزد
اصفهان
شیراز

عملگر اجتماع دارای خاصیت جابجایی و شرکت پذیری است. به عبارت دیگر :

$T1 \cup T2 \equiv T2 \cup T1$

$T1 \cup (T2 \cup T3) \equiv (T1 \cup T2) \cup T3$

۲-۲-۳) عملگر اشتراک: اشتراک دو جدول سازگار جدولی خواهد بود که مجموعهٔ عنوان آن با مجموعهٔ عنوان دو جدول یکسان است و بدنهٔ آن شامل تاپلهای مشترک بین دو جدول می‌باشد. برای عملگر اشتراک از نماد ∩ یا intersect استفاده می‌شود.

مثال ۱: T1 ∩ T2 را بدست آورید.

T1 ∩ T2

A	B	C	D
a	b	e	10

مثال ۲: نام شهرهایی را بیابید که هم تولید کننده‌ای در آنها زندگی می‌کند و هم پروژه‌ای در آنها در حال اجراست.

کافیست اشتراک ستون city از جدول S و ستون city از جدول J را بدست آوریم:

S[city] ∩ J[city]

یا:

S[city] intersect J[city]

نتیجه:

city
تهران
اصفهان

عملگر اشتراک دارای خاصیت جابجایی و شرکت پذیری است. به عبارت دیگر :

$T1 \cap T2 \equiv T2 \cap T1$

$T1 \cap (T2 \cap T3) \equiv (T1 \cap T2) \cap T3$

۳-۲-۳) عملگر تفاضل: نتیجهٔ تفاضل دو جدول سازگار(T1-T2) جدولی خواهد بود که مجموعهٔ عنوان آن با مجموعهٔ عنوان دو جدول یکسان است و بدنهٔ آن شامل تاپلهایی از جدول T1 است که در جدول T2 وجود ندارند.

مثال ۱: T1-T2 را بدست آورید.

T1 - T2

A	B	C	D
f	o	P	20
l	m	X	10
p	q	r	10

مثال ۲: نام شهرهایی را بیابید که تولید کننده‌ای در آنها قرار دارد ولی پروژه‌ای در آنها در حال اجرا نیست.

کافیست تفاضل ستون city از جدول S و ستون city از جدول J را بدست آوریم:

S[city] - J[city]

نتیجه:

city
یزد

عملگر تفاضل خاصیت جابجایی و شرکت پذیری ندارد. به عبارت دیگر:

$T1 - T2 \neq T2 - T1$

$T1 - (T2 - T3) \neq (T1 - T2) - T3$

۳-۳) عملگرهای ضرب و پیوند:

۱-۳-۳) عملگر ضرب: حاصلضرب دو جدول دلخواه، جدولی خواهد بود شامل ویژگیهای هر دو جدول(حتی ویژگیهای تکراری) و تاپلهایی که از ضرب دکارتی تاپلهای دو جدول بدست

می‌آیند. به عبارت دیگر، هر یک از تاپلهای یک جدول در کنار هر یک از تاپلهای جدول دیگر قرار می‌گیرند. بنابراین، اگر یک جدول با ۳ تاپل را در یک جدول با ۵ تاپل ضرب کنیم، جدول نتیجه شامل ۱۵=۵×۳ تاپل خواهد بود. برای عملگر ضرب از نماد × یا times استفاده می‌شود.

مثال ۱: دو جدول R1 و R2 را در نظر بگیرید و R1×R2 را بدست آورید.

R1

A	B	C	D
a	b	c	10
p	q	r	20
l	m	n	30
a	b	e	30

R2

D	E	F
10	20	a
15	60	b

R1×R2

A	B	C	R1.D	R2.D	E	F
a	b	c	10	10	20	a
a	b	c	10	15	60	b
p	q	r	20	10	20	a
p	q	r	20	15	60	b
l	m	n	30	10	20	a
l	m	n	30	15	60	b
a	b	e	30	10	20	a
a	b	e	30	15	60	b

مثال ۲: S×J را بدست آورید.

s#	sname	S.city	j#	jname	J.city
S1	تهران مصالح	تهران	J1	مرمت آثار باستانی	شیراز
S1	تهران مصالح	تهران	J2	مرمت آثار باستانی	اصفهان
S1	تهران مصالح	تهران	J3	فرودگاه	تهران
S2	یزد مصالح	یزد	J1	مرمت آثار باستانی	شیراز
S2	یزد مصالح	یزد	J2	مرمت آثار باستانی	اصفهان
S2	یزد مصالح	یزد	J3	فرودگاه	تهران
S3	البرز	اصفهان	J1	مرمت آثار باستانی	شیراز
S3	البرز	اصفهان	J2	مرمت آثار باستانی	اصفهان
S3	البرز	اصفهان	J3	فرودگاه	تهران
S4	ایران مصالح	تهران	J1	مرمت آثار باستانی	شیراز
S4	ایران مصالح	تهران	J2	مرمت آثار باستانی	اصفهان
S4	ایران مصالح	تهران	J3	فرودگاه	تهران

عملگر ضرب دارای خاصیت جابجایی و شرکت پذیری است. به عبارت دیگر:

$$R1 \times R2 \equiv R2 \times R1$$
$$R1 \times (R2 \times R3) \equiv (R1 \times R2) \times R3$$

۲-۳-۳)عملگر پیوند[1]

عملگر پیوند روی هر دو جدول دلخواه قابل اعمال است. این عملگر دارای انواع مختلف است:

۱-۲-۳-۳) پیوند شرطی یا پیوند تتا[2]: این عملگر تاپلهایی از حاصلضرب دو جدول را که دارای شرط بخصوصی هستند انتخاب می‌کند. برای پیوند شرطی از نماد ∞ یا join استفاده می‌شود.

---

1 - Join

2 - θ-Join

مثال ۱: R1 join R2 را بدست آورید.

R1.D<E

مرحلهٔ ۱: بدست آوردن R1 × R2 (نتیجهٔ این مرحله را قبلاً بدست آورده‌ایم)

مرحلهٔ ۲: انتخاب تاپلهایی که در آنها R1.D<E است:

A	B	C	R1.D	R2.D	E	F
a	b	c	10	10	20	a
a	b	c	10	15	60	b
p	q	r	20	15	60	b
l	m	n	30	15	60	b
a	b	e	30	15	60	b

تذکر: برای جلوگیری از اتلاف وقت می‌توان نتیجهٔ پیوند شرطی دو جدول را تنها در یک مرحله بدست آورد. برای این کار کافیست تنها تاپلهایی از دو جدول را در هم ضرب کنیم که شرط مورد نظر را دارا هستند.

مثال ۲: SPJ join S را بدست آورید.

qty<4000

S.s#	Sname	city	SPJ.s#	p#	j#	qty
S1	تهران مصالح	تهران	S1	P3	J1	3000
S2	یزد مصالح	یزد	S2	P1	J1	2000

عملگر پیوند شرطی دارای خاصیت جابجایی است ولی شرکت پذیری ندارد. به عبارت دیگر:

R1 ∞ R2 = R2 ∞ R1

شرط شرط

(R1 join R2) join R3 ≠ R1 join (R2 join R3)
شرط ۱ شرط ۲ شرط ۱ شرط ۲

۲-۲-۳-۳) پیوند طبیعی[1]

پیوند طبیعی مانند پیوند شرطی است با این تفاوت که در آن هیچ شرطی ذکر نمی‌شود و شرط آن بطور پیش فرض، تساوی ویژگیهای مشترک بین دو جدول است. همچنین، در پیوند طبیعی ستونهای تکراری حذف می‌شوند. برای پیوند شرطی از نماد ∞ یا join استفاده می‌شود.

مثال ۱: R1 join R2 را بدست آورید.

مرحلهٔ ۱: بدست آوردن حاصلضرب دو جدول(این مرحله را در مثالهای قبل انجام داده‌ایم)

مرحلهٔ ۲: انتخاب تاپلهایی که در آنها R1.D=R2.D است:

A	B	C	R1.D	R2.D	E	F
a	b	c	10	10	20	a

مرحلهٔ ۳: حذف ستونهای تکراری:

A	B	C	D	E	F
a	b	c	10	20	a

تذکر: برای جلوگیری از اتلاف وقت می‌توان پیوند طبیعی دو جدول را تنها در یک مرحله بدست آورد. برای این کار کافیست تنها تاپلهایی از دو جدول را در هم ضرب کنیم که مقادیر ستونهای مشترک در آنها برابر است و ستونهای مشترک را تنها یکبار در نتیجه بیاوریم.

1 - Natural Join

مثال ۲: S join SPJ را بدست آورید.

s#	sname	city	p#	j#	qty
S1	تهران مصالح	تهران	P1	J1	12000
S1	تهران مصالح	تهران	P2	J1	20000
S1	تهران مصالح	تهران	P3	J1	3000
S1	تهران مصالح	تهران	P4	J1	8000
S1	تهران مصالح	تهران	P1	J2	10000
S1	تهران مصالح	تهران	P1	J3	9000
S2	یزد مصالح	یزد	P1	J1	2000
S2	یزد مصالح	یزد	P1	J3	5000
S2	یزد مصالح	یزد	P3	J1	8000
S3	البرز	اصفهان	P1	J1	9000
S3	البرز	اصفهان	P2	J3	9000
S3	البرز	اصفهان	P1	J3	4000

نکته: چنانچه بین دو جدول هیچ ویژگی مشترکی وجود نداشته باشد، پیوند طبیعی دو جدول با حاصلضرب دو جدول معادل خواهد بود. چنانچه کلیهٔ ویژگیهای دو جدول یکسان باشند، پیوند طبیعی دو جدول با اشتراک دو جدول معادل خواهد بود.

عملگر پیوند طبیعی دارای خاصیت جابجایی و شرکت پذیری است. به عبارت دیگر:

R1 join R2 = R2 join R1

(R1 join R2) join R3 = R1 join (R2 join R3)

۳-۲-۳-۳) شبه پیوند[1]:

این عملگر دقیقاً مانند پیوند طبیعی است با این تفاوت که در نهایت تنها ستونهای جدول اول انتخاب می‌شود. برای شبه پیوند دو جدول R1 و R2 از یکی از نمادهای ذیل استفاده می‌شود:

R1 ⋉ R2
R1 SJ R2
R1 Semi-join R2

مثال ۱: R1 SJ R2 را بدست آورید.

مرحله ۱: بدست آوردن R1 × R2

مرحله 2: انتخاب تاپلهایی از R1 × R2 که در آنها ویژگیهای مشترک دو جدول دارای مقادیر مساوی هستند(تاپلهایی که در آنها R1.D= R2.D است):

A	B	C	R1.D	R2.D	E	F
a	b	c	10	10	20	a

مرحله ۳: انتخاب ستونهای مربوط به جدول R1:

A	B	C	D
a	b	c	10

مثال ۲: S SJ SPJ را بدست آورید.

s#	sname	city
S1	تهران مصالح	تهران
S2	یزد مصالح	یزد
S3	البرز	اصفهان

عملگر شبه پیوند دارای خاصیت جابجایی و شرکت پذیری نیست. به عبارت دیگر:

R1 SJ R2 ≠ R2 SJ R1
(R1 SJ R2) SJ R3 ≠ R1 SJ (R2 SJ R3)

1 - Semi-Join

٤-۲-۳-۳) عملگرهای پیوند خارجی[1]

عملگرهای پیوند خارجی به سه دسته تقسیم می‌شوند:

۱- عملگر پیوند خارجی چپ[2]

۲- عملگر پیوند خارجی راست[3]

۳- عملگر پیوند خارجی کامل[4]

۱-٤-۲-۳-۳) پیوند خارجی چپ:

پیوند خارجی چپ نوعی پیوند طبیعی است که در آن علاوه بر تاپلهای پیوند شدنی از دو جدول، تاپلهای پیوند نشدنی از جدول چپ نیز با مقادیر Null پیوند شده و در نتیجه وارد می‌شوند. نماد پیوند خارجی چپ L-O-JOIN است.

مثال ۱: R1 L-O-JOIN R2 را بدست آورید.

در محاسبهٔ نتیجهٔ این عملیات تاپلهای (p, q, r,20) و (l, m, n,30) و (a,b,e,30) از جدول R1 نیز که مقدار ویژگی D آنها با هیچ یک از مقادیر ویژگی D در جدول R2 برابر نیست، در نتیجه ظاهر می‌شوند:

A	B	C	D	E	F
a	b	c	10	20	a
p	q	r	20	Null	Null
l	m	n	30	Null	Null
a	b	e	30	Null	Null

1 - Outer-join
2 - Left Outer Join
3 - Right Outer Join
4 - Full Outer Join

مثال ۲: S L-O-JOIN SPJ را بدست آورید.

s#	sname	city	p#	j#	qty
S1	تهران مصالح	تهران	P1	J1	12000
S1	تهران مصالح	تهران	P2	J1	20000
S1	تهران مصالح	تهران	P3	J1	3000
S1	تهران مصالح	تهران	P4	J1	8000
S1	تهران مصالح	تهران	P1	J2	10000
S1	تهران مصالح	تهران	P1	J3	9000
S2	یزد مصالح	یزد	P1	J1	2000
S2	یزد مصالح	یزد	P1	J3	5000
S2	یزد مصالح	یزد	P3	J1	8000
S3	البرز	اصفهان	P1	J1	9000
S3	البرز	اصفهان	P2	J3	9000
S3	البرز	اصفهان	P1	J3	4000
S4	ایران مصالح	تهران	Null	Null	Null

۲-٤-۲-۳-۳) پیوند خارجی راست

پیوند خارجی راست نوعی پیوند طبیعی است که در آن علاوه بر تاپلهای پیوند شدنی از دو جدول، تاپلهای پیوند نشدنی از جدول راست نیز با مقادیر Null پیوند شده در نتیجه وارد می‌شوند. نماد پیوند خارجی راست R-O-JOIN است.

مثال: R1 R-O-JOIN R2 را بدست آورید.

در محاسبهٔ نتیجهٔ این عملیات تاپل (15,60,B) از جدول سمت راست یعنی R2 نیز که مقدار ویژگی D آن با هیچ یک از مقادیر ویژگی D در جدول R1 برابر نیست، در نتیجه ظاهر می‌شود:

A	B	C	D	E	F
a	b	c	10	20	a
Null	Null	Null	15	60	b

۳-٤-۲-۳-۳) پیوند خارجی کامل

پیوند خارجی کامل نوعی پیوند طبیعی است که در آن علاوه بر تاپلهای پیوند شدنی از دو جدول، تاپلهای پیوند نشدنی از هر دو جدول نیز با مقادیر Null پیوند شده در نتیجه وارد می‌شوند. نماد پیوند خارجی راست F-O-JOIN است.

مثال: R1 F-O-JOIN R2 را بدست آورید.

A	B	C	D	E	F
a	b	c	10	20	a
p	q	r	20	Null	Null
l	m	n	30	Null	Null
a	b	e	30	Null	Null
Null	Null	Null	15	60	b

٤-۳)عملگر تقسیم[1]

تنها در صورتی می‌توان جدول T1 را بر جدول T2 تقسیم کرد که مجموعهٔ عنوان جدول T2 زیر مجموعه‌ای از مجموعهٔ عنوان جدول T1 باشد. به عبارت دیگر:

$$H(T2) \subset H(T1)$$

مجموعهٔ عنوان T1 ÷ T2، شامل ویژگیهایی از جدول T1 خواهد بود که در جدول T2 وجود ندارند:

$$H(T1 \div T2) = H(T1) - H(T2)$$

از طرف دیگر، تنها مقادیری در بدنهٔ نتیجه وارد می‌شوند که به ازاء همهٔ تاپلهای T2 در جدول T1 وجود داشته باشند.

---

1 - Division

برای تقسیم، از نماد ÷ یا divideBy استفاده می‌شود.

مثال ۱: جداول T1 و T2 را در نظر بگیرید:

T1

A	B	C	D
a	b	c	10
l	m	c	10
k	k	x	20
a	b	g	20
a	b	p	40

T2

C	D
c	10
g	20

از آنجا که $H(T2) \subset H(T1)$ است می‌توان جدول T1 را بر جدول T2 تقسیم کرد.

H(T1 ÷ T2) = H(T1) – H(T2)={A,B,C,D}-{C,D}={A,B}

از طرف دیگر، تنها (A,B)-هایی در جدول نتیجه وارد می‌شوند که هم به ازاء (C,D)=(c,10) وجود داشته باشند و هم به ازاء (C,D)=(g,20).در جدول T1 تنها (A,B)=(a,b)است که هم به ازاء (C,D)=(c,10) وجود دارد و هم به ازاء(C,D)=(g,20). پس نتیجه تنها شامل تاپل (a,b) خواهد بود:

A	B
a	b

مثال ۲: نتیجهٔ SPJ[s#,p#]÷P[p#] را بدست آورید.

SPJ[s#,p#]

s#	p#
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P3
S3	P1
S3	P2

P[p#]

p#
P1
P2
P3
P4

SPJ[s#,p#]÷P[p#]

s#
S1

نکته: با استفاده از عملیات فوق کد تولیدکنندگانی را بدست آوردیم که در جدول SPJ به ازاء کد کلیهٔ محصولات وجود دارند و یا به عبارت ساده‌تر همهٔ محصولات را فروخته‌اند.

۵-۳) عملگر جایگزینی

با استفاده از عملگر جایگزینی می‌توان نتیجهٔ یک عملیات را در یک جدول ذخیره کرد و در عملیات بعدی از آن استفاده کرد. برای عملگر جایگزینی از نماد ← یا giving استفاده می‌شود.

مثال: اسامی تولید کنندگانی را که در پروژهٔ J1 شرکت کرده‌اند بدست آورید.

روش ۱:

```
select SPJ  where  j#='j1'  giving temp1
temp1  join   S giving temp2
project  temp2   over  sname
```

روش ۲:

$$temp1 \leftarrow \acute{o}_{J\#='J1'}(SPJ)$$

$$temp2 \leftarrow temp1 \infty S$$

$$\prod_{sname}(temp2)$$

روش ۳: تنها در یک مرحله

```
((select SPJ  where  j#='j1' ) join S)[sname]
```

و یا:

$$\prod_{Sname}((\sigma_{J\#='J1'}(SPJ)) \infty S)$$

نتیجه:

Sname
تهران مصالح
یزد مصالح
البرز

نکته: در جبر رابطه‌ای نیز مانند جبر اعداد، ترتیب انجام عملیات از داخلی‌ترین پرانتز به خارجی‌ترین پرانتز است.

تمرینهای حل شده:

تمرین ۱: جداول ذیل را در نظر بگیرید:

R

A	B	C	D	E
a	b	c	f	g
a	b	c	h	k
a	b	c	d	k
e	f	i	d	k
j	l	m	h	n
n	o	p	q	r
j	l	i	d	n

S

C	D	E
i	d	g
m	h	k

مطلوب است محاسبهٔ جداول ذیل:

a) R ÷ S

A	B

b) R - (R join S)

R join S

A	B	C	D	E

R - (R join S)

A	B	C	D	E
a	b	c	f	g
a	b	c	h	k
a	b	c	d	k
e	f	i	d	k
j	l	m	h	n
n	o	p	q	r
j	l	i	d	n

c) R ÷ S[C,D]

S[C,D]

C	D
i	d
m	h

R÷S[C,D]

A	B	E
j	l	n

d) R - (R join S[C,D])

S[C,D]

C	D
i	d
m	h

R join S[C,D]

A	B	C	D	E
e	f	i	d	k
j	l	m	h	n
j	l	i	d	n

R - (R join S[C,D])

A	B	C	D	E
a	b	c	f	g
a	b	c	h	k
a	b	c	d	k
n	o	p	q	r

e) (R where D='h')[D,E] intersect (S where E='k')[D,E]

(R where D='h')[D,E]

D	E
h	k
h	n

(S where E='k')[D,E]

D	E
h	k

e

D	E
h	k

f) R [D,E] - S[D,E]

R[D,E]

D	E
f	g
h	k
d	k
h	n
q	r
d	n

S[D,E]

D	E
d	g
h	k

R[D,E]-S[D,E]

D	E
f	g
d	k
h	n
q	r
d	n

تمرین ۲: جداول S و P و J و SPJ را در نظر گرفته، عملیات لازم برای بدست آوردن نتایج ذیل را در جبر رابطه‌ای نوشته، نتیجهٔ حاصل را بدست آورید.

Q1: لیستی از نام تولید کنندگان و نام محصولات تهیه کنید بطوریکه تولید کنندهٔ مورد نظر محصول مورد نظر را فروخته باشد.

(S **join** SPJ **join** P)[sname,pname]

sname	pname
تهران مصالح	تیر آهن
تهران مصالح	آرماتور
تهران مصالح	سیمان
تهران مصالح	آلومنیم
یزد مصالح	تیر آهن
یزد مصالح	سیمان
البرز	تیر آهن
البرز	آرماتور

Q2: نام پروژه‌هایی را بیابید که در آنها تیرآهن مصرف شده است.

((P where pname='تیرآهن') **join** SPJ **join** J)[jname]

jname
مرمت آثار باستانی
فرودگاه

Q3: کد تولیدکنندگانی را بیابید که محصولات مشکی‌رنگ فروخته‌اند.

(SPJ **join** (p where color='مشکی'))[s#]

s#
S1
S2
S3

Q4: نام تولیدکنندگانی را بیابید که محصولات مشکی‌رنگ فروخته‌اند.

(SPJ **join** (p where color='مشکی') **join** S)[sname]

sname
تهران مصالح
یزد مصالح
البرز

Q5: کد تولیدکنندگانی را بیابید که کلیهٔ محصولات مشکی رنگ را فروخته‌اند.

SPJ[s#,p#] **divideby** (p where color='مشکی')[p#]

s#
S1
S3

Q6: نام تولیدکنندگانی را بیابید که کلیهٔ محصولات مشکی رنگ را فروخته‌اند.

((SPJ[s#,p#] **divideby** (p where color='مشکی')[p#]) **join** S)[sname]

sname
تهران مصالح
البرز

Q7: نام تولیدکنندگانی را بیابید که هیچ محصول مشکی رنگی را نفروخته‌اند.

S[sname] - (SPJ **join** (p where color='مشکی') **join** S)[sname]

sname
ایران مصالح

Q8: نام تولید کنندگانی را بیابید که در شهر اصفهان هستند و در پروژه‌های اصفهان مشارکت داشته‌اند.

((S where city='اصفهان') **join** SPJ **join** (J where city='اصفهان')) [sname]

sname

Q9: نام تولید کنندگانی را بیابید که در شهر اصفهان هستند ولی در پروژه‌های اصفهان مشارکت نداشته‌اند.

(S where city='اصفهان')[sname]

-

((S where city='اصفهان') **join** SPJ **join** (J where city='اصفهان')) [sname]

sname
البرز

Q10: کد تولیدکنندگانی را بیابید که برای پروژه‌های اصفهان سیمان فروخته‌اند.

(SPJ **join** (J where city='اصفهان') **join** (P where pname='سیمان'))[s#]

s#

Q11: کد تولید کنندگانی را بیابید که برای پروژهٔ مرمت آثار باستانی اصفهان سیمان فروخته‌اند.

((J where jname='مرمت آثار باستانی' and city='اصفهان')

join
SPJ
join
(P where pname='سیمان'))[s#]

s#

Q12: نام تولید کنندگانی را بیابید که برای پروژهٔ مرمت آثار باستانی اصفهان سیمان فروخته‌اند.

((J where jname='مرمت آثار باستانی' and city='اصفهان')

join
SPJ
join
(P where pname='سیمان')

join
S)[sname]

sname

Q13: کد تولید کنندگانی را بیابید که برای پروژه‌های اصفهان سیمان نفروخته‌اند.

توضیح: کد تولید کنندگانی را که برای پروژه‌های اصفهان سیمان فروخته‌اند از کد همهٔ تولید کنندگان کسر می‌کنیم:

S[s#]
-
(SPJ **join** (J where city='اصفهان') **join** (P where pname='سیمان'))[s#]

s#
S1
S2
S3
S4

Q14: کد تولید کنندگانی را بیابید که در همهٔ پروژه‌های اصفهان مشارکت داشته‌اند.

SPJ[s#,j#]
divideby
(J where city='اصفهان')[j#]

s#

Q15: نام تولید کنندگانی را بیابید که در همهٔ پروژه‌های اصفهان مشارکت داشته‌اند.

((SPJ[s#,j#] **divideby** (J where city='اصفهان')[j#])

join
S)[sname]

sname

Q16: نام تولید کنندگانی را بیابید که در همهٔ پروژه‌های اصفهان مشارکت داشته‌اند و در شهر اصفهان هستند.

((SPJ[s#,j#] **divideby** (J where city='اصفهان')[j#])

join

(S where city='اصفهان'))[sname]

sname

Q17: تعداد دفعات فروش تولید کنندهٔ S1 برای پروژهٔ J1 را بدست آورید.

((**summarize** (SPJ where s#='S1' and j#='J1') by(s#) add count(qty) as count1)[count1]

count1
4

تذکر: به جای count(qty) می‌توان از count(s#) یا count(p#) یا count(j#) استفاده کرد چون هدف شمارش تاپلهاست و اینکه شمارش براساس کدام فیلد صورت گیرد مهم نیست.

تمرینهای فصل:

تمرین ۱: جداول زیر را در نظر بگیرید:

R

A	B	C	D	E
a	b	c	f	g
a	b	c	h	k
a	b	c	d	k
e	f	I	d	k
j	l	m	h	n
n	o	p	q	r

S

C	D	E
c	f	g
c	h	k

T

A	B	C	H
a	b	c	h

مطلوب است محاسبهٔ جداول ذیل:

a) $R \times S$
b) $R \div S[D,E]$
c) R-(R join S)
d) $R \div T[A,B,C]$
e) (R join S)[A,C] join (R join T)[C,H]
f) (R where D='h')[A,B,C] intersect (T where H='h')[A,B,C]
g) (R where A='b') union (R where D='d')
h) $R \div S$

تمرین ۲: پایگاه داده‌ای شامل سه جدول ذیل را در نظر بگیرید:

st (st# , sname)

در این جدول اطلاعات دانشجویان ذخیره می‌شود:

st#: شماره دانشجویی

sname: نام دانشجو

course(crs# , cname , unit)

در این جدول اطلاعات درسها ذخیره می‌شود:

crs#: شمارهٔ درس

cname: نام درس

unit: تعداد واحد درس

grades (st# , crs# , term , grade)

در این جدول اطلاعات نمرات ذخیره می‌شود:

st#: شماره دانشجویی

crs#: شمارهٔ درس

term: نیمسال اخذ درس توسط دانشجو(مثلاً ۸۳۱ به معنای نیمسال اول سال ۸۳ است)

grade: نمرهٔ اخذ شده

st

st#	sname
8001	آرش راد
8002	عسل شاملو
8003	سیاوش آزاد
8101	ساغر راد
8102	علی نیکی
8111	فرامرز نیکی
8112	علی رضایی

course

crs#	cname	unit
1100	ادبیات	2
1105	تربیت بدنی	1
1400	برنامه سازی ۱	3
1401	برنامه سازی ۲	3
1403	ذخیره و بازیابی	3
1407	پایگاه داده‌ها	3
1500	ریاضی ۱	3
1600	زبان پیش نیاز	2

grades

st#	crs#	term	grade
8001	1400	811	9
8001	1400	812	13
8001	1401	811	13
8101	1400	821	19
8101	1500	821	14
8111	1400	811	12
8111	1401	811	20

۱-۲) جداول ذیل را بدست آورید:

a)((st where st#=8001) join grades join course)[cname]
b)((grades where crs#=1401 and term=811) join st)[sname]
c)(st join (grades where term=811) join course)[sname,cname]
d) summarize grades by(st#) add sum(grade) as sumGrade
e) summarize (grades join course) by(st#) add sum (grade*unit) /sum (unit) as average
f) (st join (summarize (grades join course) by(st#) add sum(grade*unit)/sum(unit) as average))[sname,average]
g) summarize ((grades where term=811) join course) by(st#) add sum(grade*unit)/sum(unit) as average
h) (st L-O-Join (summarize ((grades where term=811) join course) by(st#) add sum(grade*unit)/sum(unit) as average))[sname,average]
i) extend (grades join course) add(grade*unit) as SumGrades
j) grades[st#,crs#] divideby course[crs#]

۲-۲)عملیات لازم در جبر رابطه‌ای برای بدست آوردن نتایج ذیل را بنویسید:

Q18: نام دروسی را بیابید که دانشجوی ۸۱۰۱ در ترم ۸۱۱ گرفته است.

Q19: نام دروسی را بیابید که دانشجوی ۸۱۰۱ در ترم ۸۱۱ نگرفته است.

Q20: نام دروسی را بیابید که دانشجوی ۸۱۰۱ در ترم ۸۱۱ در آنها نمرهٔ قبولی گرفته است.

Q21: نام دروسی را بیابید که دانشجوی ۸۱۰۱ در ترم ۸۱۱ گرفته است ولی در آنها نمرهٔ قبولی نگرفته است.

Q22: نام دروسی را بیابید که تا کنون هیچ دانشجویی آنها را نگرفته است.

Q23: شمارهٔ دانشجویانی را بیابید که همهٔ دروس را گذرانده‌اند.

Q24: نام دانشجویانی را بیابید که همهٔ دروس را گذرانده‌اند.

Q25: شمارهٔ دانشجویانی را بیابید که همهٔ دروس را نگذرانده‌اند.

Q26: شمارهٔ دانشجویانی را بیابید که هیچ درسی را نگذرانده‌اند.

Q27: معدل کل دانشجوی شمارهٔ ۸۱۱۱ را بدست آورید.

Q28: معدل دانشجوی شمارهٔ ۸۱۱۱ در ترم ۸۱۱ را بدست آورید.

Q29: لیستی از شماره دانشجویی و معدل کل کلیهٔ دانشجویان تهیه کنید.

Q30: لیستی از شماره دانشجویی و معدل کل کلیهٔ دانشجویان در ترم ۸۱۱ تهیه کنید.

Q31: لیستی از شماره دانشجویی و معدل کل کلیهٔ دانشجویان در ترم ۸۱۱ تهیه کنید بطوریکه شماره دانشجویی دانشجویانی که در ترم ۸۱۱ هیچ درسی نداشته‌اند هم در این لیست موجود باشد.

Q32: لیستی از نام و معدل کل کلیهٔ دانشجویان در ترم ۸۱۱ تهیه کنید.

Q33: لیستی از شمارهٔ کلیهٔ دانشجویان و تعداد کل واحدهای گذرانده شده توسط هر یک از آنها تهیه کنید.

Q34: لیستی از نام کلیهٔ دانشجویان و تعداد کل واحدهای گذرانده شده توسط هر یک از آنها تهیه کنید.

Q35: لیستی از نام کلیهٔ دانشجویان و تعداد کل واحدهای گذرانده شده توسط هر یک از آنها در ترم ۸۱۱ تهیه کنید.

Q36: لیستی از شمارهٔ کلیهٔ دروس و تعداد کل دانشجویانی که آنها را گذرانده‌اند تهیه کنید.

Q37: نام دانشجویانی را بیابید که در ترم ۸۳۱ درس ادبیات را گرفته‌اند.

Q38: تعداد دانشجویانی را بیابید که در ترم ۸۳۱ درس ادبیات را گرفته‌اند.

۴

زبان پرس و جوی ساخت یافته[1] (SQL)

همانطور که قبلاً نیز به این مسئله اشاره شد، هر کاربر یا برنامهٔ کاربردی برای ارتباط با پایگاه داده‌ها از یک زبان فرعی داده‌ای استفاده می‌کند. SQL معروفترین زبان فرعی داده‌ای است که توسط کلیهٔ DBMS-های کنونی دنیا شامل SQL SERVER، Oracle، DB2، Informix و ... پشتیبانی می‌شود. به عبارت دیگر برای نوشتن اغلب برنامه‌های کاربردی اعم از برنامه‌های حسابداری، حقوق و دستمزد، انبارداری، سایتهای وب که با پایگاه داده‌ها سر و کار دارند و ... فارغ از زبان برنامه‌نویسی و همچنین DBMS مورد استفاده، استفاده از زبان SQL الزامی است. دستورات SQL همانند هر زبان فرعی داده‌ای دیگر به سه گروه تقسیم می‌شوند:

۱- دستورات تعریف داده[2] یا DDL

۲- دستورات دستکاری داده‌ها[3] یا DML

۳- دستورات کنترل داده‌ها[4] یا DCL

---

1 - Structured Query Language
2 - Data Definition Language
3 - Data Manipulation Language
4 - Data Control Language

در این فصل، کاربرد و ساختار دستورات SQL را تشریح می‌کنیم. در اغلب مثالهای این فصل از جداول S و P و J و SPJ که در فصل ۳ ارائه شد استفاده شده است.

تذکر: کلیهٔ دستورات SQL یک پرس و جو[1] محسوب می‌شوند ولی برخی کارشناسان واژهٔ پرس و جو را تنها در مورد دستور select به کار می‌برند.

تذکر: برای برخی از پرس و جوهای این فصل چند روش ارائه شده است ولی این به معنای ارائهٔ کلیهٔ روشهای ممکن نیست.

۱-٤) دستورات تعریف داده‌ها

از دستورات تعریف داده‌ها برای تعیین و یا تغییر ساختار داده‌ها استفاده می‌شود. این دستورات عبارتند از:

۱-۱-٤) دستور ایجاد پایگاه داده‌ها

قبل از ایجاد جداول لازم است یک پایگاه داده‌ها ایجاد شود و سپس جداول مورد نظر در داخل آن ساخته شوند.

قالب کلی این دستور به شکل ذیل است:

create database نام پایگاه داده‌ها

مثال: پایگاه داده‌ای به نام Sale ایجاد کنید.

```
create database Sale
```

۲-۱-٤) دستور ایجاد جدول

قالب کلی این دستور به شکل ذیل است:

```
create table نام جدول ( نام ویژگی ۱  نوع ویژگی ۱  [not null][unique],
                        نام ویژگی ۲  نوع ویژگی ۲  [not null][unique],
                        نام ویژگی ۳  نوع ویژگی ۳  [not null][unique],
                        ...,
                        primary key(نام ویژگیهای تشکیل دهندهٔ کلید اصلی),
```

1 - Query

,نام جدول مورد نظر references (نام ویژگیهای کلید خارجی ۱)foreign key
,نام جدول مورد نظر references (نام ویژگیهای کلید خارجی ۲)foreign key
...
([شرط مورد نظر check]

تذکر: در کلیهٔ دستورات برای مشخص کردن قسمتهای اختیاری از کروشه استفاده شده است.

معروفترین انواع داده در SQL عبارتند از:

نوع داده‌ای	بازه
integer	اعداد صحیح
smallint	اعداد صحیح
decimal(p,q)	اعدادی با p رقم و q رقم اعشاری در سمت راست
Float	اعداد اعشاری با ممیز شناور
numeric(p,q)	اعداد حقیقی
char(n)	رشته‌های کاراکتری با طول n
varchar(n)	رشته‌های کاراکتری با طول متغیر کوچکتر یا مساوی n
date	تاریخ با فرمت yyyymmdd
Time	زمان با فرمت hhmmss

مثال ۱: جدول S را ایجاد کنید بطوریکه کاربر مجاز نباشد هیچ تاپلی با sname خالی یا تکراری در این جدول درج کند:

```
create table S(s# char(2),
        sname nchar(30) not null unique,
        city nchar(20),
        primary key(s#))
```

تذکر: برای عبارات فارسی از انواع nchar و ntext و ... استفاده می‌شود.

تذکر: در صورت استفاده از عبارت not null برای یک ویژگی، DBMS از درج تاپلهایی که در آنها برای ویژگی مورد نظر مقداری وارد نشده باشد جلوگیری می‌کند. در صورت استفاده از عبارت unique برای یک ویژگی، DBMS از درج تاپلهایی که در آنها برای ویژگی مورد نظر مقدار تکراری وارد شده باشد جلوگیری می‌کند.

مثال ۲: جدول SPJ را ایجاد کنید بطوریکه بازهٔ مجاز برای qty اعداد بین ۵۰۰ تا ۲۰۰۰۰ باشد.

برای تعیین قوانین جامعیت داده‌ای از قسمت check استفاده می‌شود:

```
create table SPJ(s# char(2),
                 p# char(2),
                 j# char(2),
                 qty integer,
                 primary key(s#,p#,j#),
                 foreign key(s#) references S,
                 foreign key(p#) references P,
                 foreign key(j#) references J
                 check(qty>500 and qty<20000))
```

۳-۱-٤) ایجاد ایندکس[1]

ایندکسها در واقع همان کلیدهای ثانویه هستند. از ایندکسها برای تسریع جستجوی یک موجودیت خاص در صورت عدم دسترسی به مقدار کلید اصلی استفاده می‌شود.

قالب کلی این دستور به شکل ذیل است:

create [unique] index نام ایندکس on نام جدول (نام ویژگیها)

مثال: روی جدول S یک ایندکس بر اساس نام تولید کننده و با نام names ایجاد کنید.

```
create index names on S(sname)
```

تذکر: در صورت استفاده از عبارت unique در این دستور، DBMS از ورود اسمهای تکراری در جدول S جلوگیری خواهد کرد.

---

1 - Index

٤-١-٤) اضافه کردن یک ستون جدید به یک جدول

قالب کلی این دستور به شکل ذیل است:

alter table نام جدول add(نام ستون جدید مشخصات ستون جدید)

مثال: در جدول S ستون جدیدی با نام tel برای درج شماره تلفن تولید کنندگان اضافه کنید.

```
alter table S add(tel char(10))
```

٤-١-٥) تغییر مشخصات یک ستون از یک جدول

قالب کلی این دستور به شکل ذیل است:

alter table نام جدول modify(نام ستون مشخصات جدید)

مثال: در جدول S، طول ستون Sname را از ۳۰ کاراکتر به ۲۰ کاراکتر تغییر دهید.

```
alter table S modify(sname nchar(20) not null unique)
```

٤-١-٦) حذف یک جدول

قالب کلی این دستور به شکل ذیل است:

drop table نام جدول

مثال: جدول S را حذف کنید.

```
drop table S
```

٤-١-٧) حذف یک ایندکس

قالب کلی این دستور به شکل ذیل است:

drop index نام ایندکس.نام جدول

مثال: ایندکس names روی جدول S را حذف کنید.

```
drop index S.names
```

۲-٤) دستورات دستکاری داده‌ها

از این دستورات برای تهیهٔ گزارشات(خواندن تاپلها)، درج تاپلها، حذف تاپلها و تغییر تاپلها استفاده می‌شود. این دستورات عبارتند از:

۱-۲-٤) دستور انتخاب

این دستور یکی از پرکاربردترین دستورات SQL است. از این دستور برای انتخاب تاپلها و ستونهای مورد نظر از یک یا چند جدول استفاده می‌شود. قالب کلی این دستور به شکل ذیل است:

select [distinct] نام ستونها و یا عبارات محاسباتی مورد نظر

from نام جداول

[where شرط روی تاپلها]

[group by نام ستونهایی که قرار است لیست بر اساس آنها گروه بندی شود]

[having شرط روی گروهها]

[order by نام ستونهایی که قرار است لیست بر اساس آنها مرتب شود]

برای درک قسمتهای مختلف این دستور به مثالهای ذیل توجه کنید:

مثال: نام کلیهٔ شهرهایی را بیابید که تولید کننده‌ای در آنها قرار دارد.

برای این کار کافی است ستون city از جدول S را انتخاب کنیم:

```
select city
from S
```

و یا می‌توان نام جدول را نیز ذکر کرد:

```
select S.city
from S
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

city
تهران
یزد
اصفهان
تهران

مشاهده می‌کنید که در این جدول تاپلهای تکراری وجود دارند. برای حذف تاپلهای تکراری کافیست از عبارت distinct استفاده کنیم:

```
select  distinct city
from S
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

city
تهران
یزد
اصفهان

مثال: مشخصات کلیهٔ تولید کنندگان را بیابید.

برای این کار کافیست تمام ستونهای جدول S را انتخاب کنیم:

```
select s#,sname,city
from S
```

به جای ذکر نام تمام ستونهای یک جدول می‌توان از * استفاده کرد (* به معنی تمام ستونهای جدول است):

```
select *
from S
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

s#	sname	city
S1	تهران مصالح	تهران
S2	یزد مصالح	یزد
S3	البرز	اصفهان
S4	ایران مصالح	تهران

قسمت where: از قسمت where برای انتخاب تاپلهایی از جدول که شرط بخصوصی دارند استفاده می‌شود.

مثال: نام تولید کنندگان تهرانی را بیابید.

```
select sname
```

```
from S
where city='تهران'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

Sname
تهران مصالح
ایران مصالح

عملگرهای in و not in: از عملگرهای in و not in برای تست وجود یا عدم وجود یک مقدار در داخل یک مجموعه استفاده می‌شود.

مثال: مشخصات فروشهای مربوط به محصولات 'P2' و 'P3' و 'P4' را بیابید.

روش ۱:

```
select *
from SPJ
where p#='P2' or p#='P3' or p#='P4'
```

روش ۲: استفاده از عملگر in

```
select *
from SPJ
where p# in ('P2','P3','P4')
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

s#	p#	j#	qty
S1	P2	J1	20000
S1	P3	J1	3000
S1	P4	J1	8000
S2	P3	J1	8000
S3	P2	J3	9000

مثال: مشخصات فروشهایی را بیابید که مربوط به محصولات 'P2' یا 'P3' یا 'P4' نیستند.

روش ۱:

```
select *
from SPJ
```

```
where p#<>'P2' and p#<>'P3' and p#<>'P4'
```

روش ۲: استفاده از عملگر not in

```
select *
from SPJ
where p# not in ('P2','P3','P4')
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

s#	p#	j#	qty
S1	P1	J1	12000
S1	P1	J2	10000
S1	P1	J3	9000
S2	P1	J1	2000
S2	P1	J3	5000
S3	P1	J1	9000
S3	P1	J3	4000

عملگر like: از عملگر like برای جستجوی یک عبارت در داخل مقادیر یک ستون رشته‌ای استفاده می‌شود.

مثال: مشخصات تولید کنندگانی را بیابید که درنام آنها عبارت "مصالح" بکار رفته باشد.

برای این پرس و جو لازم است از عملگر like استفاده کنیم.

```
select *
from S
where sname like '%مصالح%'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

s#	sname	city
S1	تهران مصالح	تهران
S2	یزد مصالح	یزد
S4	ایران مصالح	تهران

تذکر: در این پرس و جو چون محل قرار گرفتن عبارت "مصالح" مهم نبود، در دو طرف عبارت از ٪ استفاده کردیم.

چنانچه نام تولید کنندگانی که نام آنها با عبارت "مصالح" شروع می‌شود مورد نظر باشد، از دستور ذیل:

```
select *
from S
where sname like '% مصالح'
```

و چنانچه نام تولید کنندگانی که نام آنها به عبارت "مصالح" ختم می‌شود مورد نظر باشد، از دستور ذیل:

```
select *
from S
where sname like '%مصالح'
```

استفاده می کنیم.

تابع sum: از تابع sum برای محاسبهٔ مجموع مقادیر یک ستون استفاده می‌شود.

مثال: میزان کل فروش 'P1' را بیابید.

برای این کار کافیست مجموع مقادیر ستون qty را در فروشهای مربوط به 'P1' محاسبه کنیم:

```
select sum(qty)
from SPJ
where p#='P1'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

51000

تابع avg: از تابع avg برای محاسبهٔ میانگین مقادیر یک ستون استفاده می‌شود.

مثال: میانگین فروش 'P1' را بیابید.

برای این کار کافیست میانگین مقادیر ستون qty را در فروشهای مربوط به 'P1' محاسبه کنیم:

```
select avg(qty)
from SPJ
where p#='P1'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

7285.71

تابع max: از تابع max برای به دست آوردن بزرگترین مقدار یک ستون استفاده می‌شود.

مثال: حداکثر فروش 'P1' تا این لحظه را به دست آورید.

برای این کار کافیست ماکزیمم مقادیر ستون qty را در فروشهای مربوط به 'P1' محاسبه کنیم:

```
select max(qty)
from SPJ
where p#='P1'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

12000

تابع min: از تابع min برای به دست آوردن کوچکترین مقدار یک ستون استفاده می‌شود.

مثال: کمترین میزان فروش 'P1' توسط 'S1' را به دست آورید.

برای این کار کافیست مینیمم مقادیر ستون qty را در فروشهای مربوط به 'S1' و 'P1' محاسبه کنیم:

```
select min(qty)
from SPJ
where p#='P1' and s#='S1'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

9000

تابع count: از تابع count برای محاسبهٔ تعداد تاپلهای مورد نظر از یک جدول استفاده می‌شود.

مثال ۱: محصول 'P1' تا کنون چند بار فروخته شده است؟

برای این کار کافیست تعداد تاپلهایی از جدول SPJ که مربوط به 'P1' هستند را به دست آوریم:

```
select count(*)
from SPJ
where p#='P1'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

7

مثال ۲: محصول 'P1' تا کنون توسط چند تولید کننده فروخته شده است؟

برای این کار کافیست تعداد تولید کنندگان غیر تکراری از جدول SPJ که 'P1' را فروخته‌اند به دست آوریم:

```
select count(distinct s#)
from SPJ
where p#='P1'
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

3

قسمت group by: از قسمت group by برای گروه بندی لیست بر اساس مقادیر یک یا چند ستون استفاده می‌شود.

مثال ۱: لیستی از کد محصولات و میزان کل فروش هر یک از آنها تهیه کنید.

در اینجا sum(qty) برای هر یک از محصولات P1 و P2 و P3 و... باید بطور جداگانه محاسبه شود پس لازم است لیست را بر اساس p# گروه بندی کنیم:

```
select p#,sum(qty)
from SPJ
group by p#
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

p#	
P1	51000
P2	29000
P3	11000
P4	8000

تذکر: در صورت استفاده از قسمت group by ، قسمت select باید شامل نام ویژگیهایی که لیست بر اساس آنها گروه بندی شده است و در صورت نیاز یکی از توابع sum، avg ، max، min، count باشد. مثلاً در پرس و جوی بالا چون در قسمت group by از p# استفاده شده است، قسمت select حتماً باید شامل p# و در صورت لزوم، یکی از توابع sum، avg، max، min، count باشد و نمی‌توان از هیچ ویژگی دیگری در این قسمت استفاده کرد.

مثال ۲: لیستی از کد محصولات، کد پروژه‌ها و میزان کل فروش محصول مورد نظر برای پروژهٔ مورد نظر تهیه کنید.

در اینجا میزان کل فروش برای هر ترکیب شمارهٔ محصول/شمارهٔ پروژه بایستی بطور جداگانه محاسبه شود. پس لازم است لیست را بر اساس p# و j# گروه بندی کنیم:

```
select p#,j#,sum(qty)
from SPJ
group by p#,j#
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

p#	j#	
P1	J1	23000
P2	J1	20000
P3	J1	11000
P4	J1	8000
P1	J2	10000
P1	J3	18000
P2	J3	9000

قسمت having: از قسمت having برای انتخاب گروههایی که شرط بخصوصی دارند استفاده می‌شود.

مثال: کد محصولاتی را بیابید که میزان کل فروش آنها بیش از ۲۰۰۰۰ کیلوگرم است.

در اینجا میزان کل فروش برای هر محصول بایستی بطور جداگانه محاسبه شده، سپس کد محصولاتی که میزان کل فروش آنها بیش از ۲۰۰۰۰ است استخراج شود:

```
select p#
from SPJ
group by p#
having sum(qty)>20000
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

p#
P1
P2

قسمت order by: از قسمت order by برای مرتب کردن لیست بر اساس مقادیر یک یا چند ستون عددی، رشته‌ای و یا تاریخ ازیک جدول به ترتیب صعودی(asc) و یا نزولی(desc) استفاده می‌شود.

مثال ۱: لیستی از مشخصات تولید کنندگان تهیه کنید بطوریکه بر اساس ترتیب صعودی نام آنها مرتب باشد.

```
select *
from S
order by sname asc
```

تذکر: چون ترتیب مرتب سازی بطور پیش فرض صعودی است می‌توان عبارت asc را حذف کرد:

```
select *
from S
order by sname
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

s#	Sname	city
S3	البرز	اصفهان
S4	ایران مصالح	تهران
S1	تهران مصالح	تهران
S2	یزد مصالح	یزد

مثال ۲: لیستی از مشخصات تولید کنندگان تهیه کنید بطوریکه بر اساس ترتیب صعودی شهر و ترتیب نزولی نام مرتب باشد.

```
select *
from S
order by city asc,sname desc
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

s#	Sname	city
S3	البرز	اصفهان
S1	تهران مصالح	تهران
S4	ایران مصالح	تهران
S2	یزد مصالح	یزد

تذکر: در اینجا لیست ابتدا بر اساس ویژگی اول یعنی city مرتب شده است. سپس در مورد تاپلهایی که ویژگی city در آنها مساوی است، ترتیب نزولی بر اساس ویژگی دوم یعنی sname اعمال شده است.

عملگر as: از عملگر as برای تغییر نام یک ستون در لیست استفاده می‌شود.

مثال: لیستی از نام تولید کنندگان و شهر سکونت آنها تهیه کنید.

فرض کنید بخواهیم نام ستون sname را به supplierName تغییر دهیم:

```
select sname as supplierName,city
from S
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

supplierName	city
تهران مصالح	تهران
یزد مصالح	یزد
البرز	اصفهان
ایران مصالح	تهران

پرس و جو روی چند جدول

در کلیهٔ مثالهای قبل اطلاعات مورد نیاز در پرس و جو تنها از یک جدول استخراج شدند ولی در بسیاری از موارد نتیجهٔ پرس و جو را باید از چند جدول استخراج کرد.

مثال: لیستی از نام تولید کنندگان و نام محصولات تهیه کنید بطوریکه تولید کنندهٔ مورد نظر محصول مورد نظر را فروخته باشد.

نام تولید کنندگان در جدول S و نام محصولات در جدول P قرار دارد. همچنین در جدول SPJ مشخص می‌شود که چه تولید کننده‌ای چه محصولی را فروخته است. بنابراین کافیست جداول S و P و SPJ را با هم پیوند دهیم:

```
select distinct sname,pname
from S,SPJ,P
where SPJ.s#=S.s# and SPJ.p#=P.p#
```

sname	pname
تهران مصالح	تیر آهن
تهران مصالح	آرماتور
تهران مصالح	سیمان
تهران مصالح	آلومنیم
یزد مصالح	تیر آهن
یزد مصالح	سیمان
البرز	تیر آهن
البرز	آرماتور

عملگرهای exists و not exists: از عملگرهای exists و not exists برای تست وجود یا عدم وجود تاپلهایی خاص در یک جدول استفاده می‌شود.

مثال: نام تولید کنندگانی را بیابید که فروشی داشته‌اند.

روش ۱: این پرس و جو را می‌توان به این صورت تعبیر کرد: از جدول S نام تولید کنندگانی را پیدا کنید که در جدول فروش(SPJ) فروشی(تاپلی) برای آنها وجود دارد:

```
select sname
from S
where exists (select *
              from SPJ
              where SPJ.s#=S.s#)
```

روش ۲: با استفاده از پیوند دو جدول نیز می‌توان نتیجهٔ این پرس و جو را به دست آورد:

```
select sname
from S,SPJ
where S.s#=SPJ.s#
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

Sname
تهران مصالح
یزد مصالح
البرز

عملگر union: از عملگر union برای به دست آوردن اجتماع دو جدول سازگار استفاده می‌شود.

مثال: نام شهرهایی را بیابید که تولید کننده‌ای در آنها قرار دارد یا پروژه‌ای در آنها در حال اجراست.

نام شهرهایی که تولید کننده‌ای در آنها زندگی می‌کند

U

نام شهرهایی که پروژه‌ای در آنها در حال اجراست

نتیجهٔ مورد نظر=

```
(select city
 from S)
union
(select city
 from J)
```

عملگر except: از عملگر except برای تفریق دو جدول سازگار استفاده می‌شود.

مثال: نام شهرهایی را بیابید که تولید کننده‌ای در آنها زندگی می‌کند ولی پروژه‌ای در آنها در حال اجرا نیست.

نام شهرهایی که تولید کننده‌ای در آنها زندگی می‌کند

-

نام شهرهایی که پروژه‌ای در آنها در حال اجراست

نتیجهٔ مورد نظر=

```
(select city
 from S)
except
(select city
 from J)
```

۲-۲-٤) دستور ایجاد دید خارجی یا دیدگاه[1]

همانگونه که قبلاً نیز به این مسئله اشاره شد، هر کاربر از دید خود به اطلاعات موجود در پایگاه داده‌ها نگاه می‌کند. مثلاً برای یک دانشجو داده‌های زیادی مثل شماره دانشجویی، نام، نام خانوادگی، رشتهٔ تحصیلی، سال ورود به دانشگاه، وضعیت مالی دانشجو، شمارهٔ دفترچهٔ تامین اجتماعی و ... نگهداری می‌شود. مسائل مالی یا وضعیت تامین اجتماعی دانشجو به امور مالی مربوط می‌شود و نه به آموزش، پس هیچ لزومی ندارد که این داده‌ها را در دسترس کاربران قسمت آموزش قرار دهیم. بنابراین بهتر است برای کاربران قسمت آموزش یک یا چند دیدگاه ایجاد کنیم تا بدون درگیر شدن با داده‌هایی که مربوط به حوزهٔ عملیاتی آنها نیست، عملیات مورد نظر خود را انجام دهند. دیدگاه، در واقع یک جدول است که توسط طراح پایگاه داده‌ها طراحی می‌شود. DBMS وظیفه دارد پس از اعمال هر تغییر در داده‌های جداول، محتویات دیدگاههایی را که روی جداول مورد نظر ساخته شده‌اند اصلاح کند. به عنوان مثال فرض کنید درقسمتهایی از سیستم دانشگاه که کاربران قسمت آموزش با آنها سر و کار دارند، گزارشات متعددی وجود داشته باشند که در آنها از معدل دانشجو استفاده شده باشد. محاسبهٔ معدل بر اساس نمرات دانشجو در دروس مختلف و تعداد واحد هر درس انجام می‌شود. از آنجا که این عمل وقتگیر است می‌توان دیدگاهی شامل دو ستون شماره دانشجویی و معدل ایجاد کرد و نحوهٔ محاسبهٔ معدل بر

---

1 - View

اساس نمرات یک دانشجو و تعداد واحد هر درس را در آن مشخص کرد. در این صورت، پس از درج، حذف و اصلاح نمرات یک دانشجو و یا تغییر تعداد واحد هر درس، DBMS بطور اتوماتیک معدل دانشجو یا دانشجویان مورد نظر را در دیدگاه اصلاح می‌کند بنابراین، داده‌های موجود در دیدگاه همواره به روز می‌باشند. به این ترتیب، می‌توان در گزارشاتی که در آنها از معدل دانشجویان استفاده می‌شود بجای محاسبهٔ معدل بر اساس مقادیر چندین جدول، از معدلهای محاسبه شده و آماده در این دیدگاه استفاده کرد. بنابراین، استفاده از دیدگاهها باعث سهولت انجام بسیاری از پرس و جوها و در نتیجه تسریع تهیهٔ بسیاری از گزارشات می‌شود.

قالب کلی دستور ایجاد دیدگاه به شکل ذیل است:

create view نام دیدگاه (نام ستونهای دیدگاه)
as
یک دستور انتخاب

مثال: دیدگاهی به نام PartSale ایجاد کنید که شامل کد محصولات و میزان کل فروش آنها باشد.

```
create view PartSale(p#,sum1)
as
select p#,sum(qty)
from SPJ
group by p#
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

PartSale

p#	sum1
P1	51000
P2	29000
P3	11000
P4	8000

تذکر: نام ستونهای دیدگاه به ترتیب با نام ستونهای انتخاب شده در دستور select نظیر می‌شوند مثلاً در دیدگاه بالا p# با p# و sum1 با sum(qty) نظیر می‌شود.

مثال: نام محصولاتی را بیابید که میزان کل فروش آنها بیشتر از ۲۰۰۰۰ کیلوگرم است.

با فرض وجود دیدگاه PartSale، تهیهٔ این گزارش بسیار ساده خواهد بود:

```
select pname
from PartSale,P
where sum1>20000 and PartSale.p#=P.p#
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

pname
تیرآهن
آرماتور

تذکر: می‌توان دیدگاهی را روی یک دیدگاه دیگر ساخت.

مثال: دیدگاهی به نام PartSale2 ایجاد کنید که شامل نام محصولاتی باشد که میزان کل فروش آنها بیشتر از ۲۰۰۰۰ کیلوگرم است.

```
create view PartSale2(pname)
as
select pname
from PartSale,P
where sum1>20000 and PartSale.p#=P.p#
```

خروجی این پرس و جو به شکل ذیل خواهد بود:

PartSale2

Pname
تیرآهن
آرماتور

۳-۲-٤) حذف یک دیدگاه

قالب کلی این دستور به شکل ذیل است:

```
drop view نام دیدگاه
```

مثال: دیدگاه PartSale را حذف کنید.

```
drop view PartSale
```

تذکر: با حذف دیدگاه PartSale، دیدگاه PartSale2 نیز که روی آن ساخته شده است بطور خودکار نابود می‌شود.

٤-۲-٤) درج یک تاپل

قالب کلی این دستور به شکل ذیل است:

insert into نام جدول(نام ویژگیها) values(مقادیر ویژگیها)

مثال: در جدول مربوط به تولیدکنندگان، تولید کننده‌ای با کد 'S5' و نام 'بهین مصالح' اضافه کنید.

insert into S(s#,sname) values('S5','بهین مصالح')

٥-۲-٤) اصلاح تاپلها

قالب کلی این دستور به شکل ذیل است:

update نام جدول set نام ویژگی۱=مقدار۱ , مقدار۲=نام ویژگی۲ , ... [where شرط]

مثال: شهر کلیهٔ تولید کنندگان را به شیراز تغییر دهید.

update S set city='شیراز'

مثال: نام تولید کنندهٔ 'S1' را به 'بهین مصالح' و شهر وی را به 'شیراز' تغییر دهید.

update S set sname='بهین مصالح' , city='شیراز' where s#='S1'

٦-۲-٤) حذف تاپلها

قالب کلی این دستور به شکل ذیل است:

delete from نام جدول [where شرط]

مثال: کلیهٔ تولید کنندگان را حذف کنید.

delete from S

مثال: کلیهٔ تولید کنندگان تهرانی را حذف کنید.

delete from S where city='تهران'

۴-۳) دستورات کنترل داده‌ها

از دستورات DCL برای کنترل دسترسی کاربران مختلف به داده‌های پایگاه داده‌ها استفاده می‌شود. DBA وظیفه دارد برای هر کاربر یا گروه کاربران نام کاربری و کلمهٔ عبوری تعیین کرده، مجوزهای آنها را برای DBMS تعریف کند. DBMS هر کاربر را با توجه به نام کاربری وی شناسایی کرده و عملیات وی را کنترل می‌کند. چنانچه عملی با مجوزهای کاربر همخوانی نداشته باشد، DBMS از انجام آن سر باز می‌زند. دستورات DCL عبارتند از:

۴-۳-۱) دستور واگذاری مجوز

DBA وظیفه دارد پس از ایجاد پایگاه داده‌ها با توجه به حوزهٔ اختیارات هر کاربر مجوزهای دسترسی وی را برای DBMS تعریف کند. برخی از کاربران این امکان را دارند که کلیه و یا قسمتی از مجوزهای خود را با سایر کاربران سهیم شوند. قالب کلی دستور واگذاری مجوز به شکل ذیل است:

```
grant { select                          خواندن همهٔ ستونها
      { select(نام ستونها)              خواندن ستونهای مشخص شده
      { update                          اصلاح همهٔ ستونها
      { update(نام ستونها)              اصلاح ستونهای مشخص شده
      { insert                          درج تاپل
      { delete                          حذف تاپل
      { alter                           تغییر ساختار جدول
      { index                           ایجاد ایندکس
      { all                             کلیهٔ گزینه‌های بالا
        on نام جدول to نام کاربران [with grant option]
```

چنانچه عبارت with grant option ذکر شود، کاربر یا کاربرانی که مجوزهایی را دریافت می کنند به نوبهٔ خود می‌توانند این مجوزها را با دیگران سهیم شوند.

مثال: فرض کنید کاربری با نام کاربری ali بخواهد مجوز خواندن کلیهٔ ستونها و تغییر ستون sname و درج تاپل روی جدول S را با کاربری با نام کاربری arash سهیم شود و به وی اجازه دهد این مجوزها را در اختیار کاربران دیگر نیز قرار دهد. در این صورت ali با نام کاربری و کلمهٔ عبور خود وارد سیستم شده، دستور ذیل را صادر می کند:

```
grant select,update(sname),insert on S to arash with grant option
```

۲-۳-٤) دستور بازپس گیری مجوز

کاربرانی که مجوزهایی را در اختیار کاربران دیگر قرار داده‌اند، می‌توانند همه یا تعدادی از مجوزها را از آنها باز پس گیرند. فرض کنید کاربر A مجوزهایی را در اختیار کاربر B و کاربر B نیز این مجوزها را در اختیار کاربران C و D قرار داده باشد. در این صورت چنانچه کاربر A مجوزها را از کاربر B پس بگیرد، کاربران C و D نیز این مجوزها را از دست خواهند داد. قالب کلی دستور باز پس گیری مجوز به شکل ذیل است:

```
                ┌ select                          خواندن همهٔ ستونها
                ├ select(نام ستونها)              خواندن ستونهای مشخص شده
                ├ update                          اصلاح همهٔ ستونها
                ├ update(نام ستونها)              اصلاح ستونهای مشخص شده
revoke          ├ insert                          درج تاپل
                ├ delete                          حذف تاپل
                ├ alter                           تغییر ساختار جدول
                ├ index                           ایجاد ایندکس
                └ all                             کلیهٔ گزینه‌های بالا

on نام جدول  from نام کاربران
```

مثال: فرض کنید کاربری با نام کاربری ali بخواهد مجوز تغییر ستون sname از جدول S را از کاربری با نام کاربری arash پس بگیرد. در این صورت ali با نام کاربری و کلمهٔ عبور خود وارد سیستم شده، دستور ذیل را صادر می‌کند:

```
revoke update(sname) on S from arash
```

تمرینهای حل شده:

تمرین ۱: پرس و جوهای ذیل را در نظر بگیرید:

Q1: کد تولید کنندگانی را بیابید که آرماتور فروخته‌اند.

روش ۱: با استفاده از پیوند جداول

```
select s#
from SPJ,P
where pname='آرماتور' and P.p#=SPJ.p#
```

روش ۲:

تعبیر: از جدول تولید کنندگان کدهایی را انتخاب کنید که در جدول محصولات محصولی با نام آرماتور وجود دارد که این تولید کنندگان آن محصول را فروخته‌اند(در جدول فروش فروشهایی وجود دارند که مربوط به همین تولید کننده و همین محصول می‌باشند):

```
select s#
from S
where exists (select *
              from P
              where pname='آرماتور' and exists (select *
                                                from SPJ
                                                where SPJ.s#=S.s#
                                                and SPJ.p#=P.p#))
```

روش ۳:

تعبیر: از جدول فروش، کد تولید کنندگانی را انتخاب کنید که کد محصول فروخته شده توسط آنها جزء کد محصولاتی است که نام آنها آرماتور است.

```
select s#
from SPJ
where p# in (select p#
             from P
             where  pname='آرماتور' )
```

Q2: نام تولید کنندگانی را بیابید که 'آرماتور' فروخته‌اند.

روش ۱: با استفاده از پیوند جداول

```
select sname
from S,SPJ,P
where pname='آرماتور' and P.p#=SPJ.p#  and SPJ.s#=S.s#
```

روش ۲:

تعبیر: از جدول تولید کنندگان نامهایی را انتخاب کنید که در جدول محصولات محصولی با نام آرماتور وجود دارد و این تولید کنندگان آن محصول را فروخته‌اند (در جدول فروش فروشهایی وجود دارند که مربوط به همین تولید کننده و همین محصول می‌باشند):

```
select sname
from S
where exists (select *
              from P
              where  pname='آرماتور' and exists (select *
                                                 from SPJ
                                                 where SPJ.s#=S.s#
                                                 and SPJ.p#=P.p#))
```

روش ۳:

تعبیر: نام تولید کنندگانی را انتخاب کنید که کد آنها جزء کد تولید کنندگانی است که آرماتور را فروخته‌اند.

```
select sname
from S
where s# in (select s#
             from SPJ,P
             where  pname='آرماتور' and SPJ.p#=P.p#)
```

Q3: نام تولید کنندگانی را بیابید که آرماتور نفروخته‌اند.

روش ۱:

تعبیر: از جدول تولید کنندگان نامهایی را انتخاب کنید که در جدول محصولات محصولی وجود ندارد که نام آن آرماتور باشد و این تولید کنندگان آن محصول را فروخته باشند:

```
select sname
from S
where not exists (select *
                  from P
                  where  pname='آرماتور'
                     and exists (select *
                                 from SPJ
                                 where SPJ.s#=S.s#
                                    and SPJ.p#=P.p#))
```

روش ۲:

تعبیر: نام تولید کنندگانی را انتخاب کنید که کد آنها جزء کد تولید کنندگانی نیست که آرماتور را فروخته‌اند.

```
select sname
from S
where s# not in (select s#
                 from SPJ,P
                 where  pname='آرماتور' and SPJ.p#=P.p#)
```

Q4: لیستی از کد تولید کنندگان و تعداد محصولات فروخته شده توسط هر یک از آنها تهیه کنید.

```
select s#,count(distinct p#)
from SPJ
group by s#
```

Q5: کد تولیدکنندگانی را بیابید که سه محصول مختلف را فروخته‌اند.

```
select s#
from SPJ
group by s#
having count(distinct p#)=3
```

Q6: کد تولیدکنندگانی را بیابید که کلیهٔ محصولات را فروخته‌اند.

روش ۱:

تعبیر: کد تولید کنندگانی را بیابید که تعداد محصولات غیر تکراری فروخته شده توسط آنها با تعداد کل محصولات مساوی است:

```
select s#
from SPJ
group by s#
having count(distinct p#)=(select count(*)
                           from P)
```

روش ۲:

تعبیر: کد تولید کنندگانی را بیابید که محصولی وجود ندارد که آنها نفروخته باشند:

```
select s#
from S
where not exists(select *
                 from P
                 where not exists(select *
                                  from SPJ
                                  where SPJ.s#=S.s#
                                    and SPJ.p#=P.p#))
```

Q7: نام تولیدکنندگانی را بیابید که کلیهٔ محصولات را فروخته‌اند.

روش ۱:

تعبیر: نام تولید کنندگانی را بیابید که کد آنها جزء کد تولیدکنندگانی است که همهٔ محصولات را فروخته‌اند.

```
select sname
from S
where s# in (select s#
             from SPJ
```

```
        group by s#
        having count(distinct p#)=(select count(*)
                                   from P))
```

روش ۲:

تعبیر: نام تولید کنندگانی را بیابید که محصولی وجود ندارد که آنها نفروخته باشند.

```
select sname
from S
where not exists(select *
                 from P
                 where not exists(select *
                                  from SPJ
                                  where SPJ.s#=S.s#
                                    and SPJ.p#=P.p#))
```

Q8: کد محصولاتی را بیابید که برای بیش از ۷ پروژهٔ مختلف تهران فروخته شده‌اند.

```
select p#
from SPJ,J
where city='تهران' and J.j#=SPJ.j#
group by(p#)
having count(distinct j#)>7
```

Q9: نام محصولاتی را بیابید که برای بیش از ۷ پروژهٔ مختلف تهران فروخته شده‌اند.

```
select pname
from P
where p# in (select p#
             from SPJ,J
             where city='تهران' and J.j#=SPJ.j#
             group by(p#)
             having count(distinct j#)>7)
```

Q10: نام پروژه‌هایی را بیابید که برای آنها بیش از ۲۰۰۰۰ کیلوگرم سیمان خریداری شده است.

```
select jname
from J
where j# in (select j#
             from SPJ,P
             where pname='سیمان' and P.p#=SPJ.p#
             group by j#
             having sum(qty)>20000)
```

Q11: نام پروژه‌هایی را بیابید که تنها تولید کنندگان تهرانی در آنها همکاری داشته‌اند

روش ۱:

نام پروژه‌هایی که تولید کنندگان تهرانی در آنها همکاری داشته‌اند

-

نام پروژه‌هایی که تولید کنندگان غیر تهرانی در آنها همکاری داشته‌اند

نتیجهٔ مورد نظر

```
(select jname
 from J,SPJ,S
 where S.city='تهران' and S.s#=SPJ.s# and SPJ.j#=J.j#)
except
(select jname
 from J,SPJ,S
 where S.city<>'تهران' and S.s#=SPJ.s# and SPJ.j#=J.j#)
```

روش ۲:

تعبیر: نام پروژه‌هایی را بیابید که تولید کنندگان تهرانی وجود دارند که در آنها همکاری کرده باشند و تولید کنندگان غیر تهرانی وجود ندارند که در آنها همکاری کرده باشند.

```
select jname
from J
where exists (select *
              from S
              where city='تهران' and  exists  (select *
                                               from SPJ
                                               where SPJ.s#= S.s#
                                                 and SPJ.j#=J.j#))
and not exists(select *
               from S
               where city<>'تهران' and exists  (select *
                                                from SPJ
                                                where SPJ.s#= S.s#
                                                  and SPJ.j#=J.j#))
```

Q12: نام پروژه‌هایی را بیابید که هیچ تولید کنندهٔ غیر تهرانی در آنها همکاری نداشته است.

روش ۱:

نام همهٔ پروژه‌ها

-

نام پروژه‌هایی که تولید کنندگان غیر تهرانی در آنها همکاری داشته‌اند

نتیجهٔ مورد نظر

```
(select jname
 from J)
except
(select jname
 from J,SPJ,S
 where S.city<>'تهران' and S.s#=SPJ.s# and SPJ.j#=J.j#)
```

روش ۲:

```
select jname
from J
where not exists(select *
                 from S
                 where city<>'تهران'
                 and exists (select *
                             from SPJ
                             where SPJ.j#=J.j#  and SPJ.s#=S.s#))
```

تمرین ۲: پایگاه داده‌های سیستم کتابخانهٔ یک دانشگاه از جداول ذیل تشکیل شده است:

book(book# , bname , author , publisher , year , edit# , translator)

book#: شمارهٔ کتاب (به هر کتاب یک شمارهٔ منحصر بفرد داده شده است مثلاً اگر در کتابخانه سه کپی از یک کتاب وجود داشته باشد، هر یک از آنها یک شمارهٔ مجزا خواهند داشت)

bname: نام کتاب

author: نام نویسنده

publisher: نام ناشر

year: سال انتشار کتاب

edit#: ویرایش

translator: نام مترجم

student(st# , sname , field)

st#: شمارهٔ دانشجو

sname: نام دانشجو

field: رشتهٔ تحصیلی دانشجو

loan(loan# , book# , st# , loanDate , returnDate)

loan#: شمارهٔ امانت(به ازاء امانت هر کتاب یک شمارهٔ امانت منحصر بفرد اختصاص داده می‌شود)

book#: شمارهٔ کتاب

st#: شمارهٔ دانشجو

loanDate: تاریخ امانت گرفتن کتاب

returnDate: تاریخ برگشت کتاب به کتابخانه

book

book#	bname	author	publisher	year	edit#	translator
1400	شبکه‌های کامپیوتری و اینترنت	Douglas E.Comer	دانشگاه علم و صنعت ایران	1380	1	دکتر احمد اکبری- دکتر ناصر مزینی
1403	An introduction to database systems	C.I.Date	Addison Wesley	2000	7	
1404	Database systems,design,implementation, and management	Rob,Coronel	Boyed and fraser	1995	2	

student

st#	sname	field
8001	آرش راد	کامپیوتر
8002	عسل شاملو	ریاضی
8003	سیاوش آزاد	کامپیوتر
8101	ساغر راد	کامپیوتر

loan

loan#	book#	st#	loanDate	returnDate
1	1400	8001	83/06/06	83/06/10
2	1403	8001	83/06/06	83/06/10
3	1404	8003	83/06/10	83/06/16
4	1400	8003	83/06/11	83/06/20

Q13: نام کلیهٔ کتابهایی را بیابید که توسط "علی راد" امانت گرفته شده‌اند.

روش ۱:

```
select bname
from student,book,loan
where sname='علی راد'  and  student.st#=loan.st#  and
loan.book#=book.book#
```

روش۲:

```
select bname
from book
where exists (select *
              from student
              where sname='علی راد'
                 and exists(select *
                            from loan
                            where loan.st#=student.st#
                              and  loan.book#=book.book#))
```

روش۳:

```
select bname
from book
where exists ( select *
               from student,loan
               where sname='علی راد'
                 and loan.st#=student.st#
                 and  loan.book#=book.book#)
```

روش٤:

```
select bname
from book
where book# in (select book#
                from student,loan
                where sname='علی راد'
                  and loan.st#=student.st#)
```

روش۵:

```
select bname
from book
where book# in (select book#
                from loan
                where st# in (select st#
                              from student
                              where sname='علی راد'))
```

Q14: کتاب شمارهٔ ۱۰۰۳ تا کنون چند بار از کتابخانه امانت گرفته شده است؟

```
select count(*)
from loan
where book#=1003
```

Q15: نام کتابهایی را بیابید که تا کنون بیش از ۲۰ بار امانت گرفته شده‌اند.

روش۱:

```
select bname
from book
where book# in (select book#
                from loan
                group by book#
                having count(*)>20)
```

روش۲:

ابتدا یک دیدگاه شامل شمارهٔ کلیهٔ کتابها و تعداد دفعاتی که هر یک از آنها امانت گرفته شده‌اند ایجاد می‌کنیم:

```
create view view1(book#,count1)
as
select book#,count(*)
from loan
group by book#
```

سپس تاپلهایی از دیدگاه را که در آنها تعداد دفعات امانت بزرگتر از ۲۰ است انتخاب کرده، با جدول book پیوند می‌دهیم تا نام کتابها را به دست آوریم:

```
select bname
```

```
from book,view1
where count1>20 and view1.book#=book.book#
```

روش ۳:

ابتدا یک دیدگاه شامل شمارهٔ کلیهٔ کتابهایی که تعداد دفعات امانت آنها بزرگتر از ۲۰ است ایجاد می‌کنیم:

```
create view view1(book#)
as
select book#
from loan
group by book#
having count(*)>20
```

سپس برای به دست آوردن نام کتابها، دیدگاه را با جدول book پیوند می‌دهیم:

```
select bname
from book,view1
where  view1.book#=book.book#
```

تذکر: چنانچه مجاز به ایجاد دیدگاه باشیم می‌توانیم از روش ۲ یا ۳ استفاده کنیم ولی استفاده از دیدگاهها تنها در موارد خاص کاربرد دارد و در حالت کلی برای یک پرس و جوی ساده و یا کم کاربرد از دیدگاهها استفاده نمی‌شود.

Q16: نام کتابهایی را بیابید که توسط کلیهٔ دانشجویان رشتهٔ کامپیوتر امانت گرفته شده‌اند.

```
select bname
from book
where not exists(select *
                 from student
                 where field='کامپیوتر'
                 and not exists(select *
                                from loan
                                where loan.st#=student.st#
                                  and loan.book#=book.book#))
```

Q17: نام دانشجویانی را بیابید که تا کنون هیچ کتابی را به امانت نگرفته‌اند.

روش ۱:

```
select sname
from student
where not exists(select *
                 from loan
                 where loan.st#=student.st#)
```

روش ۲:

```
select sname
from student
where st# not in (select st#
                  from loan)
```

Q18: دیدگاهی ایجاد کنید که شامل نام دانشجویان و تعداد کل کتابهای به امانت گرفته شده توسط هر یک از آنها باشد.

بعلت محدودیتی که در استفاده از group by وجود دارد نمی‌توان این دیدگاه را مستقیماً ایجاد کرد چون اگر بخواهیم در قسمت select از ویژگی sname استفاده کنیم، باید در قسمت group by نیز از ویژگی sname استفاده کنیم و چون امکان تشابه اسمی دانشجویان وجود دارد، گروه بندی بر اساس نام دانشجویان منطقی نیست(مثلاً اگر دو دانشجو بنام "علی راد" وجود داشته باشند و یکی از آنها ۲۰ کتاب و دیگری ۱۵ کتاب امانت گرفته باشد، در لیست نتیجه تنها یک "علی راد" و با تعداد کتاب ۳۵ ظاهر خواهد شد که این درست نیست) و بهتر است گروه‌بندی را بر اساس st# انجام دهیم تا امکان بروز خطا را از بین ببریم. پس ابتدا یک دیدگاه با نام view1 شامل شماره دانشجویی کلیهٔ دانشجویان و تعداد کتابهای مختلفی که هر یک از آنها امانت گرفته‌اند تهیه می‌کنیم:

```
create view view1(st#,count1)
as
select st#,count(distinct book#)
from loan
group by st#
```

سپس، روی view1 دیدگاه دیگری با نام view2 ایجاد می‌کنیم که شامل نام دانشجویان و تعداد کتابهای امانت گرفته شده توسط هر یک از آنها باشد:

```
create view view2(sname,count1)
as
select sname,count1
from student, view1
where student.st#=view1.st#
```

Q19: نام کتابهای پایگاه داده‌ای را بیابید که نویسندهٔ آنها Coronel است.

```
select bname
from book
where bname like '%Database%' and author like '%Coronel%'
```

تمرین ۳: فرض کنید پایگاه داده‌ها در سیستم یک دانشگاه شامل جداول ذیل باشد:

field(field# , fieldname)

در این جدول اطلاعات مربوط به رشته‌های تحصیلی ذخیره می‌شود:

#field: کد رشتهٔ تحصیلی (برای هر رشتهٔ تحصیلی یک کد منحصر بفرد در نظر گرفته شده است)

fieldname: نام رشتهٔ تحصیلی

type(type# , typeName , fee)

در این جدول اطلاعات مربوط به نوع دروس ذخیره می‌شود:

#type: کد نوع درس (به هر نوع درس یک کد منحصر بفرد داده شده است)

typeName: نوع درس

fee: قیمت هر واحد درس

student(st# , sname , startYear , field#)

در این جدول اطلاعات مربوط به دانشجویان ذخیره می‌شود:

#st: شماره دانشجویی

sname: نام دانشجو

startYear: سال ورود به دانشگاه

field#: کد رشتهٔ تحصیلی دانشجو

course(crs# , cname , unit , type#)

در این جدول اطلاعات دروس ذخیره می‌شود:

crs#: شمارهٔ درس

cname: نام درس

unit: تعداد واحد درس

type#: کد نوع درس(نظری، عملی،...)

CF(crs# , field# , kind)

در این جدول مشخص می‌شود هر درس مربوط به کدام رشته یا رشته‌های تحصیلی است:

crs#: شمارهٔ درس

field#: کد رشتهٔ تحصیلی

kind: این ویژگی مشخص می‌کند درس مورد نظر برای رشتهٔ مورد نظر چه حالتی دارد('P' برای پیش نیاز، 'A' برای پایه، 'T' برای تخصصی ،'O' برای عمومی‌و 'E' برای اختیاری)

G(st# , crs#, term , grade)

در این جدول اطلاعات مربوط به نمرات دانشجویان ذخیره می‌شود:

st#: شماره دانشجویی

crs#: شمارهٔ درس

term: نیمسال اخذ درس توسط دانشجو

grade: نمرهٔ اخذ شده

pre(crs# , pre#)

در این جدول پیش نیازهای هر درس مشخص می‌شوند:

crs#: شمارهٔ درس

pre#: شمارهٔ درس پیش نیاز

prof (prof# , pname , degree)

در این جدول اطلاعات مربوط به اساتید ذخیره می‌شود:

prof#:شمارهٔ استاد (به هر استاد یک شماره منحصر بفرد داده شده است)

pname: نام استاد

degree: آخرین مدرک تحصیلی استاد('D' برای دکترا، 'F' برای فوق لیسانس و 'L' برای لیسانس)

PC(<u>prof# , course# , term</u>)

در این جدول مشخص می‌شود هر استاد در هر نیمسال چه دروسی را تدریس کرده است:

prof#: شمارهٔ استاد

course#: شمارهٔ درس

term: نیمسال تحصیلی

tuition(<u>field# , startYear</u> , constTuition)

در این جدول بر اساس سال ورود و رشتهٔ تحصیلی شهریهٔ ثابت مشخص می‌شود:

field#: کد رشتهٔ تحصیلی

startYear: سال ورود به دانشگاه

constTuition: شهریهٔ ثابت

به نمونه‌هایی از اطلاعات ذخیره شده در این جداول توجه کنید:

field

field#	fieldName
1	مهندسی کامپیوتر
2	مهندسی الکترونیک
3	ریاضی محض

type

type#	typeName	fee
1	نظری	5000
2	عملی	20000
3	آزمایشگاه	30000

tuition

field#	startYear	constTuition
1	80	75000
1	81	80000
1	82	90000
2	81	80000
2	82	90000
3	82	75000

st

st#	sname	startYear	field#
8001	آرش راد	80	1
8002	عسل شاملو	80	3
8003	سیاوش آزاد	80	1
8101	ساغر راد	81	1
8102	علی نیکی	81	1
8111	فرامرز نیکی	81	1
8112	علی رضایی	81	2

course

crs#	cname	unit	type#
1100	ادبیات	2	1
1105	تربیت بدنی	1	2
1400	برنامه سازی ۱	3	1
1402	برنامه سازی ۲	3	1
1403	ذخیره و بازیابی	3	1
1407	پایگاه داده‌ها	3	1
1500	ریاضی ۱	3	1
1600	زبان پیش نیاز	2	1

pre

crs#	pre#
1400	1500
1402	1400
1407	1402
1407	1403

G

St#	crs#	term	grade
8001	1400	811	9
8001	1400	812	13
8001	1401	811	13
8101	1400	821	19
8101	1500	821	14
8111	1400	811	12
8111	1401	811	20

prof

prof#	pname	degree
101	فرانک شایسته	L
102	علی پیامی	F
106	آزاده نیکوکار	F
107	سیامک فرزانه	D
108	علی نادرنژاد	F

PC

prof#	crs#	term
101	1400	801
101	1400	802
101	1401	801
102	1400	801
108	1500	811

CF

crs#	field#	kind
1100	1	O
1100	2	O
1100	3	O
1105	1	O
1105	2	O
1105	3	O
1400	1	T
1400	2	E
1400	3	E
1402	1	T
1403	1	T
1407	1	T
1500	1	P
1500	2	P
1500	3	P

Q20: نام دانشجویانی را بیابید که در نیمسال اول ۸۰ درس پایگاه داده‌ها را گرفته‌اند.

روش ۱:

```
select sname
from course,G,st
where cname='پایگاه داده‌ها'
   and course.crs#=G.crs#
   and term=801
   and G.st#=st.st#
```

روش ۲:

```
select sname
from st
where  exists (select *
                from  G,course
                where cname='پایگاه داده‌ها'
                      and course.crs#=G.crs#
                      and term=801
                      and G.st#=st.st#)
```

روش ۳:

```
select sname
from st
where  st#  in (select st#
                 from  G,course
                 where cname='پایگاه داده‌ها'
                    and course.crs#=G.crs#
                    and term=801)
```

Q21: نام دانشجویانی را بیابید که در نیمسال اول ۸۰ درس پایگاه داده‌ها را نگرفته‌اند.

روش ۱:

```
select sname
from st
where  not exists (select *
                    from  course
                    where cname='پایگاه داده‌ها'
```

```
                and exists(select *
                           from G
                           where term=801
                             and G.crs#=course.crs#
                             and G.st#=st.st#))
```

روش ۲:

```
select sname
from st
where  not exists (select *
                   from  course,G
                   where cname='پایگاه داده‌ها'
                      and term=801
                      and G.crs#=course.crs#
                      and G.st#=st.st#)
```

روش ۳:

```
select sname
from st
where  st#  not  in (select st#
                     from  G,course
                     where cname='پایگاه داده‌ها'
                        and course.crs#=G.crs#
                        and  term=801 )
```

Q22: نام دانشجویانی را بیابید که در نیمسال اول ۸۰ هیچ درسی نگرفته‌اند.

روش ۱:

```
select sname
from st
where not  exists (select *
                   from  G
                   where G.st#=st.st# and term=801)
```

روش ۲:

```
select sname
from st
where  st#  not in (select st#
                    from  G
                    where term=801 )
```

Q23: لیستی از نام دروس تخصصی رشتهٔ مهندسی کامپیوتر تهیه کنید.

```
select cname
from field,CF,course
where fieldName='مهندسی کامپیوتر'
   and  CF.field#= field.field#
   and   kind='T'
   and CF.crs#=course.crs#
```

Q24: تعداد دروس تخصصی رشتهٔ مهندسی کامپیوتر را به دست آورید.

```
select count(*)
from field,CF
where fieldName='مهندسی کامپیوتر'
   and  CF.field#= field.field#
   and   kind='T'
```

Q25: لیستی از شمارهٔ دانشجویان مهندسی کامپیوتر و تعداد کل درسهای تخصصی گذرانده شده توسط هر یک از آنها تهیه کنید.

```
select G.st#,count(distinct crs#)
from st,field,G,CF
where fieldName='مهندسی کامپیوتر'
     and st.field#=field.field#
     and CF.field#= field.field#
     and kind='T'
     and G.st#=st.st#
     and G.crs#=CF.crs#
     and grade>=10
group by(G.st#)
```

Q26: شمارهٔ دانشجویان مهندسی کامپیوتری را بیابید که کلیهٔ درسهای تخصصی رشتهٔ خود را گذرانده‌اند.

روش ۱:

تعبیر: شمارهٔ دانشجویان مهندسی کامپیوتری را بیابید که تعداد دروس تخصصی گذرانده شده توسط آنها با تعداد کل دروس تخصصی رشتهٔ کامپیوتر مساوی است.

```
select G.st#
from st,field,G,CF
where fieldName='مهندسی کامپیوتر'
      and st.field#=field.field#
      and CF.field#= field.field#
      and kind='T'
      and st.st#=G.st#
      and G.crs#=CF.crs#
      and grade>=10
group by(G.st#)
having count(distinct G.crs#)=(select count(*)
                               from field,CF
                               where fieldName='مهندسی کامپیوتر'
                                     and  CF.field#= field.field#
                                     and   kind='T')
```

روش ۲:

تعبیر: شمارهٔ دانشجویان مهندسی کامپیوتری را بیابید که هیچ درس تخصصی مربوط به مهندسی کامپیوتر وجود ندارد که این دانشجویان آنها را نگذرانده باشند.

```
select st#
from st,field
where fieldName='مهندسی کامپیوتر'
      and st.field#=field.field#
      and  not exists (select *
                       from CF
                       where CF.field#= field.field#
```

```
                and kind='T'
                and not exists(select *
                                from G
                                where G.st#=st.st#
                                  and G.crs#=CF.crs#
                                  and grade>=10))
```

Q27: شمارهٔ دانشجویان مهندسی کامپیوتری را بیابید که کلیهٔ درسهای تخصصی رشتهٔ خود را نگذرانده‌اند.

روش ۱:

تعبیر: شمارهٔ دانشجویان مهندسی کامپیوتری را بیابید که تعداد دروس تخصصی گذرانده شده توسط آنها با تعداد کل دروس تخصصی رشتهٔ کامپیوتر مساوی نیست.

```
select G.st#
from st,field,G,CF
where fieldName='مهندسی کامپیوتر'
    and st.field#=field.field#
    and CF.field#= field.field#
    and kind='T'
    and st.st#=G.st#
    and G.crs#=CF.crs#
    and grade>=10
group by(G.st#)
having count(distinct G.crs#)<>(select count(*)
                                from field,CF
                                where fieldName='مهندسی کامپیوتر'
                                  and  CF.field#= field.field#
                                  and  kind='T')
```

روش ۲: شمارهٔ دانشجویان مهندسی کامپیوتری را بیابید که درس تخصصی مربوط به مهندسی کامپیوتر وجود دارد که این دانشجویان آنها را نگذرانده باشند.

```
select st#
from st,field
where fieldName='مهندسی کامپیوتر'
  and  st.field#=field.field#
  and   exists (select *
                from CF
                where CF.field#= field.field#
                    and kind='T'
                    and not exists(select *
                                   from G
                                   where G.st#=st.st#
                                     and G.crs#=CF.crs#
                                     and grade>=10))
```

Q28: لیستی از شماره دانشجویی کلیهٔ دانشجویان و تعداد کل واحدهای گذرانده شده توسط هر یک از آنها تهیه کنید.

```
select st#,sum(unit)
from G,course
where G.crs#=course.crs# and grade>=10
group by st#
```

Q29: لیستی از نام کلیهٔ دانشجویان و تعداد کل واحدهای گذرانده شده توسط هر یک از آنها تهیه کنید.

توضیح: ابتدا باید یک دیدگاه شامل شمارهٔ دانشجویان و تعداد کل واحدهای گذرانده شده توسط هر یک از آنها تهیه کنیم:

```
create view view1(st#,sum1)
as
select st#,sum(unit)
from G,course
where G.crs#=course.crs# and grade>=10
group by st#
```

سپس پرس و جو را به شکل ذیل می‌نویسیم:

```
select sname,sum1
from st,view1
where st.st#=view1.st#
```

Q30: لیستی از شماره دانشجویانی که مجموعاً بیش از ۱۰۰ واحد درس گذرانده‌اند تهیه کنید.

```
select st#
from G,course
where G.crs#=course.crs# and grade>=10
group by st#
having sum(unit)>100
```

Q31: لیستی از نام دانشجویانی که مجموعاً بیش از ۱۰۰ واحد درس گذرانده‌اند تهیه کنید.

```
select sname
from S
where st# in (select st#
              from G,course
              where G.crs#=course.crs# and grade>=10
              group by st#
              having sum(unit)>100)
```

Q32: لیستی از شماره دانشجویی کلیهٔ دانشجویان ورودی ۸۰ و معدل هر یک از آنها در نیمسال اول ۸۱ تهیه کنید.

```
select G.st#,sum(unit*grade)/sum(unit)
from st,G,course
where startYear=80
  and st.st#=G.st#
  and G.crs#=course.crs#
  and term=811
group by G.st#
```

Q33: لیستی از نام دانشجویان ورودی ۸۰ که معدل آنها در نیمسال اول ۸۱ بیشتر از ۱۸ بوده است تهیه کنید.

تعبیر: لیستی از نام دانشجویان ورودی ۸۰ تهیه کنید که شمارهٔ آنها جزء شماره دانشجویانی است که معدل آنها بالاتر از ۱۸ است.

```
select sname
from st
where startYear=80 and st# in(select st#
                              from G,course
                              where G.crs#=course.crs#
                                  and term=811
                              group by st#
                              having sum(unit*grade)/sum(unit)>18)
```

Q34: لیستی از نام اساتیدی که درس پایگاه داده‌ها را تدریس کرده‌اند تهیه کنید.

```
select pname
from professor,PC,course
where cname='پایگاه داده‌ها'
    and course.crs#=PC.crs#
    and PC.prof#=professor.prof#
```

Q35: لیستی از شمارهٔ اساتیدی که درس پایگاه داده‌ها را بیش از چهار ترم تدریس کرده‌اند تهیه کنید.

```
select prof#
from PC,course
where cname='پایگاه داده‌ها' and  PC.crs#=course.crs#
group by(prof#)
having count(distinct term)>4
```

Q36: لیستی از نام دروس عملی تهیه کنید که توسط استاد شمارهٔ ۱۰۰ در نیمسال دوم ۸۳ تدریس شده‌اند.

```
select cname
from course,type,PC
where typeName='عملی'
    and type.type#=course.type#
    and course.crs#=PC.crs#
    and  prof#=100
    and term=832
```

Q37: لیستی از شمارهٔ دروسی تهیه کنید که پیش نیاز درس پایگاه داده‌ها هستند.

```
select pre#
from pre,course
where cname='پایگاه داده‌ها' and course.crs#=pre.crs#
```

Q38: لیستی از نام دروسی تهیه کنید که پیش نیاز درس پایگاه داده‌ها هستند.

```
select cname
from course
where crs# in(select pre#
              from pre,course
              where  cname='پایگاه داده‌ها' and pre.crs#=course.crs#)
```

Q39: لیستی از نام دروسی تهیه کنید که مربوط به رشتهٔ تحصیلی دانشجوی شمارهٔ ۸۰۰۱ هستند.

```
select cname
from course,CF,st
where st#=8001 and st.field#=CF.field# and CF.crs#=course.crs#
```

Q40: لیستی از نام دروسی تهیه کنید که دانشجوی ۸۰۰۱ آنها را نگذرانده است.

```
select cname
from course
where not exists(select *
                 from G
                 where grade>=10
                       and st#=8001
                       and G.crs#=course.crs#)
```

Q39: لیستی از نام دروسی تهیه کنید که مربوط به رشتهٔ تحصیلی دانشجوی شمارهٔ ۸۰۰۱ هستند و این دانشجو آن دروس را نگذرانده است.

```
select cname
from course
where crs# in(select crs#
              from st,CF
              where st#=8001 and st.field#=CF.field#
              and not exists(select *
                             from G
                             where G.st#=st.st#
                               and G.crs#=CF.crs#
                               and grade>=10))
```

Q40: لیستی از نام دروسی تهیه کنید که دانشجوی شمارهٔ ۸۰۰۱ همهٔ پیش نیازهای آنها را گذرانده است.

تعبیر: نام دروسی را بیابید که هیچ پیش نیازی ندارند که دانشجوی شمارهٔ ۸۰۰۱ آنها را نگذرانده باشد.

```
select cname
from course
where not exists(select *
                 from pre
                 where pre.crs#=crs.crs#
                       and not exists(select *
                                      from G
                                      where G.crs#=pre.pre#
                                        and  st#=8001
                                        and grade>=10))
```

Q41: لیستی از نام دروسی تهیه کنید که مربوط به رشتهٔ تحصیلی دانشجوی شمارهٔ ۸۰۰۱ هستند و این دانشجو آن دروس را نگذرانده است ولی کلیهٔ پیش نیازهای آنها را گذرانده است.

```
select cname
from course
where crs# in(select crs#
              from st,CF
              where st#=8001 and st.field#=CF.field#
              and not exists(select *
                             from G
                             where G.st#=8001
                               and G.crs#=CF.crs#
                               and grade>=10)
              and not exists(select *
                             from pre
                             where pre.crs#=course.crs#
                               and not exists(select *
                                              from G
                                              where G.st#=8001
                                                and crs#=pre.pre#
                                                and grade>=10)))
```

Q42: شهریهٔ ثابت دانشجوی شمارهٔ 8001 را به دست آورید.

```
select constTuition
from tuition,st
where st#=8001
   and tuition.field#=st.field#
   and tuition.startYear=st.startYear
```

Q43: هزینهٔ هر واحد درس پایگاه داده‌ها را به دست آورید.

```
select fee
from course,type
where cname='پایگاه داده‌ها' and course.type#=type.type#
```

Q44: هزینهٔ کل واحدهای دانشجوی شمارهٔ ۸۰۰۱ در نیمسال دوم ۸۰ را به دست آورید.

```
select sum(fee*unit)
from G,course,type
where st#=8001
 and  G.st#=st.st#
 and G.crs#=course.crs#
 and course.type#=type.type#
 and term=802
```

Q45: لیستی از نام دروسی که مربوط به رشتهٔ مهندسی کامپیوتر یا رشتهٔ ریاضی محض هستند به دست آورید.

روش ۱:

```
select cname
 from course,CF,field
 where course.crs#=CF.crs#
       and CF.field#=field.field#
       and (fieldname='مهندسی کامپیوتر' or fieldname='مهندسی الکترونیک')
```

روش ۲:

```
(select cname
 from course,CF,field
 where course.crs#=CF.crs#
       and CF.field#=field.field#
       and fieldname='مهندسی کامپیوتر')
union
(select cname
 from course,CF,field
 where course.crs#=CF.crs#
       and CF.field#=field.field#
       and fieldname='مهندسی الکترونیک')
```

Q46: نام و معدل دانشجویان رشتهٔ کامپیوتری را بیابید که معدل آنها از میانگین نمرات کلیهٔ دانشجویان بیشتر است.

ابتدا یک دیدگاه شامل شماره دانشجویی و معدل کلیهٔ دانشجویان رشتهٔ کامپیوتری که معدل آنها از میانگین نمرات کلیهٔ دانشجویان بیشتر است ایجاد می‌کنیم:

```
create view view1(st#,ave)
as
select G.st#,sum(unit*grade)/sum(unit)
from field,st,G,course
where fieldname='مهندسی کامپیوتر'
     and  st.field#=field.field#
     and st.st#=G.st#
     and G.crs#=course.crs#
group by G.st#
having sum(unit*grade)/sum(unit) > (select sum(unit*grade)/sum(unit)
                                    from G,course
                                    where G.crs#=course.crs#)
```

سپس این دیدگاه را با st پیوند می‌دهیم تا نام دانشجویان را به دست آوریم:

```
select sname,ave
from st,view1
where st.st#=view1.st#
```

تمرینهای فصل:

۱- پایگاه داده‌های نمونه سوال ۳ را در نظر گرفته، پرس و جوهای ذیل را به زبان SQL بنویسید:

Q47: تعداد اساتیدی را بیابید که دارای مدرک دکترا هستند.

Q48: نام دانشجویانی را بیابید که هم درس پایگاه داده‌ها و هم درس مهندسی اینترنت را گذرانده‌اند.

Q49: نام دانشجویانی را بیابید که درس پایگاه داده‌ها را گذرانده‌اند ولی درس مهندسی اینترنت را نگذرانده‌اند.

Q50: تعداد دروس تخصصی را که دانشجوی شمارهٔ ۸۰۰۱ گذرانده است به دست آورید.

Q51: لیستی از شمارهٔ دانشجویان و تعداد کل واحدهای تخصصی گذرانده شده توسط هر یک از آنها تهیه کنید.

Q52: لیستی از نام دانشجویان و تعداد کل واحدهای تخصصی گذرانده شده توسط هر یک از آنها تهیه کنید.

Q53: میانگین نمرات درس پایگاه داده‌ها در نیمسال اول ۸۰ را به دست آورید.

Q54: لیستی از نیمسالهای تحصیلی و میانگین نمرات درس پایگاه داده‌ها در هر یک از آنها تهیه کنید.

Q55: میانگین کل نمرات دانشجویان رشتهٔ مهندسی کامپیوتر در نیمسال دوم ۸۳ را به دست آورید.

Q56: لیستی از نام رشته‌های تحصیلی و میانگین کل نمرات هر یک از این رشته‌ها در نیمسال دوم ۸۳ تهیه کنید.

Q57: لیستی از نام رشته‌های تحصیلی که میانگین کل نمرات دانشجویان آنها در نیمسال اول ۸۳ بیشتر از ۱۴ بوده است تهیه کنید.

Q58: لیستی ازشمارهٔ دانشجویانی که هم در نیمسال اول ۸۳ و هم در نیمسال دوم ۸۳ مشروط شده‌اند تهیه کنید.

Q59: کل هزینه ثبت نام دانشجوی شمارهٔ ۸۰۰۱ شامل شهریهٔ ثابت و هزینهٔ مربوط به واحدها در نیمسال اول سال ۸۲ را محاسبه کنید.

Q60: لیستی از نام دانشجویانی که بیش از نصف دروس رشتهٔ مهندسی کامپیوتر را گذرانده‌اند تهیه کنید.

Q61: لیستی از نام اساتید و تعداد کل واحدهای تدریس شده توسط هر یک از آنها در نیمسال اول ۸۳ تهیه کنید.

Q62: لیستی از شمارهٔ اساتیدی که در نیمسال اول ۸۳ بیش از ۲۳ واحد درس داشته‌اند تهیه کنید.

Q63: لیستی از نام اساتیدی که در نیمسال اول ۸۳ بیش از ۲۳ واحد درس داشته‌اند تهیه کنید.

Q64: لیستی از شماره‌های دانشجویی دانشجویان رشتهٔ کامپیوتر و هزینهٔ کل واحدهای انتخاب شده توسط آنها در نیمسال اول ۸۳ تهیه کنید.

Q65: لیستی از نام دانشجویان رشتهٔ کامپیوتر و هزینهٔ کل واحدهای انتخاب شده توسط آنها در نیمسال اول ۸۳ تهیه کنید.

Q66: لیستی از نام کلیهٔ اساتید و تعداد کل دروسی که هر یک از آنها تا کنون تدریس کرده‌اند تهیه کنید.

Q67: لیستی از نام اساتیدی که تا کنون بیش از چهار درس مختلف تدریس کرده‌اند تهیه کنید.

Q68: معدل کل دانشجوی شمارهٔ ۸۰۰۲ را محاسبه کنید.

Q69: لیستی از نام دروس تخصصی رشتهٔ کامپیوتر که قیمت هر واحد آنها بیشتر از ۸۰۰۰ تومان است تهیه کنید.

Q70: نام اساتیدی را بیابید که در نیمسال اول ۸۰ هم دروس تخصصی رشتهٔ مهندسی کامپیوتر و هم دروس تخصصی رشتهٔ مهندسی الکترونیک را تدریس کرده‌اند.

Q71: نام اساتیدی را بیابید که در نیمسال اول ۸۰ دروس تخصصی رشتهٔ مهندسی کامپیوتر را تدریس کرده‌اند ولی دروس تخصصی رشتهٔ مهندسی الکترونیک را تدریس نکرده‌اند.

Q72: نام اساتیدی را بیابید که کلیهٔ دروس رشتهٔ کامپیوتر را تدریس کرده‌اند.

Q73: لیستی از نام دروس عملی تهیه کنید که دانشجوی ۸۰۰۱ آنها را نگذرانده است.(دروسی که این دانشجو نگرفته و یا درآنها نمرهٔ قبولی نگرفته است)

Q74: لیستی از نام دروسی که مربوط به هر دو رشتهٔ مهندسی کامپیوتر و ریاضی محض هستند به دست آورید.

Q75: لیستی از نام دروس عملی کامپیوتر تهیه کنید که توسط اساتید لیسانسه تدریس نشده‌اند.

Q76: لیستی از نام دروس عملی کامپیوتر تهیه کنید که در کلیهٔ ترمها تنها توسط اساتیدی که دارای مدرک فوق لیسانس هستند تدریس شده‌اند.

۲) فرض کنید پایگاه داده‌های یک وب سایت اطلاع رسانی در مورد سینما از جداول ذیل تشکیل شده باشد:

film (film# , fname , year , subject)

film# : شمارهٔ فیلم

fname: نام فیلم

year: سال ساخت

subject: موضوع فیلم (اجتماعی، خانوادگی...)

people(id , name , biography)

در این جدول مشخصات کلیهٔ کارگردانان، بازیگران، نویسندگان و تهیه کنندگان نگهداری می‌شود. به هر شخص یک شناسهٔ منحصر بفرد داده شده است. یک شخص ممکن است مثلاً هم کارگردان و هم بازیگر و هم نویسنده باشد ولی اطلاعات وی تنها یکبار ثبت می‌شود.

id: شناسه یا شمارهٔ شخص

name: نام

biography: زندگینامهٔ شخص

Film_Director (film# , id)

film#: شمارهٔ فیلم

id: شمارهٔ شخصی که فیلم را کارگردانی کرده است (توجه کنید یک فیلم ممکن است بیش از یک کارگردان داشته باشد.)

Film_Writer (film# , id)

#film: شمارهٔ فیلم

id: شمارهٔ شخصی که فیلمنامهٔ فیلم را نوشته است (توجه کنید یک فیلم ممکن است بیش از یک نویسنده داشته باشد.)

Film_Actor (film# , id , role)

#film: شمارهٔ فیلم

id: شمارهٔ شخصی که در فیلم بازی کرده است (توجه کنید یک فیلم بیش از یک بازیگر دارد.)

role: نوع نقش (نقش اول یا دوم...)

Film_Producer (film# , id)

#film: شمارهٔ فیلم

id: شمارهٔ شخصی که فیلم را تهیه کرده است (توجه کنید یک فیلم ممکن است بیش از یک تهیه کننده داشته باشد.)

award (award# , aname)

#award: کد جایزه (به هر نوع جایزه مانند بهترین کارگردانی، بهترین بازیگر نقش اول زن و ... یک کد اختصاص یافته است.)

aname: نام جایزه

festival (fes# , fname , year , place)

#fes: شمارهٔ جشنواره

fname: نام جشنواره

year: سال برگزاری جشنواره

place: محل برگزاری جشنواره

Fes_Film (fes# , film#)

در این جدول مشخص می‌شود در هر جشنواره چه فیلمهایی شرکت کرده‌اند. در هر جشنواره فیلمهای متعددی شرکت می‌کنند و هر فیلم ممکن است در جشنواره‌های مختلف شرکت کند.

fes#: شمارهٔ جشنواره

film#: شمارهٔ فیلم

grant (<u>fes# , film# , award# , id</u> , kind , result)

در این جدول اطلاعات مربوط به مقامهایی که فیلمهای مختلف در جشنواره‌های مختلف کسب کرده‌اند ذخیره می‌شود. اطلاعات ذخیره شده باید نمایانگر اطلاعاتی به فرم ذیل باشد:

در جشنوارهٔ جشن خانهٔ سینما-دورهٔ چهارم، سلیمه رنگزن برای بازی در فیلم عروس آتش برندهٔ بهترین بازیگر نقش دوم زن شد و دیپلم افتخار گرفت.

fes#: شمارهٔ جشنواره

film#: شمارهٔ فیلم

award#: کد نوع جایزه

id: شمارهٔ شخصی که جایزه به وی اختصاص یافته است

kind: نوع جایزه (دیپلم افتخار، سیمرغ بلورین...)

result: نتیجه(برنده یا کاندیدا)

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q77: نام کارگردان فیلم قرمز را به دست آورید.

Q78: نام فیلمهایی را بیابید که در جشنوارهٔ هیجدهم فجر جایزهٔ بهترین کارگردانی را گرفته‌اند.

Q79: نام اشخاصی را بیابید که هم بازیگری و هم کارگردانی کرده‌اند.

Q80: نام اشخاصی را بیابید که بیش از ۱۰ فیلم را کارگردانی کرده‌اند.

Q81: نام اشخاصی را بیابید که بیش از ۳ بار جایزهٔ بهترین بازیگر نقش اول مرد را به خود اختصاص داده‌اند.

Q82: نام اشخاصی را بیابید که هم نویسنده و هم کارگردان یک یا چند فیلم بوده‌اند.

Q83: نام اشخاصی را بیابید که در بیش از ۵ فیلم مختلف نقش اول داشته‌اند.

Q84: نام فیلمهایی را بیابید که آقای علی حاتمی آنها را کارگردانی کرده است.

Q85: نام اشخاصی را بیابید که تنها در نقش اول بازی کرده‌اند.

Q86: نام اشخاصی را بیابید که هم در نقش اول و هم در نقش دوم بازی کرده‌اند.

Q87: نام اشخاصی را بیابید که تنها بازیگر هستند.

Q88: نام اشخاصی را بیابید که تعداد بازیهای آنها در نقش دوم بیشتر از تعداد بازیهای آنها در نقش اول بوده است.

Q89: نام اشخاصی را بیابید که تنها فیلمهای خانوادگی کارگردانی کرده‌اند.

Q90: نام فیلمهایی را بیابید که در جشنوارهٔ فجر-دورهٔ هیجدهم برندهٔ بیش از ۴ نوع جایزهٔ مختلف شده‌اند.

Q91: نام جایزه‌هایی را بیابید که فیلم عروس آتش در جشنواره‌های مختلف کاندیدای دریافت آنها شده است.

Q92: تعداد جایزه‌هایی را بیابید که فیلم عروس آتش در جشنواره‌های مختلف کاندیدای دریافت آنها شده است.

Q93: تعداد فیلمهایی را بیابید که در جشنوارهٔ فجر-دورهٔ هیجدهم شرکت کرده‌اند.

Q94: تعداد فیلمهایی را بیابید که هم در جشنوارهٔ فجر-دورهٔ هیجدهم و هم در جشنوارهٔ خانهٔ سینما-دورهٔ چهارم شرکت کرده‌اند.

Q95: تعداد فیلمهایی را بیابید که در جشنوارهٔ فجر-دورهٔ هیجدهم شرکت کرده‌اند ولی در جشنوارهٔ خانهٔ سینما-دورهٔ چهارم شرکت نکرده‌اند.

Q96: لیستی از نام جشنواره‌های مختلف و تعداد فیلمهای شرکت کننده در هر یک از آنها تهیه کنید.

Q97: لیستی از نام جشنواره‌های مختلف و نام فیلم یا فیلمهایی که در هر یک از آنها بیشترین تعداد جایزه را به خود اختصاص داده‌اند تهیه کنید .

Q98: نام فیلمهایی را بیابید که در سه جشنوارهٔ مختلف برندهٔ جایزهٔ بهترین کارگردانی شده است.

Q99: تعداد فیلمها به کارگردانی آقای علی حاتمی‌را بیابید.

۳- فرض کنید پایگاه داده‌های سیستم یک بیمارستان از جداول ذیل تشکیل شده باشد:

dep (dep# , depname , headnurse#)

در این جدول اطلاعات مربوط به بخشهای مختلف بیمارستان ذخیره می‌شود:

dep#: شمارهٔ بخش

depname: نام بخش

headnurse#: شمارهٔ پرستار مسئول بخش(در هر بخش، یکی از پرستارها مسئول بخش است)

servant (ser# , sername , dep#)

در این جدول اطلاعات مربوط به خدمهٔ بیمارستان ذخیره می‌شود:

ser#: شمارهٔ کارگر

sername: نام کارگر

dep#: شمارهٔ بخشی که کارگر در آن کار می‌کند

doctor (doc# , docname , id , dep#)

در این جدول اطلاعات مربوط به پزشکان بیمارستان ذخیره می‌شود:

doc#: شمارهٔ پزشک

docname: نام پزشک

id: شمارهٔ نظام پزشکی

dep#: شمارهٔ بخشی که پزشک در آن کار می‌کند

profession (p# , pname)

در این جدول اطلاعات مربوط به تخصصهای پزشکی ذخیره می‌شود:

p#: کد تخصص(به هر تخصص پزشکی یک کد اختصاص یافته است)

pname: نام تخصص

Doc_Profession (doc# , p# , university)

در این جدول تخصصهای پزشکان مختلف مشخص می‌شود (ممکن است یک پزشک چند تخصص داشته باشد):

doc#: شمارهٔ پزشک

p#: شمارهٔ تخصص

university: نام دانشگاهی که پزشک مورد نظر تخصص مورد نظر را در آن کسب کرده است

nurse (nurse# , nname , degree , dep#)

در این جدول اطلاعات مربوط به پرستاران بیمارستان ذخیره می‌شود:

nurse#: شمارهٔ پرستار

nname: نام پرستار

degree: مدرک تحصیلی پرستار

dep#: شمارهٔ بخشی که پرستار در آن کار می‌کند

shift_nurse (date , shift , nurse#)

در این جدول مشخص می‌شود در هر شیفت از هر تاریخ چه پرستارانی کشیک بوده‌اند(در هر شیفت از هر روز چند پرستار در هر بخش حضور دارند):

date: تاریخ

shift: شمارهٔ شیفت(۱ یا ۲ یا ۳)

nurse#: شمارهٔ پرستار کشیک

shift_doctor (date , shift, doc#)

در این جدول مشخص می‌شود در هر شیفت از هر تاریخ کدام پزشکان کشیک بوده‌اند(در هر شیفت از هر روز حداقل ۱ پزشک در هر بخش حضور دارد):

date: تاریخ

shift: شمارهٔ شیفت(۱ یا ۲ یا ۳)

doc#: شمارهٔ پزشک کشیک

room (dep# , room# , bedCount)

در این جدول اطلاعات مربوط به اتاقهای هر بخش از بیمارستان ذخیره می‌شود(هر بخش تعدادی اتاق دارد):

dep#: شمارهٔ بخش

room#: شمارهٔ اتاق

bedCount: تعداد تخت در اتاق

bed (dep# , room# , bed# , cost)

در این جدول اطلاعات مربوط به تختهای هر اتاق از بیمارستان ذخیره می‌شود(هر اتاق تعدادی تخت دارد):

dep#: شمارهٔ بخش

room#: شمارهٔ اتاق

bed#: شمارهٔ تخت

cost: هزینهٔ تخت برای یک شب

patient (p# , pname , address , tel , assurance#)

در این جدول اطلاعات مربوط به بیماران ذخیره می‌شود:

p#: شمارهٔ بیمار

pname: نام بیمار

address: آدرس بیمار

tel: شماره تلفن بیمار

assurance#: کد نوع بیمه(هر بیمار تحت پوشش یک نوع بیمه است)

assurance (assurance# , aname)

در این جدول اطلاعات مربوط به انواع بیمه‌ها ذخیره می‌شود:

assurance#: کد نوع بیمه

aname: نام بیمه

hospitalization(fac#,p#,startDate,endDate,dep#,room#,bed#,doc#,illness)

در این جدول اطلاعات مربوط به اقامت بیماران در بیمارستان ذخیره می‌شود(هر بیمار ممکن است چند بار در بیمارستان بستری شود. هر بیمار در هر دورهٔ اقامت یک تخت دارد و تحت مراقبت یک پزشک است. در هر دورهٔ اقامت برای بیمار یک صورتحساب صادر می‌شود):

fac#: شمارهٔ صورتحساب(هر صورتحساب یک شمارهٔ منحصر بفرد دارد و تنها مربوط به یک دورهٔ اقامت یک بیمار است)

p#: شمارهٔ بیمار

startDate: تاریخ بستری شدن در بیمارستان

endDate: تاریخ مرخص شدن از بیمارستان

dep#: شمارهٔ بخشی که بیمار در آن بستری شده است

room#: شمارهٔ اتاقی که بیمار در آن بستری شده است

bed#: شمارهٔ تخت بیمار

doc#: شمارهٔ پزشک معالج

illness: بیماری یا علت بستری شدن

Factor_item (fac# , item# , qty)

در این جدول ریز اقلام صورتحسابها ذخیره می‌شود (هر صورتحساب، شامل هزینه‌های کلیهٔ داروها و وسایل و تجهیزات مورد استفاده برای تشخیص بیماری و یا درمان بیمار در طول مدت زمان اقامت وی در بیمارستان است):

fac#: شمارهٔ صورتحساب

item#: کد دارو یا امکانات استفاده شده

qty: مقدار یا تعداد دفعات استفاده از دارو یا امکانات مورد نظر(مثلاً ممکن است بیمار سه بار آندوسکوپی شده باشد و یا ۱۰ سرم قندی نمکی به وی تزریق شده باشد)

item (item# , iname , fee)

در این جدول اطلاعات مربوط به داروها یا امکانات و تجهیزات مورد استفاده در بیمارستان ذخیره می‌شود:

item#: کد دارو یا امکانات مورد استفاده در بیمارستان(به هر یک از داروها یا عملیات مورد استفاده برای تشخیص بیماری و یا مداوای بیماران یک کد اختصاص یافته است.)

iname: نام دارو یا امکانات

fee: هزینهٔ هر واحد

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q100: تعداد بخشهای بیمارستان را بیابید.

Q101: نام مسئول بخش جراحی زنان را به دست آورید.

Q102: تعداد خدمهٔ بخش جراحی زنان را به دست آورید.

Q103: تعداد بیمارانی را بیابید که در تاریخ ۸۳/۶/۴ در بخش جراحی زنان بستری شده‌اند.

Q104: بیمار شمارهٔ ۱۰۰ چند بار در بیمارستان بستری شده است؟

Q105: نام بخشهایی را بیابید که بیمار شمارهٔ ۱۰۰ در آنها بستری بوده است.

Q106: بیمار شمارهٔ ۱۰۰ چند بار در بخش CCU بستری شده است؟

Q107: نام آخرین پزشک معالج بیمار شمارهٔ ۱۰۰ را به دست آورید.

Q108: کل هزینهٔ صورتحساب شمارهٔ ۲۰۰۱ را به دست آورید.

Q109: شماره وتاریخ و هز ینهٔ کل هر یک از صورتحسابهای مربوط به بیمار شمارهٔ ۲۰۰ را به دست آورید.

Q110: لیستی ازنام کلیهٔ بیماران پزشک شمارهٔ ۱۲ که پس از تاریخ ۸۳/۴/۴ در بیمارستان بستری شده‌اند تهیه کنید.

Q111: تعداد بیمارانی را بیابید که پس از تاریخ ۸۳/۴/۴ در بخش جراحی زنان بستری شده‌اند.

Q112: لیستی از نام بیمارانی که پس از تاریخ ۸۳/۴/۴ در بخش جراحی زنان بستری شده‌اند تهیه کنید.

Q113: لیستی از نام پزشکانی که از تاریخ ۸۳/۴/۴ تا کنون بیش از ۲۰۰ بیمار داشته‌اند تهیه کنید.

Q114: لیستی از نام پزشکانی که تخصص قلب دارند تهیه کنید.

Q115: لیستی از نام پزشکانی که تخصص مغز و اعصاب دارند و از تاریخ ۸۳/۴/۴ تا کنون بیش از ۱۰۰ بیمار داشته‌اند تهیه کنید.

Q116: لیستی از نام بیمارانی که پس از تاریخ ۸۳/۴/۴ بعلت شکستگی استخوان در بیمارستان بستری شده‌اند تهیه کنید.

Q117: برای بیمار شمارهٔ ۲۰۰۱ در آخرین بار بستری شدن چه داروها یا درمانهایی تجویز شده است؟

Q118: نام بیمارانی را بیابید که پس از تاریخ ۸۳/۴/۴ مورد عمل جراحی قلب قرار گرفته‌اند.

Q119: نام پرستارانی را بیابید که درشیفت سوم ۸۳/۶/۴ در بخش جراحی زنان کشیک بوده‌اند.

Q120: تعداد بیمارانی را بیابید که تحت پوشش بیمهٔ تامین اجتماعی هستند و پس از تاریخ ۸۳/۴/۴ در بخش CCU بستری شده‌اند.

Q121: نام بیمارانی را بیابید که پس از تاریخ ۸۳/۴/۴ بیش از سه بار در بخش CCU بستری شده‌اند.

Q122: در بخش قلب چند اتاق سه تخته وجود دارد؟

Q123: لیستی از نام پزشکان بخش قلب و تعداد کل بیماران آنها تا این لحظه تهیه کنید.

٤) سیستم حقوق و مدیریت پروژه در یک سازمان را در نظر بگیرید. در این سازمان کارمندان به دو دستهٔ رسمی و قراردادی تقسیم می‌شوند. به کارمندان رسمی‌حقوق ثابت تعلق می‌گیرد ولی حقوق کارمندان قراردادی بر اساس مبلغ ساعتی توافق شده در آخرین قرارداد منعقد شده با هر یک از آنها محاسبه می‌شود. در هر لحظه چندین پروژه در سازمان در حال اجرا است. روی هر پروژه چندین کارمند کار می‌کنند و ممکن است هر کارمند روی چند پروژه کار کند. در هر پروژه یکی از کارمندان مدیر است. مدیر، پروژه را به تعدادی وظایف[1] کوچکتر تقسیم کرده و هر یک از آنها را به تعدادی از کارمندانی که روی پروژه کار می‌کنند محول می‌کند. هر کارمند موظف است هر روز برای هر یک از وظایفی که در حال کار روی آنهاست یک گزارش پیشرفت کار جداگانه پر کند و مشخص کند برای انجام قسمت انجام شده از وظیفهٔ مورد نظر چه مقدار

1 - Task

زمان مصرف کرده است. حقوق کارمندان قراردادی بر اساس کل ساعاتی که در گزارشات پیشرفت کار ذکر کرده‌اند محاسبه می‌شود. فرض کنید پایگاه داده‌های این سیستم شامل جداول ذیل باشد:

emp (emp# , ename , tel , address)

در این جدول مشخصات کلی همهٔ کارمندان ذخیره می‌شود:

#emp: شمارهٔ کارمند

ename: نام کارمند

tel: شماره تلفن

address: آدرس

contract(con# , emp#, description, startDate, endDate, PayPerHour)

در این جدول مشخصات قراردادهای بسته شده با کارمندان قراردادی ذخیره می‌شود:

#con: شمارهٔ قرارداد

#emp: شمارهٔ کارمند

description: شرح قرارداد

startDate: تاریخ شروع قرارداد

endDate: تاریخ خاتمهٔ قرارداد

payPerHour: حقوق پرداختی توافق شده برای هر ساعت

offEmp (emp# , startDate , pay)

در این جدول مشخصات خاص کارمندان رسمی‌ذخیره می‌شود:

#emp: شمارهٔ کارمند

startDate: تاریخ استخدام

pay: حقوق ثابت

profession (p# , pname)

#p: شمارهٔ تخصص

pname: نام تخصص

Emp_profession (emp# , p# , degree)

در این جدول مشخص می‌شود هر کارمند چه تخصصی دارد(ممکن است یک کارمند چند تخصص داشته باشد):

emp#: شمارهٔ کارمند

p#: شمارهٔ تخصص

degree: سطح مهارت کارمند مورد نظر در تخصص مورد نظر

project (proje# , pname , headEmp# , status)

در این جدول اطلاعات پروژه‌ها ذخیره می‌شود:

proje#: شمارهٔ پروژه

pname: نام پروژه

headEmp#: شمارهٔ مدیر پروژه

status: وضعیت انجام پروژه(در حال اجرا، در حال پشتیبانی،...)

task (proje# , task# , tname , startDate , endDate)

در این جدول اطلاعات وظایف ذخیره می‌شود:

proje#: شمارهٔ پروژه

task#: شمارهٔ وظیفه

tname: عنوان وظیفه

startDate: تاریخ شروع وظیفه

endDate: تاریخ خاتمهٔ وظیفه

Emp_Task (proje# , task# , emp#)

در این جدول مشخص می‌شود کدام کارمندان روی کدام وظایف کار می‌کنند:

proje#: شمارهٔ پروژه

task#: شمارهٔ وظیفه

emp#: شمارهٔ کارمندی که در وظیفه مشارکت دارد

report (report# , emp# , proje# , task# , rdate , description , hours)

در این جدول اطلاعات وارد شده در گزارشات پیشرفت کار ذخیره می‌شود:

report#: شمارهٔ گزارش

#emp: شمارهٔ کارمند گزارش دهنده

#proje: شمارهٔ پروژه

#task: شمارهٔ وظیفه‌ای که کارمند در بارهٔ آن گزارش می‌دهد

rdate: تاریخ گزارش

description: شرح گزارش

hours: تعداد ساعات صرف شده برای انجام کار مذکور در گزارش

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q124: نام کارمندان رسمی را بیابید که در طراحی سایت دانشگاه تهران مشارکت دارند.

Q125: نام پروژه‌هایی را بیابید که کارمند شمارهٔ ۱۰۰ در آنها مشارکت داشته است.

Q126: تعداد پروژه‌هایی را بیابید که کارمند شمارهٔ ۱۰۰ در آنها مشارکت داشته است.

Q127: لیستی از شمارهٔ کارمندان و تعداد کل پروژه‌هایی که هر یک در آنها مشارکت داشته‌اند تهیه کنید.

Q128: لیستی از شمارهٔ پروژه‌های "طراحی وب سایت" و تعداد کل کارمندانی که در هر یک از آنها مشارکت داشته‌اند تهیه کنید.

Q129: عنوان وظایف مربوط به پروژهٔ شمارهٔ ۱۰۰ و تعداد کل کارمندانی را که روی هر یک از آنها کار می‌کنند به دست آورید.

Q130: تعداد گزارشات پیشرفت کار کارمند شمارهٔ ۱۰۰ را به دست آورید.

Q131: لیستی ازترکیبات شمارهٔ پروژه- شمارهٔ وظیفه که کارمند شمارهٔ ۱۰۰ در آنها مشارکت داشته است و تعداد کل گزارشات پیشرفت کاری که این کارمند برای هر یک از آنها پر کرده است، تهیه کنید.

Q132: لیستی از شمارهٔ کارمندانی که جمعاً بیش از ۱۰۰۰ ساعت برای شرکت کار کرده‌اند تهیه کنید.

Q133: نام کارمندانی را بیابید که تا کنون بیش از ۴ قرارداد با آنها تنظیم شده است.

Q134: نام کارمندانی را بیابید که در بیش از ۵ پروژهٔ مختلف همکاری کرده‌اند.

Q135: نام کارمندانی را بیابید که در زمینهٔ طراحی شی گرا تخصص دارند و در پروژهٔ "طراحی وب سایت انرژی اتمی" همکاری داشته‌اند.

Q136: تعداد کل ساعاتی را بیابید که کارمند شمارهٔ ۱۰۰ برای وظیفهٔ "طراحی پایگاه داده‌های وب سایت انرژی اتمی" صرف کرده است.

Q137: تعداد کل ساعاتی را بیابید که برای "طراحی پایگاه داده‌های وب سایت انرژی اتمی" صرف شده است.

Q138: کل مبلغ پرداخت شده به کارمندان قراردادی برای پروژهٔ "طراحی وب سایت انرژی اتمی" را به دست آورید.

Q139: کل مبلغ پرداختی به کارمند شمارهٔ ۱۰۰ (کارمند قراردادی) برای طراحی پایگاه داده‌های وب سایت انرژی اتمی را به دست آورید (حاصلضرب کل ساعات صرف شده توسط این کارمند روی وظیفهٔ مورد نظر در حقوق ساعتی توافق شده در آخرین قرارداد این کارمند).

۵- فرض کنید پایگاه داده‌ها در سیستم یک ISP [1] شامل جداول ذیل باشد:

account (<u>userId</u> , password , credit , cost)

در این جدول اطلاعات کارتهای اینترنت این ISP ذخیره می‌شود:

userId: شناسهٔ کاربر

password: کلمهٔ عبور

credit: کل اعتبار کارت (به دقیقه)

cost: قیمت کارت

connection (<u>userId , cdate , startTime</u> , endTime)

در این جدول اطلاعات مربوط به اتصال کاربران به اینترنت ذخیره می‌شود:

userId: شناسه

---

1 -Internet Service Provider

cdate: تاریخ اتصال

startTime: زمان شروع اتصال

endTime: زمان قطع اتصال

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q140: تعداد کل کارتهایی را بیابید که اعتبار آنها ۴۰۰۰ تومان بوده است.

Q141: شناسهٔ کاربرانی را بیابید که همیشه تنها پس از ساعت ۲ بامداد و قبل از ساعت ۷ بامداد به اینترنت وصل بوده‌اند.

Q142: تعداد کاربرانی را بیابید که در تاریخ ۸۳/۶/۴ به اینترنت متصل شده‌اند.

Q143: شناسهٔ کاربرانی را بیابید که در تاریخ ۸۳/۶/۴ به اینترنت متصل شده‌اند.

Q144: لیستی از شناسهٔ کاربرانی که در تاریخ ۸۳/۶/۶ بیش از ۵ بار به اینترنت متصل شده‌اند تهیه کنید.

Q145: لیستی از شناسهٔ کاربرانی که در تاریخ ۸۳/۶/۶ به اینترنت متصل نشده‌اند تهیه کنید.

Q146: بیشترین تعداد اتصالات در چه تاریخ یا تاریخهایی صورت گرفته است؟

Q147: تاریخ یا تاریخهایی را بیابید که در آنها تعداد کاربرانی که به اینترنت متصل شده‌اند حداکثر بوده است.

Q148: کاربر با شناسهٔ iran2003 جمعاً چند دقیقه به اینترنت متصل بوده است؟

Q149: کاربر با شناسهٔ iran2003 مجموعاً چند بار به اینترنت متصل شده است؟

Q150: لیستی از تاریخهای مختلف و تعداد کل اتصالات در هر یک از آنها تهیه کنید.

Q151: شناسهٔ کارتهایی را بیابید که تا کنون مصرف نشده‌اند(اتصالی نداشته‌اند).

۶- یک وب سایت برای برگزاری آزمونهای مختلف را درنظر بگیرید. اعضاء این سایت می‌توانند با وارد کردن نام کاربری و کلمهٔ عبور خود وارد سایت شده، آزمون مورد نظر خود را انتخاب کرده، به سوالات چند گزینه‌ای آن پاسخ دهند و امتیاز بگیرند. فرض کنید پایگاه داده‌های این سیستم شامل جداول ذیل باشد:

user (userid , password)

در این جدول اطلاعات کاربران ذخیره می‌شود:

userid: نام کاربری

password: کلمهٔ عبور

test (test# , title , creditPerTrue , negPerFalse , total , neededCredit)

در این جدول اطلاعات کلی آزمونهای مختلف ذخیره می‌شود:

test#: شمارهٔ آزمون

title: عنوان آزمون

creditPerTrue: امتیاز به ازاء هر جواب صحیح

negPerFalse: مقدار امتیازی که باید به ازاء هر جواب نادرست کسر شود

total: کل امتیاز آزمون

neededCredit: حداقل امتیاز مورد نیاز برای قبولی در آزمون

question (test# , q# , qtext , trueChoice#)

در این جدول مشخصات سوالات آزمونهای مختلف ذخیره می‌شود:

test#: شمارهٔ آزمون

q#: شمارهٔ سوال

qtext: متن سوال

trueChoice#: شمارهٔ گزینهٔ درست(در هر یک از سوالها تنها یکی از گزینه‌ها صحیح است.)

choice(test# , q# , choice# , ctext)

در این جدول مشخصات گزینه های کلیهٔ سوالات ذخیره می‌شود:

test#: شمارهٔ آزمون

q#: شمارهٔ سوال

choice#: شمارهٔ گزینه

ctext: متن گزینه

user_test (userid , test# , tdate)

در این جدول مشخص می‌شود کدام کاربر در کدام آزمون شرکت کرده است. هر کاربر تنها یک بار می‌تواند در هر آزمون شرکت کند:

userid: نام کاربری

test#: شمارهٔ آزمون

tdate: تاریخ شرکت کاربر در آزمون

user_test_question (userid , test# , q# , userChoice#)

در این جدول مشخص می‌شود هر کاربر برای هر سوال از یک آزمون کدام گزینه را انتخاب کرده است(برای هر سوال کاربر مجاز است تنها یکی از گزینه‌ها را انتخاب کند):

userid: نام کاربری

test#: شمارهٔ آزمون

q#: شمارهٔ سوال

userChoice#: شمارهٔ گزینه‌ای که کاربر برای این سوال انتخاب کرده است

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q152: نام کاربرانی را بیابید که در آزمون asp.net شرکت کرده‌اند.

Q153: کل امتیاز کسب شده توسط کاربر با نام کاربری aftab در آزمون ویژوال بیسیک پیشرفته چقدر بوده است؟

Q154: لیستی از نام کاربرانی که در آزمون طراحی شبکه شرکت کرده‌اند و کل امتیاز کسب شده توسط هر یک از آنها تهیه کنید.

Q155: لیستی از نام کاربرانی که در آزمون طراحی شبکه قبول شده‌اند تهیه کنید.

Q156: لیستی از نام کاربرانی که هم در آزمون طراحی شبکه و هم در آزمون مهندسی اینترنت قبول شده‌اند تهیه کنید.

Q157: تعداد کاربرانی را به دست آورید که به سوال ۳ از آزمون شمارهٔ ۳ پاسخ درست داده‌اند.

Q158: نام کاربرانی را بیابید که به ۶۰ درصد از سوالات آزمون مهندسی اینترنت پاسخ صحیح داده‌اند.

Q159: نام کاربرانی را بیابید که در ۵ آزمون مختلف شرکت کرده‌اند.

Q160: نام کاربرانی را بیابید که در همهٔ آزمونها شرکت کرده‌اند.

Q161: نام کاربرانی را بیابید که در همهٔ آزمونها قبول شده‌اند.

Q162: نام کاربرانی را بیابید که در هیچ یک از آزمونها شرکت نکرده‌اند.

Q163: نام کاربرانی را بیابید که در هیچ یک از آزمونها قبول نشده‌اند.

۷- فرض کنید پایگاه داده‌های یک سایت اینترنتی فروش اسباب بازی از جداول ذیل تشکیل شده باشد:

company (company# , cname)

در این جدول اطلاعات شرکتهای سازندهٔ اسباب بازی ذخیره می‌شود:

company#: شمارهٔ شرکت سازنده

cname: نام شرکت سازنده

toy (toy# , name , company# , fee , count)

در این جدول اطلاعات اسباب بازیهای مختلف ذخیره می‌شود:

toy#: شمارهٔ اسباب بازی

name: نام اسباب بازی

company#: شمارهٔ شرکت سازنده

fee: قیمت واحد

count: تعداد موجود در انبار

age (age# , startAge , endAge)

در این جدول به هر گروه سنی یک کد اختصاص یافته است مثلاً گروه سنی ب ، سنین۳ تا ۵ سال را شامل می‌شود:

age#: کد گروه سنی

startAge: سن شروع

endAge: سن پایان

toy_age (toy# , age#)

در این جدول مشخص می‌شود هر اسباب بازی برای کدام گروههای سنی مناسب است (ممکن است یک اسباب بازی برای چند گروه سنی مناسب باشد):

toy#: شمارهٔ اسباب بازی

age#: کد گروه سنی

toy_type (toy# , type)

در این جدول مشخص می‌شود هر اسباب بازی در چه انواعی طبقه بندی می‌شود(ممکن است یک اسباب بازی در انواع مختلف طبقه بندی شود مثلاً هم فکری و هم ورزشی باشد):

toy#: شمارهٔ اسباب بازی

type: نوع اسباب بازی (ورزشی، فکری...)

factor (fac# , fdate , address , tel , cc# , ccType)

در این جدول اطلاعات سربرگ صورتحسابهای صادر شده برای مشتریان ذخیره می‌شود:

fac#: شمارهٔ صورتحساب

fdate: تاریخ صدور

address: آدرس مشتری

tel: شماره تلفن مشتری

cc#: شمارهٔ کارت اعتباری

ccType: نوع کارت اعتباری

fac_toy (fac# , toy# , qty)

در این جدول ریز اقلام صورتحسابها ذخیره می‌شود:

fac#: شمارهٔ صورتحساب

toy#: کد اسباب بازی

qty: تعداد خرید

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q164: نام اسباب بازیهایی را بیابید که در میان اسباب بازیها بیشترین تعداد فروش را داشته‌اند.

Q165: نام اسباب بازیهایی را بیابید که بیشترین فروش ریالی را داشته‌اند.

Q166: نام اسباب بازیهایی را بیابید که هم ورزشی و هم فکری هستند.

Q167: نام اسباب بازیهایی را بیابید که برای گروه سنی الف مناسبند.

Q168: نام اسباب بازیهایی را بیابید که برای گروه سنی الف مناسب نیستند.

Q169: نام اسباب بازیهایی را بیابید که قیمت آنها بین ۳۰۰۰ و ۵۰۰۰ تومان است.

Q170: نام اسباب بازیهایی را بیابید که برای سه گروه سنی مختلف مناسب هستند.

Q171: نام اسباب بازیهایی را بیابید که برای همهٔ گروههای سنی مناسب هستند.

Q172: نام اسباب بازیهایی را بیابید که برای گروه سنی الف مناسبند ولی برای گروه سنی ب مناسب نیستند.

Q173: لیستی از نام اسباب بازیها و تعداد کل فروش آنها پس از تاریخ ۸۳/۴/۴ به دست آورید.

Q174: نوع کارت اعتباری را بیابید که بیشترین مورد استفاده را در میان کاربران سایت داشته است.

۸- فرض کنید پایگاه داده‌های یک وب سایت اطلاع رسانی در مورد بازیهای جام جهانی فوتبال شامل جداول ذیل باشد:

worldcup (<u>cup#</u> , startDate , endDate , place , team1# , team2# , team3#)

در این جدول، اطلاعات مربوط به جامهای مختلف ذخیره می‌شود:

cup#: شمارهٔ جام جهانی(سال برگزاری جام جهانی)

startDate: تاریخ شروع

endDate: تاریخ خاتمه

place: محل برگزاری

team1#: تیم قهرمان اول

team2#: تیم قهرمان دوم

team3#: تیم قهرمان سوم

team (<u>team#</u> , country , history)

در این جدول، اطلاعات مربوط به تیمهای کشورهای مختلف ذخیره می‌شود:

team#: شمارهٔ تیم(به تیم هر کشور یک شمارهٔ منحصر بفرد داده شده است)

country: کشور

history: تاریخچهٔ تیم

cup_team (cup# , team# , color)

در این جدول مشخص می‌شود هر تیم در چه جامهایی شرکت داشته است:

cup#: شمارهٔ جام

team#: شمارهٔ تیم

color: رنگ پیراهن

person (id , name , biography)

در این جدول اطلاعات مربوط به بازیکنان، مربیان و داوران ذخیره می‌شود. (ممکن است یک شخص در جامهای مختلف با سمتهای مختلف حضور یابد مثلاً در یک جام بازیکن و در جام دیگر مربی باشد):

id: شناسهٔ شخص (به هر شخص، فارغ از سمت یا سمتهایی که در جامهای مختلف داشته است یک شناسهٔ منحصر بفرد اختصاص یافته است)

name: نام شخص

biography: زندگینامه

team_player (cup# , team# , id , role , number)

در این جدول، بازیکنان تیمها در هریک از جامها مشخص می‌شوند(در هر جام، هر تیم حداقل ۲۰ بازیکن دارد ولی هر بازیکن تنها در یک تیم بازی می‌کند) :

cup#: شمارهٔ جام

team#: شمارهٔ تیم

id: شناسهٔ شخص(بازیکن)

role: نقش بازیکن(دفاع ، ذخیره، حمله و ...)

number: شمارهٔ پیراهن

تذکر: ترکیب cup#+team#+number نیز می‌تواند به عنوان کلید اصلی جدول معرفی شود.

team_coatch (cup# , team# , id , role)

در این جدول، مربیان هر تیم در هریک از جامها مشخص می‌شوند(در هر جام، هر تیم حداقل یک مربی دارد ولی هر مربی تنها مربی‌گری یک تیم را بر عهده دارد) :

cup#: شمارهٔ جام

team#: شمارهٔ تیم

id: شناسهٔ شخص(مربی)

role: نقش مربی (سرمربی، کمک مربی و ...)

play (cup# , play# , pdate , team1# , team2# , result)

در این جدول، اطلاعات کلیهٔ بازیهای جامهای مختلف ذخیره می‌شود:

cup#: شمارهٔ جام

play#: شمارهٔ بازی

pdate: تاریخ بازی

team1#: شمارهٔ تیم ۱

team2#: شمارهٔ تیم ۲

result: نتیجهٔ بازی

play_referee (cup# , play# , id , role)

در این جدول، داوران بازیها مشخص می‌شوند(هر بازی چند داور دارد):

cup#: شمارهٔ جام

play#: شمارهٔ بازی

id: شناسهٔ شخص(داور)

role: نقش داور (داور یا کمک داور یا ...)

goals (cup# , play#, goal# , id , time)

در این جدول، اطلاعات گلهای زده شده در بازیهای مختلف ذخیره می‌شود:

cup#: شمارهٔ جام

play#: شمارهٔ بازی

goal#: شمارهٔ گل زده شده

id: شناسهٔ بازیکنی که گل را به ثمر رسانده است

time: دقیقه‌ای که گل زده شده است

penalty (cup# , play# , penalty# , id , type , time , punishment)

در این جدول، اطلاعات خطاهای صورت گرفته در بازیهای مختلف ذخیره می‌شود:

#cup: شمارهٔ جام

#play: شمارهٔ بازی

#penalty: شمارهٔ خطا(به هر یک از خطاهای رخ داده در بازی یک شماره داده شده است)

id: شناسهٔ بازیکنی که خطا را مرتکب شده است

type: نوع خطا(برخورد بدنی، تماس دست با توپ و ...)

time: دقیقه‌ای که در آن خطا صورت گرفته است

punishment: مجازات (کارت قرمز، کارت زرد، ...)

پرس و جوهای ذیل را به زبان SQL بنویسید:

Q175: نام بازیکنان تیم آلمان در جام ۱۹۹۸ را بیابید.

Q176: نام قهرمان اول جام ۱۹۹۸ را بیابید.

Q177: تیم ایران درچند جام جهانی شرکت داشته است؟

Q178: لیستی شامل نام تیمها و تعداد کل بازیهای آنها در جام ۱۹۹۸ تهیه کنید.

Q179: نام تیم یا تیمهایی را بیابید که در جام ۱۹۹۸ بیشترین گل را زده‌اند.

Q180: نام بازیکن یا بازیکنانی را بیابید که در جام ۱۹۹۸ بیشترین گل را زده‌اند.

Q181: در جام ۱۹۹۸ کلاً چه تعداد گل زده شده است؟

Q182: نام تیمهایی را بیابید که در کلیهٔ جامها شرکت داشته‌اند.

Q183: نام تیمهایی را بیابید که بیش از ۲ بار قهرمان اول بازیهای جام جهانی بوده‌اند.

Q184: نام سرمربی تیم آلمان در جام ۱۹۹۸ را به دست آورید.

Q185: نام اشخاصی را بیابید که هم به عنوان بازیکن و هم به عنوان مربی در جامهای مختلف شرکت داشته‌اند.

Q186: تعداد بازیهایی را بیابید که شخص شمارهٔ ۱۰۰ داوری آنها را بر عهده داشته است.

Q187: لیستی از شمارهٔ جامهایی که شخص شمارهٔ ۱۰۰ در آنها حضور داشته است تهیه کنید.

Q188: تعداد بازیهای انجام شده در جام ۱۹۹۸ را به دست آورید.

Q189: نام تیم یا تیمهایی را بیابید که بازیکنان آنها در جام ۱۹۹۸ بیشترین کارت قرمز را گرفته‌اند.

Q190: نام تیم یا تیمهایی را بیابید که بازیکنان آنها در جام ۱۹۹۸ بیشترین تعداد برخورد بدنی با بازیکنان تیم مقابل را داشته‌اند.

۵

نرمال‌سازی

برای کاهش افزونگی و ناهنجاریهای حذف، درج و اصلاح لازم است جداول پایگاه داده‌ها نرمال باشند. از نظر درجهٔ نرمال بودن جداول را می‌توان به ترتیب به ۶ گروه تقسیم کرد:

۱- جداول آنرمال(کمترین درجهٔ نرمالی)

۲- جداول نرمال ۱ یا 1NF یا FNF [1]

۳- جداول نرمال ۲ یا 2NF یا SNF [2]

٤- جداول نرمال ۳ یا 3NF و جداول نرمال BCNF [3]

۵- جداول نرمال ۴

۶- جداول نرمال ۵ یا PJNF [4](بالاترین درجهٔ نرمالی)

در این فصل مراحل نرمال سازی جداول را مورد بررسی قرار می‌دهیم.

---

[1] - First Normal Form

[2] - Second Normal Form

[3] - Boyce-Codd Normal Form

[4]- Projection-Join Normal Form

۵-۱) جداول آنرمال

جدول student را که شامل شماره دانشجویی، نام و شماره تلفنهای تماس دانشجو است در نظر بگیرید:

Student

st#	name	telephones
7801	آرش	0311-6262778 0913-311-5234
7902	عسل	021-2956677 0912-314-4532

همانطور که مشاهده می‌کنید در برخورد هر سطر جدول با ستون telephones این احتمال وجود دارد که به جای یک شماره تلفن، مجموعه‌ای از شماره تلفنها یا به عبارت بهتر یک گروه اطلاع تکرار شونده[1] از شماره تلفنها وجود داشته باشد بنابراین، جدول student آنرمال است. جدول آنرمال به جدولی اطلاق می‌شود که گروه اطلاع تکرار شونده دارد یا به عبارت دیگر، در برخورد هر سطر با هر ستون آن به جای یک مقدار اتمی و تجزیه ناپذیر، مجموعه‌ای از مقادیر وجود دارد. در نمودار وابستگی، برای نمایش گروه اطلاع تکرار شونده از یک خط در بالای ویژگی استفاده می‌شود:

Student (st # , name , telephones)

۵-۲) جداول نرمال ۱

مهمترین عیب یک جدول آنرمال این است که برای هر یک از عملیات درج، حذف و اضافه به دو دسته عملگر احتیاج داریم: یکی در سطح تاپل و دیگری در سطح مجموعه. مثلاً در جدول student اگر بخواهیم اطلاعات دانشجوی جدیدی را درج کنیم به عملگر درج در سطح تاپل و اگر بخواهیم برای یک دانشجو شماره تلفن جدیدی درج کنیم به عملگر درج در سطح مجموعه نیاز خواهیم داشت. برای رفع این معایب، جداول آنرمال را به نرمال ۱ تبدیل می‌کنیم.

---

1 - Repeating Group

تذکر: اصولاً در مدل رابطه‌ای یک جدول حداقل باید نرمال ۱ باشد.

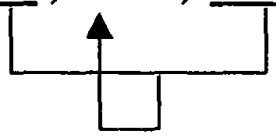
یک جدول در حالت نرمال یک است اگر هیچ گروه اطلاع تکرار شونده در آن وجود نداشته باشد و یا به عبارتی دیگر، در برخورد هر سطر با هر ستون جدول به یک مقدار تجزیه ناپذیر برسیم.

مثال ۱: برای آنکه جدول آنرمال student را به نرمال ۱ تبدیل کنیم، لازم است مقادیر ویژگیهای #st و name را به ازاء هر شماره تلفن تکرار کنیم:

st#	name	telephone
7801	آرش	0311-6262778
7801	آرش	0913-311-5234
7902	عسل	021-2956677
7902	عسل	0912-314-4532

با تبدیل جدول student به این فرم، در برخورد هر سطر با ستون telephone تنها به یک شماره تلفن می‌رسیم ولی در این حالت، #st به تنهایی نمی‌تواند کلید اصلی باشد چون برای آن مقادیر تکراری وجود دارند. برای رفع این مشکل، کلید اصلی را به st#+telephone تغییر می‌دهیم:

Student (st # , name , telephone)

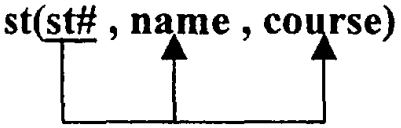


مثال ۲: جدول st را در نظر بگیرید:

st

st #	name	course				
7801	علی	1400	پایگاه داده	3	20	79-2
		1500	ریاضی ۱	3	10	80-1
		1600	تجزیه و تحلیل	3	20	80-1
7902	آرش	1400	پایگاه داده	3	7	80-1
		1700	تربیت بدنی	1	20	80-1

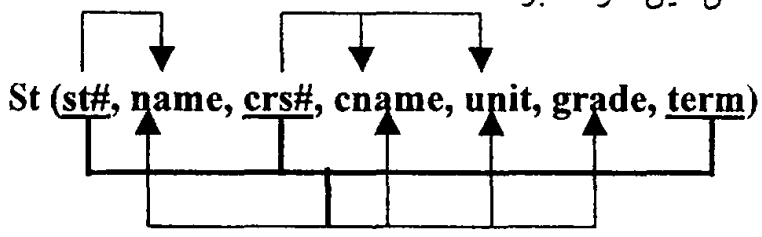
در ستون course از این جدول گروههای اطلاع تکرار شونده وجود دارند. این گروههای تکرار شونده هم بصورت سطری و هم بصورت ستونی قابل تجزیه‌اند. اطلاعات هر درس بصورت ستونی قابل تجزیه به شمارهٔ درس، نام درس، تعداد واحد درس، نمرهٔ دانشجو در درس و ترمی که دانشجو درس را گرفته است می‌باشد و از آنجا که ممکن است هر دانشجو چندین درس را گرفته باشد، این گروههای تکرار شونده از نظر سطری نیز قابل تجزیه هستند. نمودار وابستگی این جدول به شکل ذیل است:



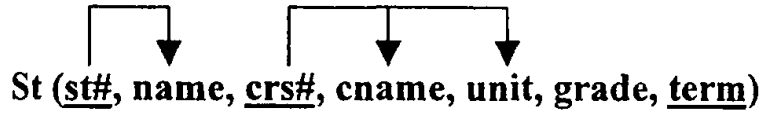
برای تبدیل این جدول به حالت نرمال ۱ باید آن را به فرم ذیل تبدیل کنیم:

st #	name	crs#	cname	unit	grade	term
7801	علی	1400	پایگاه داده	3	20	79-2
7801	علی	1500	ریاضی ۱	3	10	80-1
7801	علی	1600	تجزیه و تحلیل	3	20	80-1
7902	آرش	1400	پایگاه داده	3	7	80-1
7902	آرش	1700	تربیت بدنی	1	20	80-1

برای جلوگیری از وجود مقادیر تکراری باید کلید اصلی جدول را به st#+crs#+term تغییر دهیم. نمودار وابستگی این جدول به شکل ذیل خواهد بود:



و یا بصورت ساده‌تر:



۳-۵) جداول نرمال ۲

برخی از جداول در حالت نرمال ۱ هستند ولی هنوز ناهنجاری دارند مثلاً با آنکه جدول St به نرمال ۱ تبدیل شد دارای ناهنجاریهای ذیل است:

۱- ناهنجاری در حذف :

فرض کنید علی تنها دانشجویی باشد که درس ریاضی ۱ را گرفته است. اگر لازم باشد اطلاعات علی را حذف کنیم بطور ناخواسته اطلاعات ریاضی ۱ را نیز از دست می‌دهیم.

۲- ناهنجاری در درج :

فرض کنید بخواهیم اطلاعات دانشجوی جدیدی را که هنوز هیچ درسی را نگرفته است، درج کنیم. از آنجا که #course و term جزئی از کلید اصلی هستند و نمی‌توانند مقدار تهی داشته باشند انجام این عمل ممکن نیست.

۳- ناهنجاری در اصلاح :

فرض کنید بخواهیم تعداد واحد درس پایگاه داده را از ۳ به ۴ تغییر دهیم. در این صورت، برای حفظ سازگاری داده‌ها لازم است این تغییر را به دفعات مکرر و در تاپلهای مختلف اعمال کنیم (اصلاح منتشر شونده[1]).

برای رفع این معایب جداول نرمال ۱ را به جداول نرمال ۲ تبدیل می کنیم.

یک جدول در حالت نرمال ۲ است اگر :

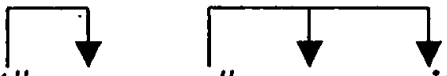
۱- نرمال یک باشد.

۲- در آن هیچ وابستگی جزئی به کلید اصلی وجود نداشته باشد. به عبارت دیگر، هیچ ویژگی جدول تنها به قسمتی از کلید اصلی وابستگی نداشته باشد.

---

1 - Propagating Update

مثال ۱:جدول st که در مثال ۲ تبدیل به نرمال ۱ شد، نرمال ۲ نیست چرا که کلید اصلی این جدول st#+crs#+term است در حالیکه تنها با داشتن st# می‌توان به name رسید و یا تنها با داشتن crs# می‌توان به cname و unit رسید:



St (st#, name, crs#, cname, unit, grade, term)

برای تبدیل یک جدول نرمال ۱ به نرمال ۲ مراحل ذیل را دنبال کنید:

۱- کلیهٔ ترکیبات ممکن میان اجزاء کلید اصلی را به ترتیب در سطرهای مجزا بنویسید(اول ترکیبات یک جزئی، سپس ترکیبات دوجزئی، سپس ترکیبات سه جزئی و...):

st#

crs#

term

st#+crs#

st#+term

crs#+term

st#+crs#+term

۲- در کنار هر یک از ردیفها ویژگیهایی را که در تشکیل کلید اصلی نقشی ندارند و به ویژگیهای مورد نظر وابستگی تابعی دارند و در سطرهای بالاتر نیز نوشته نشده‌اند بنویسید. مثلاً با داشتن یک st# تنها به یک name می‌رسیم پس name به st# وابستگی تابعی دارد. name را در ردیف اول اضافه کرده و در ردیفهای بعدی آن را در نظر نمی‌گیریم:

st# , name

crs# , cname , unit

term

st#+crs#

st#+term

crs#+term

st#+crs#+term , grade

۳- کلیهٔ ردیفهایی را که هیچ ویژگی غیر کلیدی در آنها وجود ندارد و شامل هیچ اطلاعات مفیدی که در جداول دیگر نیز وجود ندارد نمی‌باشند حذف کرده، سایر سطرها را به جداول مجزا تبدیل کنید:

Student(st# , name)

Course(crs# , cname , unit)

SC(st# , crs# , term , grade)

به این ترتیب جدول st به جداول ذیل تبدیل خواهد شد:

student

st#	name
7801	علی
7902	آرش

course

crs#	cname	unit
1400	پایگاه داده	3
1500	ریاضی ۱	3
1600	تجزیه و تحلیل	3
1700	تربیت بدنی	1

SC

st#	crs#	term	grade
7801	1400	79-2	20
7801	1500	80-1	10
7801	1600	80-1	20
7902	1400	80-1	7
7902	1700	80-1	20

٤-٥) جداول نرمال ۳ و BCNF

١-٤-٥) جداول نرمال ۳

برخی از جداول در حالت نرمال ۲ هستند ولی هنوز مشکلاتی دارند. به عنوان مثال جدول Professor را که شامل شماره، نام، کد آخرین مدرک تحصیلی و نام آخرین مدرک تحصیلی استاد است در نظر بگیرید:

Professor

prof#	pname	lastDegree#	lastDegreeName
100	علی راد	2	کاردانی
101	آرش رضایی	2	کاردانی
102	عسل شاملو	4	فوق لیسانس

نمودار وابستگی این جدول به شکل ذیل است:

professor(prof# , pname , lastDegree#, lastDegreeName)

این جدول نرمال ۲ است چون نرمال ۱ است و از آنجا که کلید اصلی تنها یک جز دارد، بدون شک هیچ وابستگی جزئی در آن وجود ندارد. با این وجود، این جدول هنوز دارای ناهنجاریهای ذیل است:

۱- ناهنجاری حذف :

فرض کنید عسل شاملو تنها کسی باشد که مدرک فوق لیسانس دارد. در این صورت، اگر بخواهیم اطلاعات عسل را حذف کنیم اطلاعات مربوط به مدرک فوق لیسانس(کد مدرک و نام مدرک) را نیز از دست می‌دهیم.

۲- ناهنجاری در اصلاح :

اگر بخواهیم نام مدرک ۴ را از فوق لیسانس به کارشناسی ارشد تغییر دهیم، در این صورت مجبوریم این تغییر را در تاپلهای مختلف تکرار کنیم (اصلاح منتشر شونده)

برای رفع این معایب، جداول نرمال ۲ را به نرمال ۳ تبدیل می‌کنیم. یک جدول در حالت نرمال ۳ است اگر :

۱) نرمال ۲ باشد.

۲) هیچ وابستگی متعدی(وابستگی با واسطه)[1] در آن وجود نداشته باشد به عبارت دیگر، در آن هیچ ویژگی غیر کلیدی به ویژگی غیر کلیدی دیگر وابستگی تابعی نداشته باشد.

جدول professor نرمال ۳ نیست چون:

lastDegree#→lastDegreeName

برای تبدیل جداول نرمال ۲ به نرمال ۳ مراحل ذیل را دنبال کنید:

۱- ویژگیهایی را که در وابستگی متعدی شرکت دارند در یک جدول مجزا قرار داده، ویژگیهایی را که در طرف چپ این وابستگی قرار دارند به عنوان کلید اصلی معرفی کنید.

Degree(lastDegree# , lastDegreeName)

۲- بقیهٔ ویژگیهای جدول اولیه را در جدول مجزای دیگری قرار داده(#prof و pname)، ویژگیهای تشکیل دهندهٔ کلید اصلی در جدول اول(#lastDegree) را به آنها اضافه کنید.

Prof(prof# , pname , lastDegree#)

Degree

lastDegree#	lastDegreeName
2	کاردانی
4	فوق لیسانس

Prof

prof#	pname	lastDegree#
100	علی راد	2
101	آرش رضایی	2
102	عسل شاملو	4

1 - Transitive Dependence

۲-٤-٥) جداول نرمال BCNF

برخی از جداول در حالت نرمال ۳ هستند ولی هنوز ناهنجاری دارند.

به عنوان مثال دانشگاهی با قوانین ذیل را در نظر بگیرید:

۱- هر دانشجو می‌تواند در چند رشتهٔ تحصیلی تحصیل کند ولی در هر یک از رشته‌های تحصیلی تنها یک استاد راهنما دارد.

۲- هر استاد تنها می‌تواند در یک رشتهٔ تحصیلی تدریس کند ولی در هر رشتهٔ تحصیلی چندین استاد وجود دارند.

جدول Project را که شامل شماره دانشجویی، رشتهٔ تحصیلی و استاد راهنمای دانشجو در رشتهٔ مربوطه است در نظر بگیرید:

Project

st#	field	tutor
7801	مهندسی کامپیوتر	مجید رضایی
7801	ریاضی محض	آرش ریاضیدان
7801	هنر	گلناز هنردوست
7902	مهندسی کامپیوتر	مجید رضایی
8001	مهندسی کامپیوتر	پروین صبا

در جدول project دو کلید کاندیدا وجود دارد:

کلید کاندیدای ۱:st#+field

کلید کاندیدای ۲:st#+tutor

فرض کنید کلید کاندیدای اول را به عنوان کلید اصلی انتخاب کنیم. در اینصورت، نمودار وابستگی این جدول به شکل ذیل خواهد بود:

جدول Project نرمال 2 است چون در آن هیچ ویژگی غیر کلیدی به قسمتی از کلید اصلی وابستگی ندارد(تنها ویژگی غیر کلیدی این جدول tutor است که نه به #st تنها وابسته است و نه به field تنها)

این جدول نرمال ۳ نیز هست چون در آن هیچ ویژگی غیر کلیدی به ویژگی غیر کلیدی دیگر وابستگی ندارد(چون این جدول تنها یک ویژگی غیر کلیدی دارد، این مسئله بدیهی است)

با وجود آنکه این جدول نرمال ۳ است ولی هنوز ناهنجاریهایی دارد:

۱-**ناهنجاری در درج:** به عنوان مثال نمی‌توان استاد جدیدی را که هنوز هیچ دانشجویی با وی پروژه نگرفته است در جدول درج کرد چون #st جزئی از کلید اصلی است و نمی‌تواند تهی باشد.

۲ـ **ناهنجاری در حذف:** فرض کنید دانشجوی شمارهٔ ۸۰۰۱ تنها دانشجویی باشد که با پروین صبا پروژه گرفته است. در این صورت اگر این دانشجو را حذف کنیم، اطلاعات مربوط به این استاد نیز از بین می‌رود.

برای رفع این معایب، دو شخص به نامهای Boyce و Codd فرم بهینه تری از نرمال ۳ را پیشنهاد کردند که به افتخار این دو شخص با نام BCNF معروف شد.

یک جدول نرمال BCNF است اگر و تنها اگر کلیهٔ تعیین کننده[1] های آن کلید کاندیدا باشند.

تعریف تعیین کننده: اگر در یک جدول ویژگی B به ویژگی A وابستگی تابعی داشته باشد (A→B)، آنگاه A یک تعیین کننده خواهد بود.

جدول Project نرمال BCNF نیست چون:

tutor → field

1 - Determinant

پس tutor یک تعیین کننده است در حالی که کلید کاندیدا نیست.

روند تبدیل جداول نرمال ۲ به نرمال BCNF کاملاً مشابه روند تبدیل جداول نرمال ۲ به نرمال ۳ است. برای تبدیل جدول project به جداول نرمال BCNF:

۱- tutor و field را در یک جدول مجزا قرار داده، tutor را به عنوان کلید اصلی این جدول معرفی می‌کنیم:

TC (tutor, field)

۲- بقیهٔ ویژگیها(st#) را در جدول دیگری قرار داده، کلید اصلی جدول اول یعنی tutor را به آنها اضافه می‌کنیم:

ST (st# , tutor)

TC

tutor	field
مجید رضایی	مهندسی کامپیوتر
آرش ریاضیدان	ریاضی محض
گلناز هنردوست	هنر
مجید رضایی	مهندسی کامپیوتر
پروین صبا	مهندسی کامپیوتر

ST

st#	tutor
7801	مجید رضایی
7801	آرش ریاضیدان
7801	گلناز هنردوست
7902	مجید رضایی
8001	پروین صبا

تذکر: حالتهای نرمال ۳ و نرمال BCNF معمولاً با هم معادلند. تنها در صورتی که جدول بیش از یک کلید کاندیدا داشته باشد و میان کلیدهای کاندیدا ویژگی مشترک وجود داشته باشد این دو حالت با هم معادل نخواهند بود. مثلاً در جدول Project دو کلید کاندیدا وجود دارد و st# میان دو کلید کاندیدا مشترک است بهمین دلیل حالت نرمال ۳ و BCNF این جدول معادل نیستند.

۵-۵) جداول نرمال ٤

برخی از جداول در حالت نرمال BCNF هستند ولی هنوز ناهنجاریهایی دارند. جدول employee را در نظر بگیرید. این جدول شامل شمارهٔ کارمندی و مهارتها و زبانهایی است که هر کارمند بر آنها مسلط است:

employee

emp#	skills	languages
100	برنامه نویسی جاوا تجزیه و تحلیل شی گرا	انگلیسی
101	برنامه نویسی دلفی تجزیه و تحلیل شی گرا طراحی وب سایت	انگلیسی آلمانی

در این جدول ستونهای skills و languages دارای گروههای اطلاع تکرار شونده هستند.نمودار وابستگی این جدول به شکل ذیل است:

employee(emp# , skills , languages)

فرض کنید این جدول را بصورت ذیل تبدیل به نرمال ۱ کنیم:

employee2

emp#	skill	Language
100	برنامه نویسی جاوا	انگلیسی
100	تجزیه و تحلیل شی گرا	انگلیسی
101	برنامه نویسی دلفی	انگلیسی
101	تجزیه و تحلیل شی گرا	انگلیسی
101	طراحی وب سایت	انگلیسی
101	برنامه نویسی دلفی	آلمانی
101	تجزیه و تحلیل شی گرا	آلمانی
101	طراحی وب سایت	آلمانی

نمودار وابستگی جدول employee2 به شکل ذیل خواهد بود:

employee2(emp# , skill , language)

جدول employee2 نرمال BCNF است(چون یک جدول تمام کلید است این مسئله بدیهی است) ولی در درج، حذف و اصلاح ناهنجاری دارد. مثلاً اگر کارمند ۱۰۱ مهارت "برنامه نویسی جاوا" را نیز کسب کند، بجای اضافه کردن یک تاپل باید دو تابل جدید ذیل به جدول اضافه شود:

انگلیسی	برنامه نویسی جاوا	101
آلمانی	برنامه نویسی جاوا	101

و یا اگر بخواهیم زبان آلمانی را از اطلاعات کارمند ۱۰۱ حذف کنیم، بجای حذف یک تاپل باید سه تاپل ذیل را حذف کنیم:

آلمانی	برنامه نویسی دلفی	101
آلمانی	تجزیه و تحلیل شی گرا	101
آلمانی	طراحی وب سایت	101

برای رفع این معایب جداول را به نرمال ۴ تبدیل می‌کنیم.

یک جدول نرمال ۴ است اگر:

۱- نرمال BCNF باشد.

۲- هیچ وابستگی چند مقداری[1] در آن وجود نداشته باشد.

1 - MultiValued Dependency

تعریف وابستگی چند مقداری: اگر در جدول T(A,B,C)، به ازاء هر مقدار برای ویژگی A مجموعه‌ای از مقادیر برای ویژگی C وجود داشته باشد و این مجموعه مستقل از مقادیر ویژگی B باشد، ویژگی C به ویژگی A وابستگی چندمقداری دارد(A→→C).

مثال ۱: جدول employee2(emp#,skill,language) را در نظر بگیرید:

(emp#,skill)=(101,برنامه نویسی دلفی) ⟹ language={آلمانی,انگلیسی}

(emp#,skill)=(101,تجزیه و تحلیل شی گرا) ⟹ language={آلمانی,انگلیسی}

(emp#,skill)=(101,طراحی وب سایت) ⟹ language={آلمانی,انگلیسی}

مشاهده می‌کنید که تنها #emp مجموعهٔ language را تعیین می‌کند و تغییر skill هیچ نقشی در تغییر مجموعهٔ language ندارد. دلیل این امر نیز آنست که مهارتهای یک شخص و زبانهایی که شخص بر آنها مسلط است هیچ ارتباطی با یکدیگر ندارند. پس language به #emp وابستگی چندمقداری دارد(emp#→→language).

از طرف دیگر:

(emp#,language)=(101,انگلیسی) ⟹ skill={برنامه نویسی دلفی, تجزیه و تحلیل شی گرا, طراحی وب سایت}

(emp#,language)=(101,آلمانی) ⟹ skill={برنامه نویسی دلفی, تجزیه و تحلیل شی گرا, طراحی وب سایت}

در واقع، تنها #emp مجموعهٔ skill را تعیین می‌کند و language هیچ نقشی در تغییر مجموعهٔ skill ندارد پس skill نیز به #emp وابستگی چندمقداری دارد(emp#→→skill).

برای جلوگیری از ایجاد وابستگی چند مقداری، در همان مرحلهٔ اول و هنگام تبدیل جدول آنرمال به نرمال ۱ به ازاء هر گروه اطلاع تکرار شونده باید جدول مجزایی در نظر بگیریم:

Emp_language

emp#	language
100	انگلیسی
101	انگلیسی
101	آلمانی

Emp_skill

emp#	skill
100	برنامه نویسی جاوا
100	تجزیه و تحلیل شی گرا
101	برنامه نویسی C#
101	تجزیه و تحلیل شی گرا
101	طراحی وب سایت

Emp_language(emp#,language)

Emp_skill(emp#,skill)

هر دو جدول Emp_skill و Emp_language نرمال ۴ هستند.

مثال ۲: جدول معروف SPJ را در نظر بگیرید:

SPJ

S#	P#	J#	Qty
S1	P1	J1	12000
S1	P2	J1	20000
S1	P3	J1	3000
S1	P4	J1	8000
S1	P1	J2	10000
S1	P1	J3	9000
S2	P1	J1	2000
S2	P1	J3	5000
S2	P3	J1	8000
S3	P1	J1	9000
S3	P2	J3	9000
S3	P1	J3	4000

هر تولید کننده ممکن است برای چندین پروژه محصولاتی تولید کند بنابراین، به ازاء هر #s مجموعه‌ای از #j-ها و مجموعه‌ای از #p-ها وجود دارند اما:

۱) ویژگی #j به #s وابستگی چندمقداری ندارد چون:

$(s\#,p\#)=(S2,P1) \Longrightarrow j\#=\{J1,J3\}$

$(s\#,p\#)=(S2,P3) \Longrightarrow j\#=\{J1\}$

پس تنها #s مجموعهٔ #j را تعیین نمی‌کند بلکه با تغییر مقدار #p نیز احتمال تغییر مجموعهٔ مقادیر #j وجود دارد.

۲) ویژگی #p به #s وابستگی چندمقداری ندارد چون:

$(s\#,j\#)=(S1,J1) \Longrightarrow p\#=\{P1,P2,P3,P4\}$

$(s\#,j\#)=(S1,J2) \Longrightarrow p\#=\{P1\}$

پس تنها #s مجموعهٔ #p را تعیین نمی‌کند و با تغییر مقدار #j نیز احتمال تغییر مجموعهٔ مقادیر #p وجود دارد.

می‌توان نتیجه گرفت در جدول SPJ وابستگی چندمقداری وجود ندارد و این جدول نرمال ۴ است.

۶-۵) جداول نرمال ۵

یک جدول در حالت نرمال ۵ است اگر:

۱- نرمال ۴ باشد.

۲- نتوان آنرا به جداول کوچکتر تجزیه کرد بطوریکه حداقل یکی از جداول شامل هیچ یک از کلیدهای کاندیدای جدول اولیه نباشد.

مثال ۱: طبق قوانین یک دانشگاه:

- هر دانشجو می‌تواند هر درس را چند بار ولی تنها یکبار با هر استاد بگیرد.
- هر استاد می‌تواند درسهای مختلفی را تدریس کند و هر درس توسط اساتید مختلف تدریس می‌شود.

جدول enroll را در نظر بگیرید:

enroll

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P2
S2	C2	P1

تنها کلید کاندیدای این جدول st#+crs#+prof# و نمودار وابستگی این جدول به شکل ذیل است:

enroll(st#,crs#,prof#)

این جدول نرمال BCNF است و چون وابستگی چندمقداری ندارد، نرمال ۴ نیز هست. برای تشخیص نرمال ۵ بودن این جدول باید بررسی کنیم آیا این جدول قابل تجزیه به چند جدول کوچکتر که پیوند آنها جدول اولیه را نتیجه دهد هست یا خیر؟

این جدول را به دو جدول enroll1 و enroll2 می‌شکنیم(هر یک از این جداول یک پرتو[1] از جدول اصلی هستند):

enroll1

St#	Crs#
S1	C1
S1	C2
S2	C2

enroll2

St#	Prof#
S1	P1
S1	P2
S2	P1

1 - Projection

نتیجهٔ پیوند دو جدول بصورت ذیل خواهد بود:

enroll1 join enroll2

St#	Crs#	Prof#
S1	C1	P1
S1	C1	P2
S1	C2	P1
S1	C2	P2
S2	C2	P1

همانطور که مشاهده می‌کنید پیوند دو جدول دو تاپل اضافی تولید کرد که در جدول enroll اولیه وجود نداشت پس این تجزیه بی‌فایده است.

بهمین ترتیب، اگر این جدول را به دو جدول enroll3 و enroll4 بشکنیم:

enroll3

St#	Crs#
S1	C1
S1	C2
S2	C2

enroll4

Crs#	Prof#
C1	P1
C2	P1
C2	P2

نتیجهٔ پیوند دو جدول بصورت ذیل خواهد بود:

enroll3 join enroll4

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P1
S1	C2	P2
S2	C2	P1
S2	C2	P2

همانطور که مشاهده می‌کنید پیوند دو جدول دو تاپل اضافی تولید کرد که در جدول enroll اولیه وجود نداشت پس این تجزیه نیز بی‌فایده است.

حال روی جدول enroll سه پرتو می‌گیریم:

SC

St#	Crs#
S1	C1
S1	C2
S2	C2

CP

Crs#	Prof#
C1	P1
C2	P2
C2	P1

SP

St#	Prof#
S1	P1
S1	P2
S2	P1

ویژگی مشترک بین دو جدول SC و CP، #st است. نتیجهٔ پیوند دو جدول، جدول ذیل خواهد بود:

SC join CP

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P2
S1	C2	P1
S2	C2	P2
S2	C2	P1

ویژگی مشترک بین دو جدول (SC join CP) و SP، #st+prof# است. نتیجهٔ پیوند دو جدول ، جدول ذیل خواهد بود:

(SC join CP) join SP

St#	Crs#	Prof#
S1	C1	P1
S1	C2	P2
S1	C2	P1
S2	C2	P1

همانگونه که مشاهده می‌کنید نتیجهٔ SC join CP join SP با جدول enroll برابر نیست و شامل تاپل اضافی (S1,C2,P1)است که در جدول enroll وجود ندارد. این مشکل از آنجا ناشی می‌شود که:

۱- دانشجوی S1 درس C2 را گرفته است

۲- دانشجوی S1 با استاد P1 درس گرفته است

۳- درس C2 جز درسهایی است که استاد P1 تدریس می‌کند

چنین نتیجه شده است که دانشجوی S1 درس C2 را با P1 گرفته است درحالیکه این واقعیت ندارد.

جدول enroll سه ستون دارد پس نمی‌توان آنرا به صورت ستونی به بیش از ۳ جدول تجزیه کرد و چون پیوند هیچ یک از تجزیه‌های دوتایی و سه تایی ما را به جدول اصلی نرساندند، می‌توان نتیجه گرفت جدول enroll قابل تجزیه به جداول کوچکتر که پیوند آنها جدول اولیه را نتیجه دهد نیست **پس جدول enroll نرمال ۵ است.**

مثال ۲: فرض کنید قوانین ذیل در یک دانشگاه حاکم باشند:

- هر استاد می‌تواند درسهای مختلفی را تدریس کند و هر درس توسط اساتید مختلف تدریس می‌شود.
- هر استاد برای هر درس می‌تواند چند کتاب معرفی کند.
- اگر استادی یک درس را تدریس کند، علاوه بر کتابهایی که خود معرفی می‌کند مجبور است کلیهٔ کتابهایی که سایر مدرسین آن درس برای آن درس معرفی کرده اند را نیز معرفی کند!!!

جدول PCB را در نظر بگیرید:

PCB

prof#	crs#	book#
P1	C1	B1
P1	C2	B2
P3	C7	B8

تنها کلید کاندیدای این جدول prof#+crs#+book# و نمودار وابستگی این جدول به شکل ذیل است:

PCB(prof#,crs#,book#)

این جدول نرمال ۴ است ولی یک ناهنجاری بسیار عجیب دارد. مثلاً اگر بخواهیم تاپل (P2,C2,B5) را در جدول درج کنیم، برای رعایت محدودیت اعمال شده از طرف دانشگاه، باید تاپل (P2,C2,B2) و تاپل (P1,C2,B5) را نیز درج کنیم!!!

حال باید تشخیص دهیم که با توجه به محدودیت اعمال شده از طرف دانشگاه این جدول قابل تجزیه به جداول کوچکتر است یا خیر؟ تجزیه به دو جدول نتیجه‌ای ندارد پس تجزیهٔ سه‌تایی را مورد بررسی قرار می‌دهیم:

PC

prof#	Crs#
P1	C1
P1	C2
P3	C7

CB

Crs#	book#
C1	B1
C2	B2
C7	B8

PB

prof#	book#
P1	B1
P1	B2
P3	B8

پیوند سه جدول فوق ما را به جدول اولیه می‌رساند. به عبارت دیگر:

PC join CB join PB=PCB

از طرف دیگر هیچ یک از جداول PC یا CB یا PB شامل کلید کاندیدای جدول اصلی نیستند **پس جدول PCB نرمال ۵ نیست** و باید به سه جدول PC و CB و PB که هر سه نرمال ۵ هستند شکسته شود.

مثال ۳: جدول course را در نظر بگیرید:

course

crs#	cname	unit
1100	ادبیات	2
1105	تربیت بدنی	1
1400	برنامه سازی ۱	3
1402	برنامه سازی ۲	3

کلیدهای کاندیدای این جدول عبارتند از:

کلید کاندیدای ۱: #crs

کلید کاندیدای ۲: cname

اگر #crs را به عنوان کلید اصلی معرفی کنیم، نمودار وابستگی این جدول به شکل ذیل خواهد بود:

course(crs#,cname,unit)

این جدول را می‌توان:

۱) به دو جدول {(crs#,cname),(cname,unit)} تجزیه کرد. در اینصورت نتیجهٔ پیوند دو جدول با جدول course اولیه یکسان خواهد بود ولی انجام این تقسیم کاملاً بی‌مورد است چون در هر دو جدول حداقل یکی از کلیدهای کاندیدا حضور دارند.

۲) به دو جدول {(crs#,cname),(crs#,unit)} تجزیه کرد. در اینصورت نتیجهٔ پیوند دو جدول با جدول course اولیه یکسان خواهد بود ولی انجام این تقسیم کاملاً بی‌مورد است چون در هر دو جدول حداقل یکی از کلیدهای کاندیدا حضور دارند.

پس می‌توان نتیجه گرفت که جدول course نرمال ۵ است و لازم نیست به جداول کوچکتر تجزیه شود.

تذکر: نرمال ۵ در عمل کاربردی ندارد و اکثر جداول تنها تا مرحلهٔ 3NF نرمال می‌شوند چون در حالت نرمال ۳ هیچ مشکلی ندارند.

تمرینهای حل شده:

تمرین ۱: سیستم کتابخانهٔ یک دانشگاه را در نظر بگیرید. در این کتابخانه به هر کتاب یک شمارهٔ منحصر بفرد داده شده است مثلاً اگر در کتابخانه سه نسخه از یک کتاب وجود داشته باشد، هر یک از آنها یک شمارهٔ مجزا خواهد داشت. در هر امانت دانشجو می‌تواند حداکثر سه کتاب را به امانت بگیرد(مثلاً در امانت شمارهٔ ۱۰۰۱ ممکن است دو کتاب ۱۴۰۳ و ۱۸۰۹ به امانت گرفته

شوند) و باید همهٔ کتابهایی را که با هم از کتابخانه امانت گرفته است با هم به کتابخانه برگرداند. با توجه به این قوانین، نمودار وابستگی جدول loan را رسم کرده، آنرا به جداول نرمال ۳ تبدیل کنید:

loan(loan# , book# , bname , author ,st# , sname , field ,loanDate , returnDate)

loan#: شمارهٔ امانت

book#: شمارهٔ کتاب

bname: نام کتاب

author: نام نویسنده

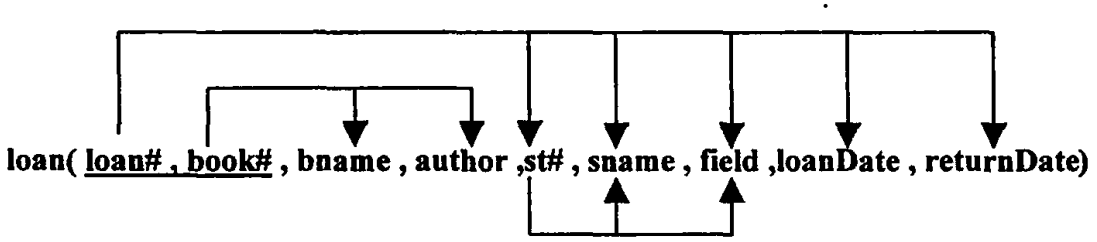
st#: شمارهٔ دانشجویی که کتاب را به امانت گرفته است

sname: نام دانشجو

field: رشتهٔ تحصیلی دانشجو(هر دانشجو تنها در یک رشته تحصیل می‌کند)

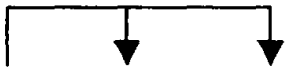
loanDate: تاریخ امانت گرفتن کتاب

returnDate: تاریخ برگشت کتاب به کتابخانه توسط دانشجو



تبدیل به نرمال ۲:

این جدول نرمال ۲ نیست چون bname و author به #book که جزئی از کلید اصلی است وابستگی تابعی دارند. همچنین، #st و sname و field و loanDate و returnDate به #loan که جزئی از کلید اصلی است وابستگی تابعی دارند. پس این جدول را به جداول نرمال ۲ می‌شکنیم:



L(loan# , st# , sname , field , loanDate, returnDate)

Book(book#, bname , author)

LB(loan#,book#)

نکته: توجه کنید اگر چه هیچ ویژگی غیر کلیدی که به #loan#+book وابستگی داشته و در سطرهای قبل نیامده باشد وجود ندارد، ولی این ردیف را به جدول LB تبدیل کرده ایم. دلیل این امر آنست که تنها در این جدول مشخص می‌شود در هر امانت چه کتابهایی امانت گرفته شده اند پس این جدول، اطلاعات مفیدی در اختیار می‌گذارد که در جداول دیگر موجود نیست.

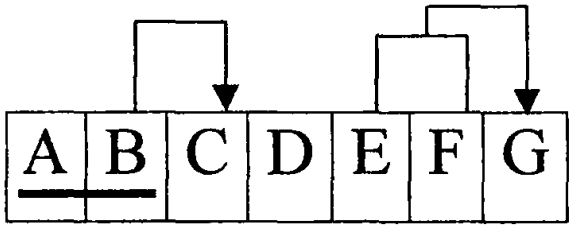
تبدیل به نرمال ۳: جداول B و LB نرمال ۳ هستند چون در آنها هیچ ویژگی غیر کلیدی به ویژگی غیر کلیدی دیگر وابستگی تابعی ندارد ولی جدول L نرمال ۳ نیست چون در آن sname و field به #st که غیر کلیدی است وابستگی دارد. پس جدول L را به جداول ذیل که هر یک نرمال ۳ هستند می‌شکنیم:

L1 (st# , sname , field)

L2 (loan# , loanDate , returnDate)

پس جدول loan به چهار جدول LB و book و L1 و L2 که همگی نرمال ۳ هستند شکسته شد.

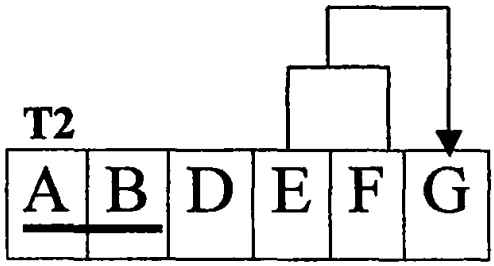
تمرین ۲: جدول ذیل را با ذکر کلیهٔ مراحل به جداول نرمال ۳ تبدیل کنید.



تبدیل به نرمال ۲: از آنجا که ویژگی C به ویژگی B که بخشی از کلید اصلی است وابستگی تابعی دارد، در این جدول وابستگی جزئی وجود دارد پس نرمال ۲ نیست. برای تبدیل به نرمال ۲:

T1

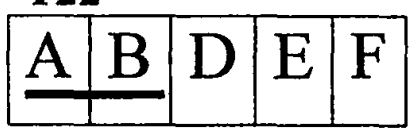
T2



تبدیل به نرمال ۳: جدول T1 نرمال ۳ است ولی در جدول T2 ویژگی G که یک ویژگی غیر کلیدی است، به ویژگیهای E و F که غیر کلیدی می‌باشند وابستگی تابعی دارد پس جدول T2 نرمال ۳ نیست. برای تبدیل به نرمال ۳ این جدول را به دو جدول T21 و T22 می‌شکنیم:

T21

T22



تمرینهای فصل

۱- جدول factor را که شامل اطلاعات مربوط به فاکتورهای فروش یک فروشگاه است در نظر بگیرید. در هر فاکتور ممکن است اقلام زیادی وجود داشته باشند.

factor (fac#,fdate,customer#,cname,address,item#,iname,fee,qty)

#fac: شمارهٔ فاکتور

fdate: تاریخ صدور فاکتور

#customer: شمارهٔ مشتری

cname: نام مشتری

address: آدرس مشتری

#item: کد کالای خریداری شده

iname: نام کالا

fee: هزینهٔ هر واحد

qty: مقدار یا تعداد کالای خریداری شده

نمودار وابستگی این جدول را رسم کرده، آنرا به جداول نرمال ۳ تبدیل کنید.

۲- یک وب سایت برای برگزاری آزمونهای مختلف را درنظر بگیرید. هر آزمون تعدادی سوال و هر سوال تعدادی گزینه دارد که تنها یکی از آنها صحیح است. اعضاء سایت با وارد کردن نام کاربری و کلمهٔ عبور خود وارد سایت شده، آزمون مورد نظر خود را انتخاب کرده، به سوالات چند گزینه ای آن پاسخ می‌دهند و امتیاز می‌گیرند. هر کاربر می‌تواند در آزمونهای مختلف ولی در هر آزمون تنها یکبار شرکت کند. نحوهٔ ارزیابی آزمونهای مختلف متفاوت است مثلاً ممکن است در یک آزمون امتیاز منفی به ازا هر جواب نادرست ۰,۲۵ و در دیگری ۰,۵ باشد و یا حداقل امتیاز کل برای قبولی در یک آزمون ۱۰ و در دیگری ۸۰ باشد. جدول test را که شامل اطلاعات آزمونهای مختلف است در نظر گرفته، نمودار وابستگی آنرا رسم کرده، به جداول نرمال ۳ تبدیل کنید:

test (userid , password , test# , title , creditPerTrue , negPerFalse , total , neededCredit , tdate , q# , qtext ,trueChoice# ,choice#, ctext, userChoice#)

userid: نام کاربری که در آزمون شرکت کرده است(هر کاربر یک نام منحصر بفرد دارد)

password: کلمهٔ عبور کاربر

test#: شمارهٔ آزمون(هر آزمون یک شمارهٔ منحصر بفرد دارد)

title: عنوان آزمون

creditPerTrue: امتیاز به ازاء هر جواب صحیح

negPerFalse: امتیاز منفی به ازاء هر جواب نادرست

total: کل امتیاز آزمون

neededCredit: حداقل امتیاز مورد نیاز برای قبولی در آزمون

tdate: تاریخ شرکت کاربر در آزمون

q#: شمارهٔ سوال(در هر آزمون هر سوال یک شمارهٔ منحصر بفرد دارد)

qtext: متن سوال

trueChoice#: شمارهٔ گزینهٔ درست(در هر یک از سوالها تنها یکی از گزینه‌ها صحیح است.)

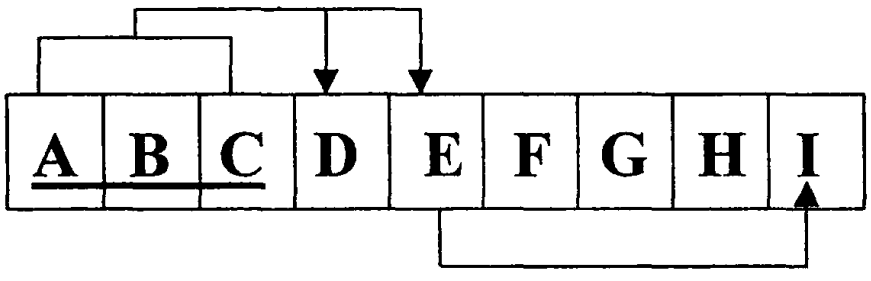
choice#: شمارهٔ گزینه(در هر سوال هر گزینه یک شماره دارد)

ctext: متن گزینه

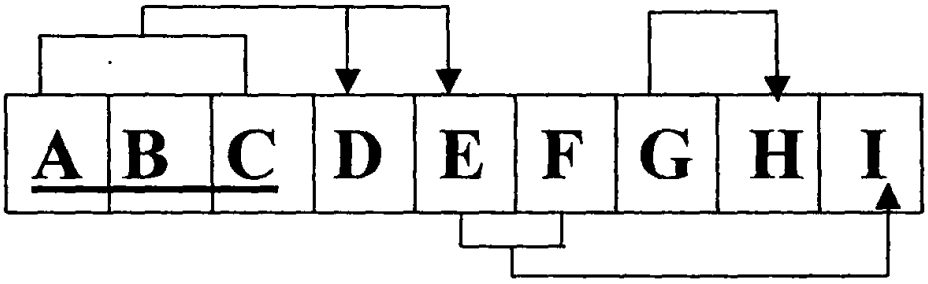
userChoice#: شمارهٔ گزینه ای که کاربر برای این سوال انتخاب کرده است

۳–جداول ذیل را به جداول نرمال BCNF تبدیل کنید:

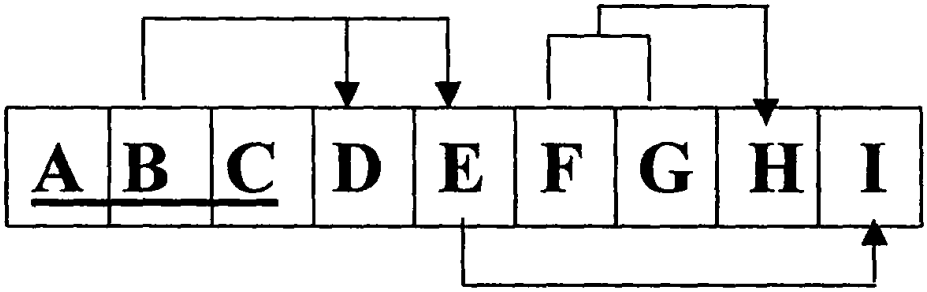
الف–



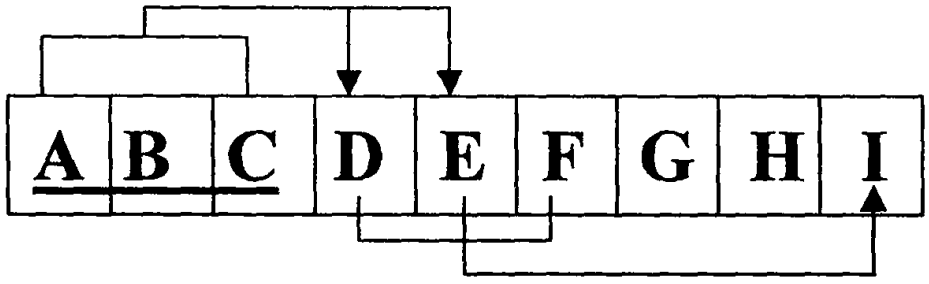
ب-



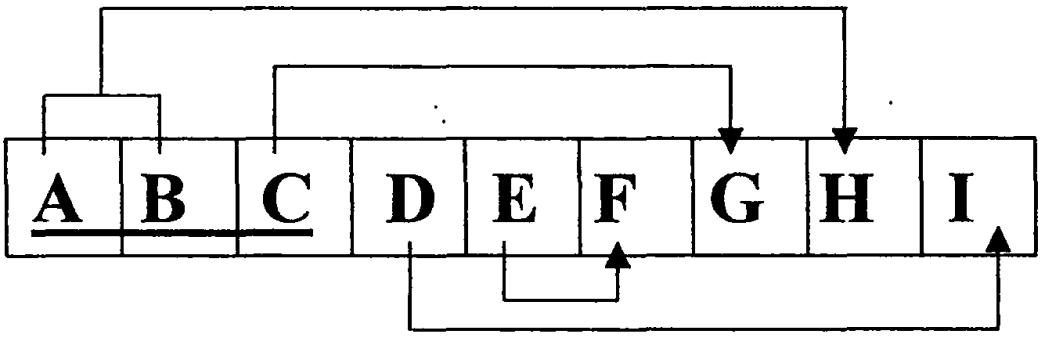
ج-



د-



هـ-



۶

مدلسازی داده‌ها با استفاده از نمودارهای موجودیت ـ رابطه (ERD[1])

اگر کل داده‌های موجود در سیستم را به یک فیل تشبیه کنیم، دید کاربران، طراحان و برنامه‌نویسان نسبت به داده‌های سیستم بی‌شباهت به دید مردان کوری که هر یک می‌توانستند قسمتی از بدن فیل را لمس کنند نخواهد بود چرا که هر یک از آنها از زاویهٔ خاص خود به داده‌های سیستم نگاه می‌کنند. مثلاً دید مدیر کل یک سازمان نسبت به داده‌های سازمان یک دید کلی است در حالیکه مدیر یک بخش تنها به جزئیات داده‌های بخش خود آگاه است و یا آنچه از دید یک برنامه‌نویس اهمیت دارد، فرمت داده‌ها و نحوهٔ استخراج گزارشات مورد نیاز از داده‌های ذخیره شده است. مدلهای داده، نمایشی نسبتاً ساده و غالباً گرافیکی از ساختارهای داده در دنیای واقعی هستند که می‌توانند به عنوان یک پل ارتباطی میان طراحان، برنامه‌نویسان و کاربران نهایی مورد استفاده قرار گرفته، آنها را در رسیدن به یک دید مشترک و فهم متقابل از برداشتهای یکدیگر یاری کنند. نمودارهای ER یکی از رایجترین ابزارها برای مدلسازی داده‌ها هستند. در این فصل نحوهٔ رسم نمودارهای ER و همچنین، قواعد نگاشت این نمودارها به جداول پایگاه داده‌ها را مورد بررسی قرار می‌دهیم.

1- Entity–Relationship Diagram

۱-٦) گروه‌بندی مدلهای داده براساس درجات تجرید داده‌ها[1]

مدلهای داده از نظر درجهٔ تجرید داده‌ها به ۴ گروه تقسیم می‌شوند :

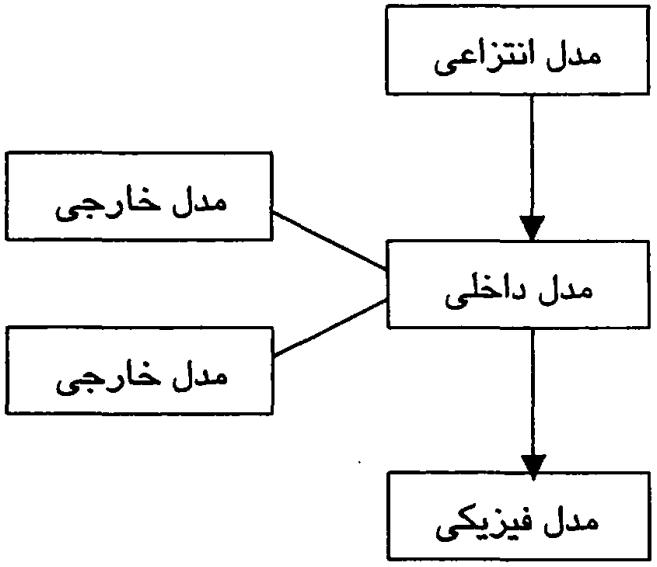
۱- مدل انتزاعی[2] (بالاترین درجهٔ تجرید)

۲- مدل داخلی[3]

۳- مدل خارجی[4]

٤- مدل فیزیکی[5] (پایینترین درجهٔ تجرید)

شکل ذیل ارتباط این مدلها را نشان می‌دهد:



۱-۱-٦) مدل انتزاعی

مدل انتزاعی یک دید سراسری نسبت به کل داده‌های سیستم ارائه می‌دهد و در واقع، دید مدیران سطح بالا نسبت به داده‌ها است.

ایجاد مدل انتزاعی اولین مرحله از طراحی پایگاه داده‌ها می‌باشد. غالباً برای ایجاد این مدل از نمودارهای ER استفاده می‌شود. مدل انتزاعی مستقل از نرم افزار(DBMS)، سخت افزار و سیستم عاملی است که قرار است پایگاه داده‌ها روی آنها پیاده‌سازی شود.

---

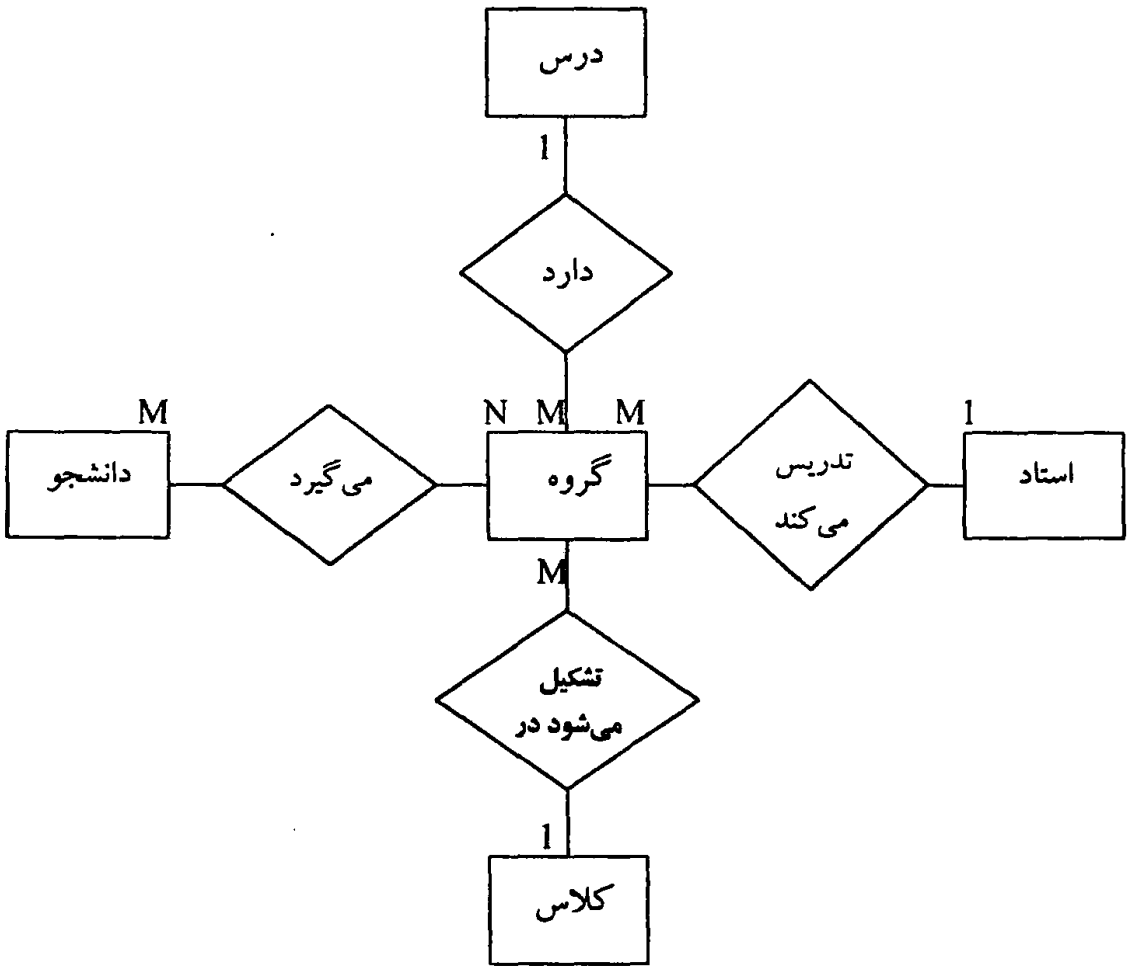
1 - Data Abstraction
2 - Conceptual Model
3 - Internal Model
4 - External Model
5 - Physical Model

مثال: سیستم یک دانشگاه را در نظر بگیرید. موجودیتهای این سیستم عبارتند از : دانشجو، استاد، درس، گروه درسی، کلاس.

طبق قوانین این دانشگاه:

- برای هر درس ممکن است یک یا چند گروه مختلف ارائه شود مثلاً برای درس پایگاه داده‌ها ممکن است چهار گروه مختلف ارائه شود.
- هر استاد در یک یا چند گروه درسی تدریس می‌کند ولی هر گروه درسی تنها توسط یک استاد تدریس می‌شود.
- در هر گروه درسی حداقل ۱۰ و حداکثر ۳۰ دانشجو ثبت نام می‌کنند. هر دانشجو می‌تواند در یک یا چند گروه درسی ثبت نام کند.
- هر گروه درسی در یک کلاس برگزار می‌شود ولی در طول هفته در هر کلاس ممکن است چندین گروه درسی برگزار شود.

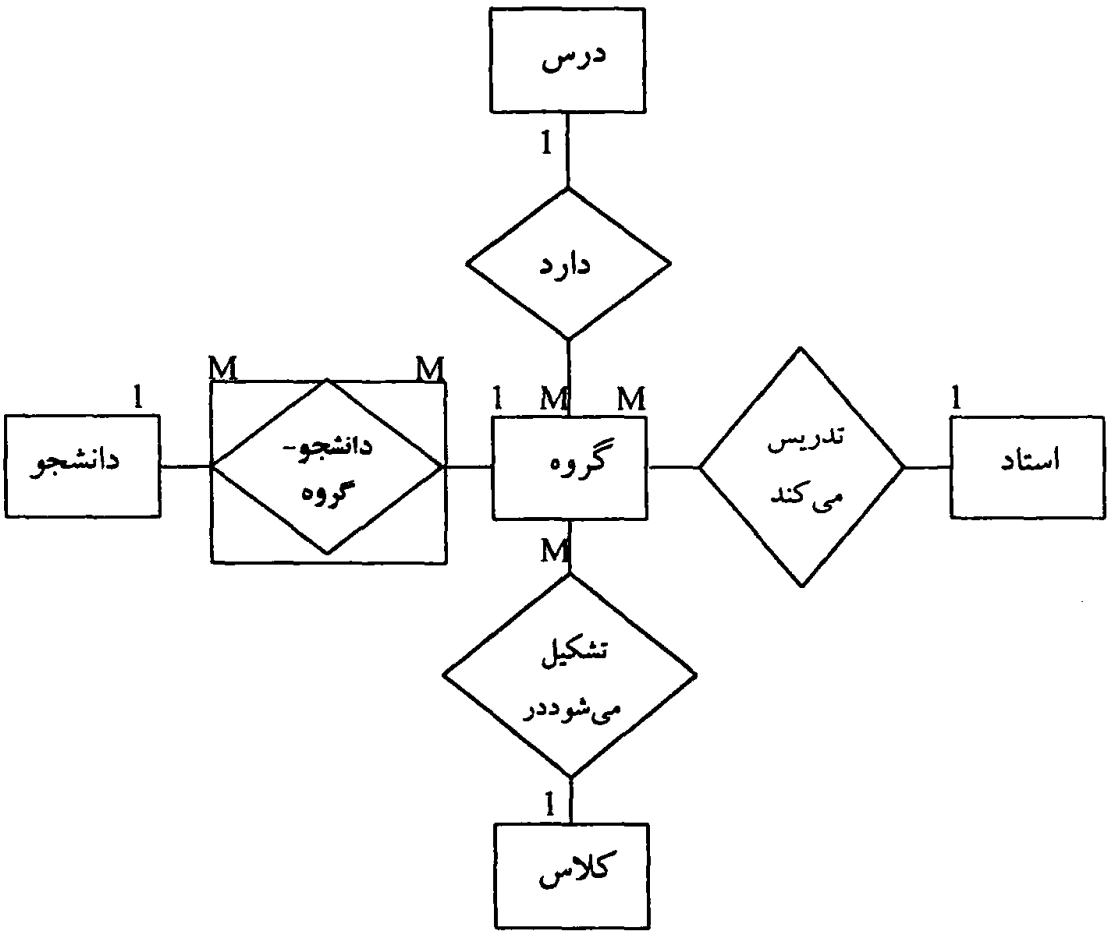
شکل ذیل مدل انتزاعی این سیستم را نشان می‌دهد:



۲-۱-٦) مدل داخلی

مدل داخلی، همان مدل انتزاعی است با این تفاوت که جزئیات و شرایط خاص مدل DBMS (مدل شبکه‌ای ـ مدل سلسله‌مراتبی ـ مدل رابطه‌ای و ...) نیز در آن در نظر گرفته می‌شود. به عبارت دیگر، مدل داخلی وابسته به نرم افزار است. در صورت استفاده از مدل شبکه‌ای یا سلسله‌مراتبی، بعلت وابستگی شدید مدل به آدرسهای فیزیکی و نحوهٔ ذخیره‌سازی داده‌ها، در مدل داخلی باید جزئیات مربوط به مسیرها و شیوهٔ ذخیره‌سازی داده‌ها مشخص شود ولی در صورت استفاده از مدل دادهٔ رابطه‌ای، نیازی به ارائهٔ این جزئیات نیست چرا که DBMS خود بصورت مخفیانه و بدون اطلاع طراح پایگاه داده‌ها، عملیات مربوط به ذخیره‌سازی فیزیکی داده‌ها را انجام می‌دهد. مدل داخلی از سخت افزار یا سیستم عاملی که قرار است پایگاه داده‌ها روی آن پیاده‌سازی شود مستقل است.

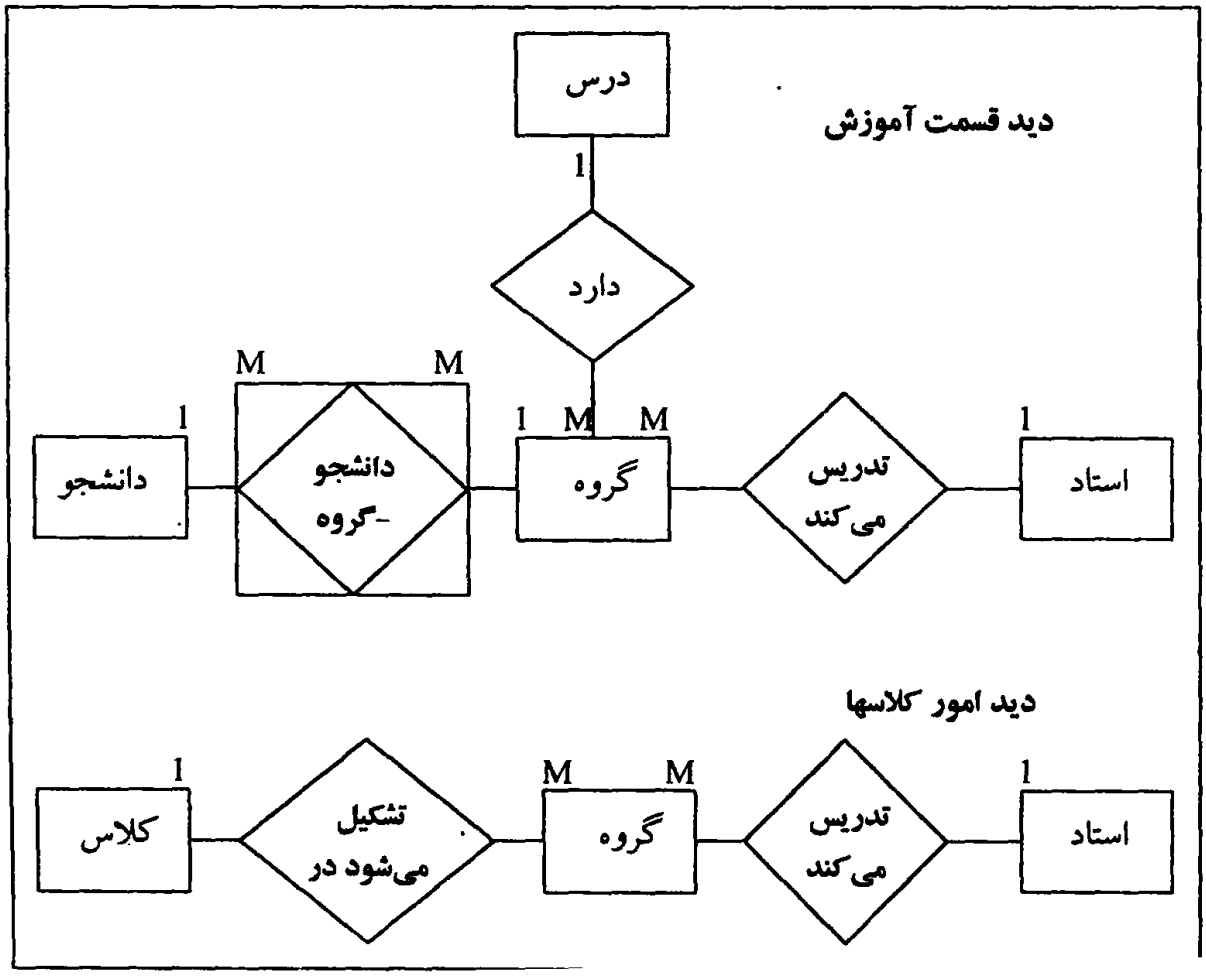
مثال: شکل ذیل مدل داخلی مربوط به سیستم دانشگاه ارائه شده در مثال قبل را نشان می‌دهد (هر رابطهٔ M:N به دو رابطهٔ 1:M شکسته شده است):



۳-۱-۶) مدل خارجی

مدل خارجی، دیدی را که هر یک از کاربران نهایی یا گروههای مختلف از کاربران نسبت به داده‌ها دارند نشان می‌دهد. کاربران مختلف با توجه به حوزهٔ مسئولیتها و وظایف خود دید متفاوتی نسبت به داده‌های سیستم دارند مثلاً دید مسئول امور کلاسها نسبت به داده‌های ذخیره شده در پایگاه داده‌های دانشگاه با دید کارمندان آموزش کاملاً متفاوت است. آنچه از دید مسئول امور کلاسها حائز اهمیت است مشخص کردن شمارهٔ کلاس برای هر گروه درسی و اعلام آن به استاد مورد نظر است. در مقابل، آنچه از دید مسئولان قسمت آموزش اهمیت دارد، ثبت نام دانشجویان در گروههای درسی است و نه محل برگزاری کلاسها. به همین ترتیب، دید برنامه‌نویسی که پیاده‌سازی قسمتهای مربوط به امور کلاسها را بر عهده دارد با دید برنامه‌نویسی که پیاده‌سازی قسمتهای مربوط به آموزش را بر عهده دارد متفاوت است چرا که دید هر یک از آنها از دید کاربران نهایی که قرار است از قسمتهای مورد نظر برنامه استفاده کنند سرچشمه می‌گیرد.

مثال: شکل ذیل مدل خارجی مثال قبل را نشان می‌دهد. این مدل شامل دیدهای خارجی قسمت آموزش و مسئول امور کلاسها می‌باشد. این دیدهای خارجی از دید داخلی که در قسمت قبل ارائه شد استخراج شده‌اند.



٤-١-٦) مدل فیزیکی

مدل فیزیکی نسبت به مدلهای دیگر دارای کمترین درجهٔ تجرید داده است. این مدل، نحوهٔ ذخیره‌سازی داده‌ها روی رسانه‌های ذخیره‌سازی مانند دیسکها را نشان می‌دهد. از آنجا که مدلهای سلسله‌مراتبی و شبکه‌ای با اشاره‌گرها سر و کار دارند، روش ذخیره‌سازی و بازیابی فیزیکی داده‌ها بر کارآیی سیستم تاثیر بسزایی دارد در حالیکه در پایگاه داده‌های رابطه‌ای چنین وضعیتی حاکم نیست. مدل فیزیکی به سخت افزار و نرم افزار مورد استفاده برای پیاده‌سازی پایگاه داده‌ها وابسته است.

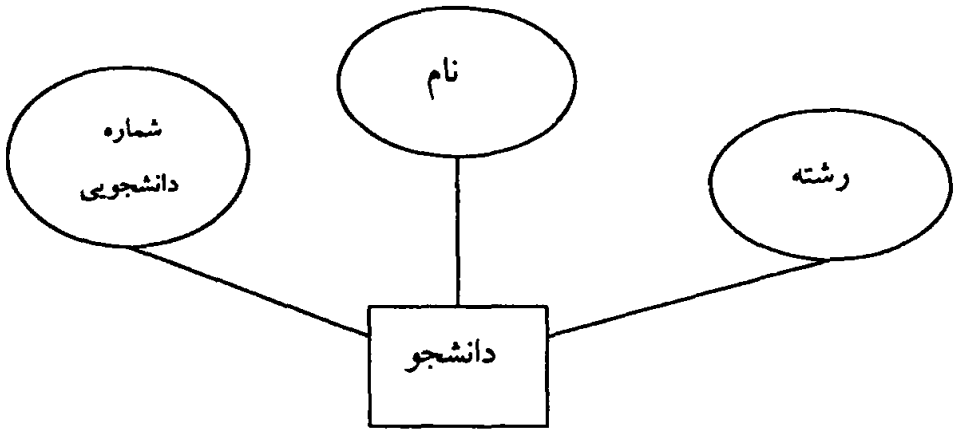
٢-٦) نمودارهای ER

نمودارهای ER رایجترین ابزار برای نمایش مدل انتزاعی داده‌ها می‌باشند. این نمودارها به عنوان ابزاری برای ایجاد درک متقابل و تفاهم میان طراحان پایگاه داده‌ها، مدیران، برنامه‌نویسان و کاربران نهایی مورد استفاده قرار می‌گیرند. علاوه بر این، می‌توان ساختار جداول پایگاه داده‌ها را از این نمودارها استخراج کرد. در صورت رعایت کلیهٔ اصول، جداول استخراج شده از نمودارهای ER کاملاً نرمال بوده و نیازی به نرمال‌سازی نخواهند داشت.

١-٢-٦) اجزاء اصلی نمودارهای ER

١-١-٢-٦) موجودیتها: چیزهایی هستند که می‌خواهیم دربارهٔ آنها اطلاعات جمع‌آوری کنیم. مثلاً در محیط دانشگاه موجودیتها عبارتند از: دانشجو، درس، استاد، کلاس. برای نمایش موجودیتها از مستطیل استفاده می‌شود.

٢-١-٢-٦) ویژگیها: خصلتهایی از یک موجودیت هستند که از نظر سیستم اهمیت دارند. مثلاً در مورد دانشجو آنچه از نظر سیستم آموزش دانشگاه اهمیت دارد شماره دانشجویی، نام، رشتهٔ تحصیلی، سال ورود و ... دانشجو می‌باشد و نه رنگ چشم یا قد دانشجو! برای نمایش ویژگیها از بیضی استفاده می‌شود.



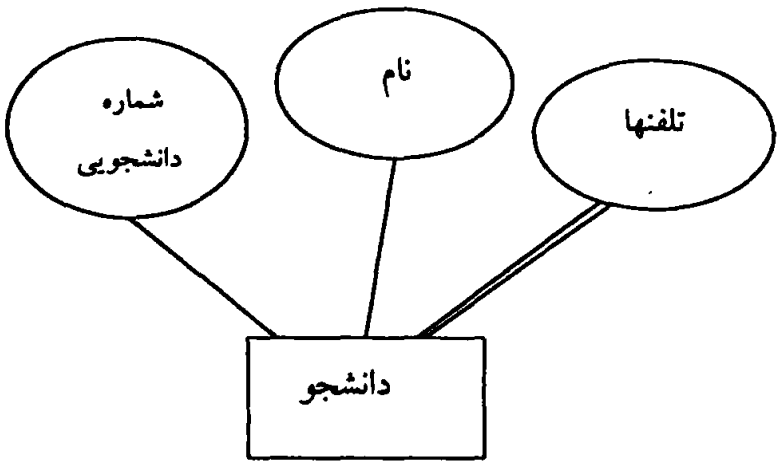
انواع ویژگیها:

-**ویژگی ساده:** ویژگیی است که تنها از یک جزء تشکیل می‌شود مثل شماره دانشجویی یا نام.

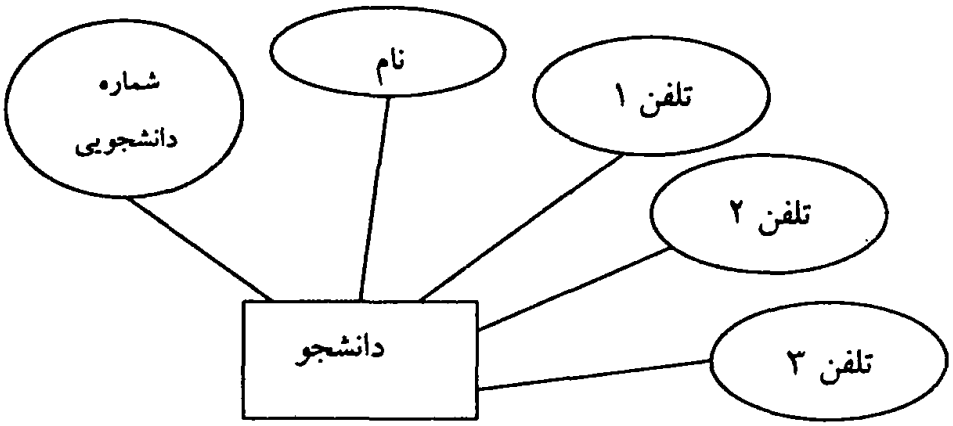
-**ویژگی مرکب:** ویژگیی است که از چند جزء تشکیل می‌شود مثل آدرس که شامل شهر، خیابان، کوچه و ... است.

-**ویژگی تک‌مقداری:** ویژگیی است که به ازاء هر موجودیت تنها یک مقدار دارد مثلاً هر دانشجو تنها یک نام خانوادگی دارد پس نام خانوادگی تک‌مقداری است.

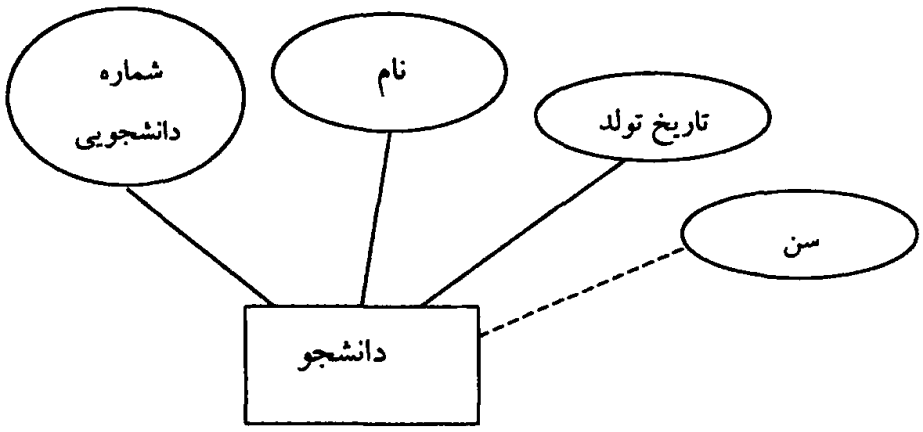
-**ویژگی چند مقداری:** ویژگیی است که به ازاء هر موجودیت چند مقدار دارد مثلاً ممکن است یک دانشجو چند شماره تلفن داشته باشد. ویژگی چندمقداری با دو خط به موجودیت وصل می‌شود.



تذکر: ویژگیهای چند مقداری را باید به ویژگیهای تک‌مقداری تبدیل کرد. مثلاً در مورد شماره تلفن می‌توان فرض کرد که برای هر دانشجو حداکثر سه شماره تلفن و هر یک از شماره تلفنها در یک ویژگی مجزا ذخیره شود:



-ویژگی محاسباتی: ویژگیی است که مقدار آن با توجه به مقدار ویژگیهای دیگر قابل محاسبه است. مثلاً مقدار ویژگی سن را می‌توان بر اساس مقدار ویژگی تاریخ تولد محاسبه کرد. ویژگی محاسباتی با خط چین به موجودیت وصل می‌شود.



تذکر: برای جلوگیری از پیچیدگی نمودارهای ER غالباً از نمایش ویژگیها خودداری می‌شود.

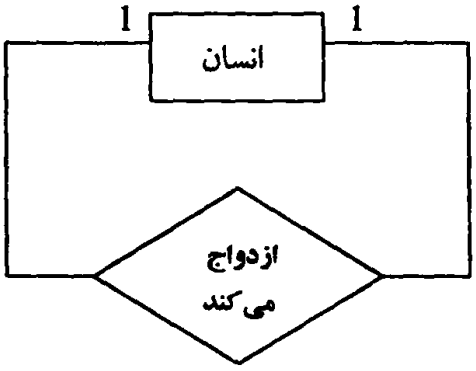
۳-۱-۲-٦) رابطه‌ها[1] : رابطه‌ها نحوهٔ ارتباط موجودیتها در سیستم را نشان می‌دهند مثلاً دانشجویان درسها را می‌گیرند. برای نمایش رابطه‌ها از لوزی استفاده می‌شود.

تذکر: مفهوم رابطه در نمودارهای ER با مفهوم رابطه در پایگاه داده‌های رابطه‌ای(که با مفهوم جدول هم ارز است) متفاوت است.

۱-۳-۱-۲-٦) انواع رابطه‌ها

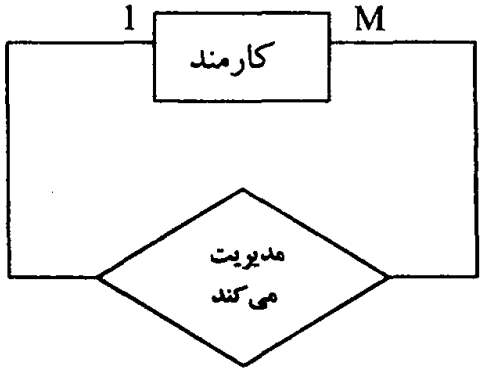
- **رابطهٔ یکتایی:** رابطهٔ یک نوع موجودیت است با همان نوع موجودیت مثل رابطهٔ درس با درس، رابطهٔ دانشجو با دانشجو و ...

مثال ۱: رابطهٔ ازدواج در برخی کشورها را در نظر بگیرید. هر انسان می‌تواند تنها با یک انسان دیگر ازدواج کند. انسان طرف مقابل نیز می‌تواند تنها با یک انسان ازدواج کند. پس این رابطه، یکتایی و یک-به-یک(1:1) است.

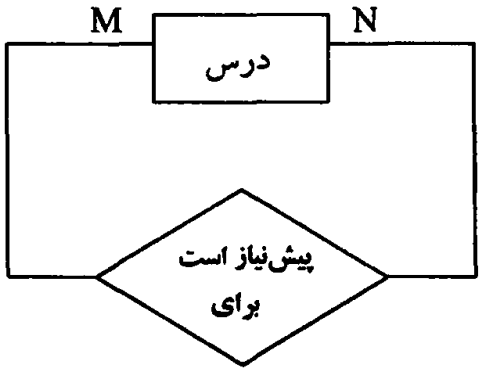


1 - Relationships

مثال ۲: رابطهٔ مدیریت در یک سازمان را در نظر بگیرید. طبق قوانین این سازمان، هر کارمند می‌تواند مدیریت چند کارمند دیگر را بر عهده داشته باشد ولی هر کارمند تنها یک مدیر دارد. این رابطه، یک رابطهٔ یکتایی یک-به-چند(1:M) است.



مثال ۳: رابطهٔ پیش‌نیاز بودن را در نظر بگیرید. یک درس می‌تواند چند پیش‌نیاز داشته باشد. درس طرف مقابل(درس پیش‌نیاز) نیز می‌تواند پیش‌نیاز چند درس باشد. پس رابطهٔ پیش‌نیاز بودن، یک رابطهٔ یکتایی چند-به-چند(M:N) است.

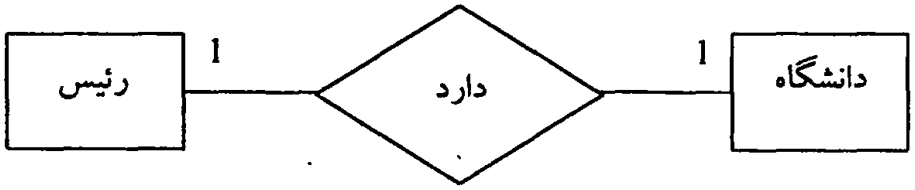


-رابطهٔ دوتایی[1] : رابطهٔ دوتایی، رابطهٔ بین دو نوع موجودیت مختلف است مثل رابطهٔ دانشجو با درس، رابطهٔ درس با استاد و ...

مثال ۱: رابطهٔ ریاست یک دانشگاه را در نظر بگیرید. هر دانشگاه یک رئیس دارد و هر رئیس تنها ریاست یک دانشگاه را بر عهده دارد. این رابطه دوتایی و 1:1 است.

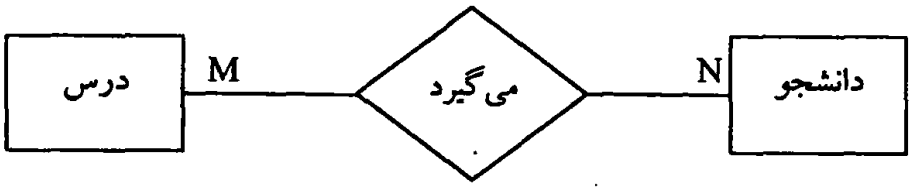
---

1 - Binary



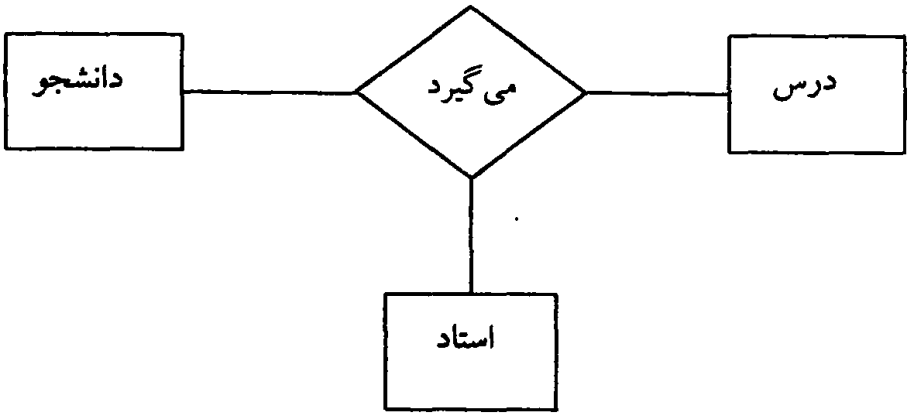
مثال ۲: طبق قوانین یک نمایشگاه نقاشی، هر نقاش می‌تواند چندین تابلو ارائه کند ولی هر تابلو تنها توسط یک نقاش ارائه می‌شود. این رابطه، دوتایی و 1:M است.

مثال۳: رابطهٔ دانشجو با درس را در نظر بگیرید. هر دانشجو می‌تواند چندین درس را بگیرد و هر درس توسط چندین دانشجو اخذ می‌شود. این رابطه یک رابطهٔ دوتایی M:N است.



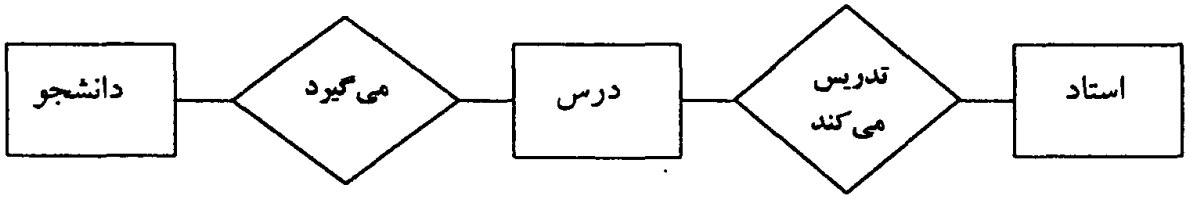
رابطهٔ n- تایی[1] (n>2): رابطهٔ بین n نوع موجودیت مختلف است.

مثال: دانشجو یک درس را با یک استاد می‌گیرد:



1 - n-ary

تذکر : در نمودارهای ER اولیه، تنها روابط یکتایی و دوتایی وجود داشتند و طراح موظف بود کلیهٔ روابط n-تایی با $n>2$ را به روابط دوتایی بشکند. مثلاً رابطهٔ سه تایی دانشجو-درس-استاد را می‌توان به دو رابطهٔ دوتایی دانشجو-درس و درس-استاد شکست:



اگر چه تجزیهٔ روابط n-تایی به روابط دوتایی باعث سادگی مدل می‌شود ولی گاهی نیز مدل را با برخی نواقص و ناسازگاریها مواجه می‌کند. برای رفع این نواقص، در نمودارهای ER پیشرفته برای روابط n-تایی با $n>2$ نیز قوانینی در نظر گرفته شد. با این حال، تا حد امکان باید این روابط را به روابط دوتایی شکست و تنها در صورت ضرورت از آنها استفاده کرد.

۲-۳-۱-۲-۶) مفاهیم مربوط به رابطه‌ها :

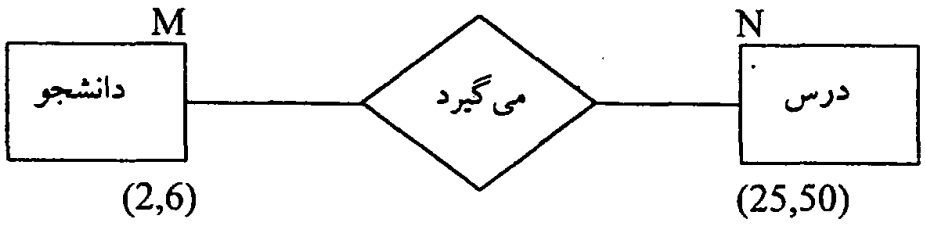
-**درجهٔ رابطه**[1] : مشخص می‌کند یک رابطه 1:1، 1:M یا M:N است.

- **کاردینالی رابطه**[2]: مشخص می‌کند در یک رابطه هر موجودیت با حداقل و حداکثر چند موجودیت از طرف مقابل در ارتباط است.

مثال ۱: طبق قوانین یک دانشگاه:

-هر دانشجو در طول ترم باید حداقل در ۲ و حداکثر در ۶ درس ثبت نام کند.

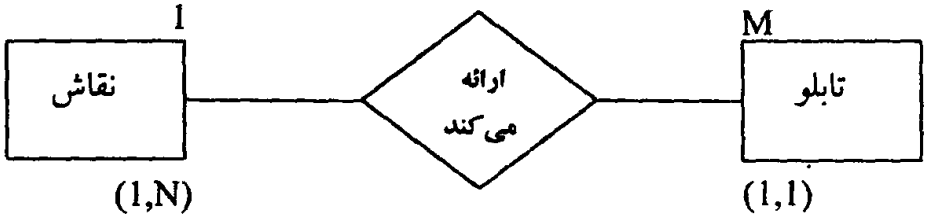
-در هر درس باید حداقل ۲۵ و حداکثر ۵۰ دانشجو ثبت نام کنند.



1 - Relationship Degree
2 - Relationship Cardinality

نکته: توجه کنید که درجهٔ یک رابطه با توجه به حداکثر کاردینالیتیها نیز قابل تشخیص است. در این مثال کاردینالیتی دانشجو (2,6) است. چون حداکثر کاردینالیتی بزرگتر از ۱ است، طرف درس "چند" می‌شود. از طرف دیگر، کاردینایتی درس (25,50) است. چون حداکثر کاردینالیتی بزرگتر از ۱ است، طرف دانشجو نیز "چند" می‌شود. پس این رابطه M:N است.

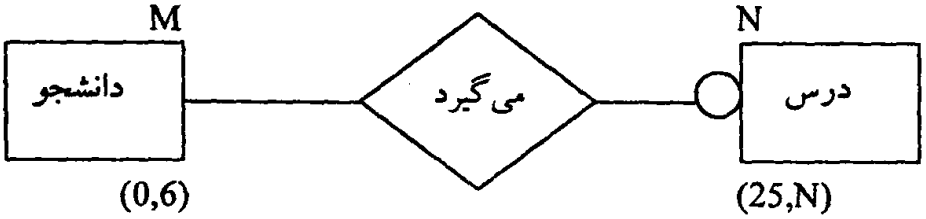
مثال ۲: طبق قوانین یک نمایشگاه نقاشی، هر تابلو توسط یک نقاش ارائه می‌شود و هر نقاش می‌تواند چندین تابلو ارائه کند.



در اینجا حداکثر تعداد تابلوهایی که هر نقاش می‌تواند ارائه کند مشخص نشده است ولی طبق قوانین سیستم می‌تواند بیشتر از ۱ باشد پس با N مشخص شده است. چون حداکثر کاردینالیتی در طرف نقاش بزرگتر از ۱ است، طرف تابلو چند خواهد بود. از طرف دیگر، چون حداکثر کاردینالیتی در طرف تابلو ۱ است، طرف نقاش ۱ می‌شود. پس این رابطه M:1 است.

- **مشارکت در رابطه**[1]: مشخص می‌کند آیا شرکت یک موجودیت در یک رابطه اجباری است یا اختیاری.

مثال ۱: طبق قوانین یک دانشگاه هر دانشجو می‌تواند حداکثر در ۶ درس ثبت نام کند و یا در هیچ درسی ثبت نام نکند. در هر درس حداقل ۲۵ دانشجو باید ثبت نام کنند.



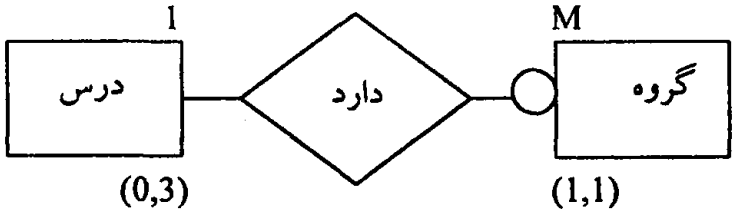
1 - Relationship Participation

دانشجو اجباری ندارد که حتماً درس بگیرد پس موجودیت درس برای موجودیت دانشجو اختیاری است و در نمودار با یک دایره مشخص می‌شود. از طرف دیگر، در هر درس باید حداقل ۲۵ دانشجو ثبت نام کنند پس وجود دانشجو برای درس اجباری است.

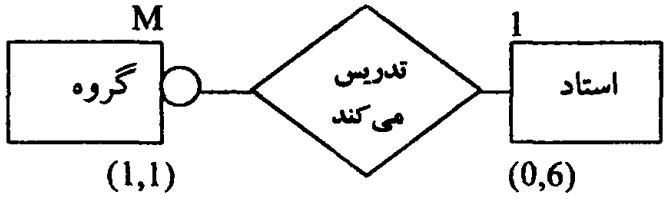
نکته: اگر در یک رابطه کاردینالیتی حداقل برای یک موجودیت صفر باشد، موجودیت طرف مقابل اختیاری خواهد بود.

مثال ۲: طبق قوانین یک دانشگاه در طول یک ترم:

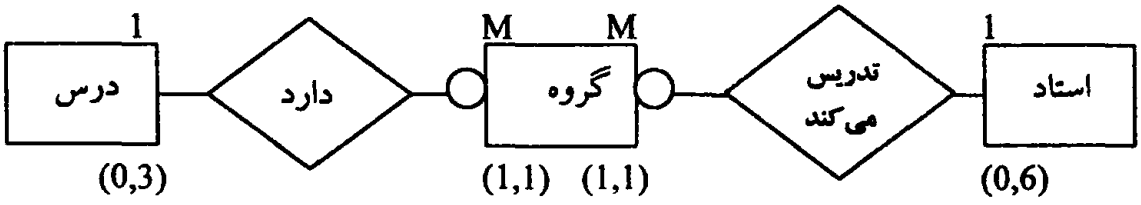
-برای هر درس ممکن است هیچ گروهی ارائه نشود یا حداکثر ۳ گروه ارائه شود.



-هر گروه درسی توسط یک استاد ارائه می‌شود ولی هر استاد مجاز است حداکثر ۶ گروه درسی را تدریس کند.



نمودار کلی این سیستم به شکل ذیل خواهد بود:

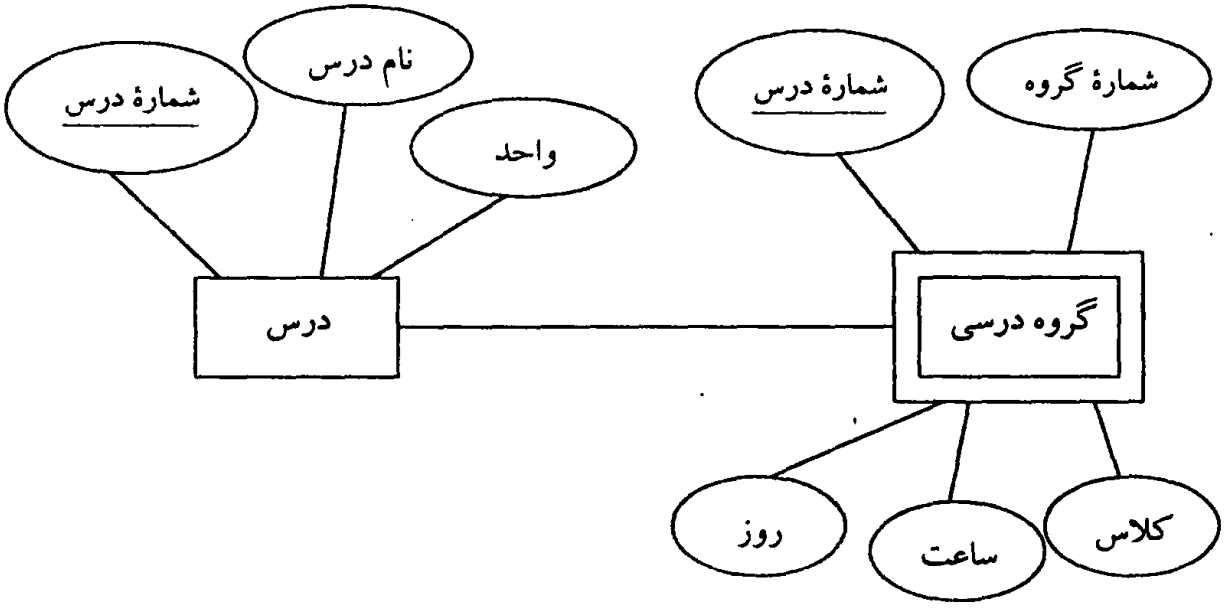


۲-۲-۶) انواع موجودیتها

- **موجودیت قوی:** موجودیتی است که وجود آن بستگی به وجود موجودیتهای دیگر ندارد مثل دانشجو و درس.

- **موجودیت ضعیف:** موجودیتی است که وجود آن وابسته به وجود موجودیت دیگر(موجودیت قوی) می‌باشد و کلید اصلی موجودیت قوی جزئی از کلید اصلی آن است. برای نمایش موجودیت ضعیف از دو مستطیل تو در تو استفاده می‌شود.

مثال: اگر یک درس را حذف کنیم، کلیهٔ گروههای آن نیز حذف خواهند شد. پس گروه درسی، یک موجودیت ضعیف است که وجود آن بستگی به وجود موجودیت درس دارد. این مطلب را بصورت ذیل نشان می‌دهیم:



برای درک بهتر مطلب به جداولی که در نهایت از این نمودار استخراج می‌شوند توجه کنید:

course

c#	cname	unit
1400	پایگاه داده‌ها	3
1500	ادبیات	2

group

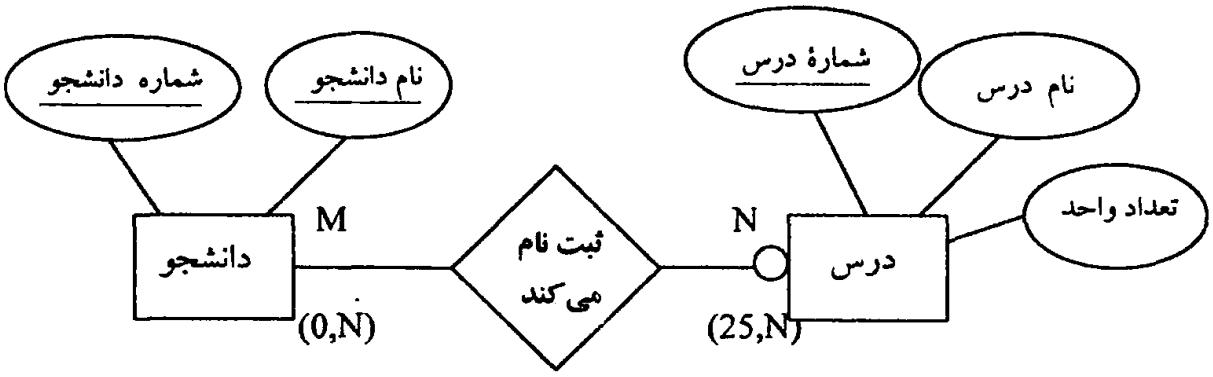
c#	group#	day	time	class
1400	1	دوشنبه	8	315
1400	2	دوشنبه	11	112
1500	1	سه شنبه	8	114

- **موجودیت بازگشتی**[1] : موجودیتی است که با خودش در ارتباط است مثل درس که پیش‌نیاز است برای درس دیگر.

-**موجودیت مرکب**[2] : درنمودارهای ER روابط چند به چند را باید به روابط یک به چند تبدیل کرد. برای این کار، رابطه را با یک موجودیت مرکب جایگزین می‌کنیم.

مثال: طبق قوانین یک دانشگاه:

- هر دانشجو ممکن است چندین درس را گرفته باشد و یا هنوز هیچ درسی نگرفته باشد.
- در هر درس حداقل ۲۵ دانشجو باید ثبت نام کنند.

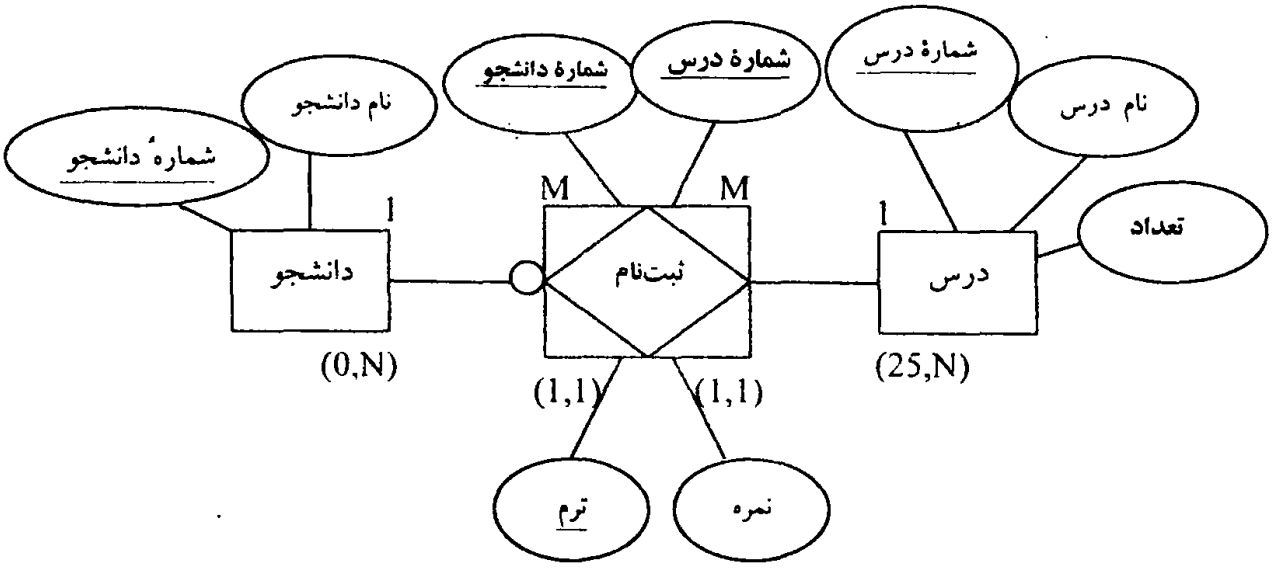


رابطهٔ میان درس و دانشجو از نوع M:N است پس لازم است این رابطه را به دو رابطهٔ 1:M بشکنیم. برای این کار کافیست رابطهٔ "ثبت نام می‌کند" را با یک موجودیت مرکب بنام "ثبت نام" جایگزین کنیم. کلید اصلی هر دو موجودیت درس و دانشجو باید جزء ویژگیهای این موجودیت باشند. به این ترتیب، رابطهٔ M:N میان درس و دانشجو به دو رابطهٔ 1:M شکسته می‌شود: یک رابطهٔ 1:M میان درس و ثبت نام و یک رابطهٔ 1:M میان دانشجو و ثبت نام.

---

1 - Recursive Entity

2 - Composite Entity



برای درک بهتر مطلب، به ساختار جداولی که در نهایت از این نمودار استخراج می‌شوند توجه کنید:

st

st#	sname
8001	آرش راد
8101	ساغر راد
8102	علی نیکی
8111	بابک یاری

enroll

st#	crs#	term	grade
8001	1400	811	9
8001	1400	812	13
8001	1407	811	13
8101	1400	821	19
8111	1407	821	12

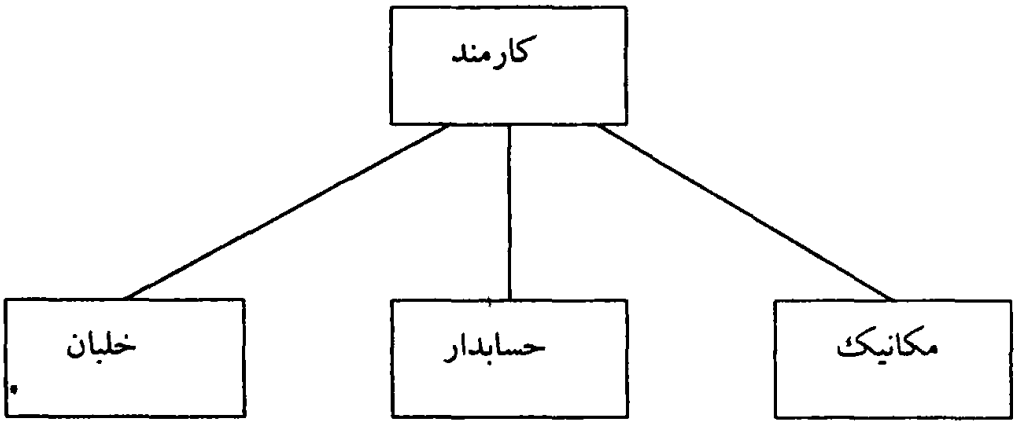
course

crs#	cname	unit
1100	ادبیات	2
1105	تربیت بدنی	1
1400	برنامه سازی۱	3
1407	پایگاه داده‌ها	3

هر ثبت نام(هر تاپل از جدول enroll) مربوط به یک و تنها یک دانشجو است(1,1) . طبق قوانین دانشگاه ممکن است یک دانشجو در هیچ یک از این ثبت نامها وجود نداشته باشد و یا در چندین ثبت نام وجود داشته باشد(0,M). بنابراین، رابطهٔ میان دانشجو و ثبت نام 1:M و طرف ثبت نام اختیاری است.

از طرف دیگر، هر ثبت نام(هر تاپل از جدول enroll) مربوط به یک و تنها یک درس است (1,1) و طبق قوانین حاکم بر این دانشگاه، برای هر درس حداقل ۲۵ ثبت نام وجود دارد(25,N). بنابراین، رابطهٔ میان درس و ثبت نام 1:M و هر دو طرف اجباری است.

۳-۲-٦) نوع و زیر نوعهای موجودیت[1] : اغلب میان موجودیتها، رابطهٔ سلسله‌مراتبی وجود دارد. به عنوان مثال فرض کنید کارمندان یک شرکت به سه گروه خلبان، مکانیک و حسابدار تقسیم شوند. در این صورت، موجودیت کارمند یک ابرنوع برای سه زیرنوع خلبان، مکانیک و حسابدار محسوب می‌شود. این رابطه را بصورت ذیل نمایش می‌دهیم:



برای تبدیل یک رابطهٔ ابرنوع-زیرنوع به ساختار جداول، ویژگیهایی را که میان کلیهٔ زیرنوعها مشترک هستند درجدول مربوط به موجودیت ابرنوع و ویژگیهای خاص هر موجودیت زیرنوع را در جدول مربوطه قرار داده، کلید اصلی جدول مربوط به موجودیت ابرنوع را در جداول زیرنوع قرار می‌دهیم. به عنوان مثال جداول مربوط به موجودیهای کارمند و خلبان می‌توانند به شکل ذیل باشند:

employee

emp#	name	Start_date
100	آرش راد	۷۸/۱۰/۱۰
101	علی رضایی	۷۹/۱۰/۱۵
102	سیاوش آزاد	۷۹/۱۱/۲۰
103	عسل رضایی	۸۲/۱۲/۲۰
104	علی تقوی	۷۵/۱۰/۱۰

1. Entity Supertypes and Subtypes

Pilot

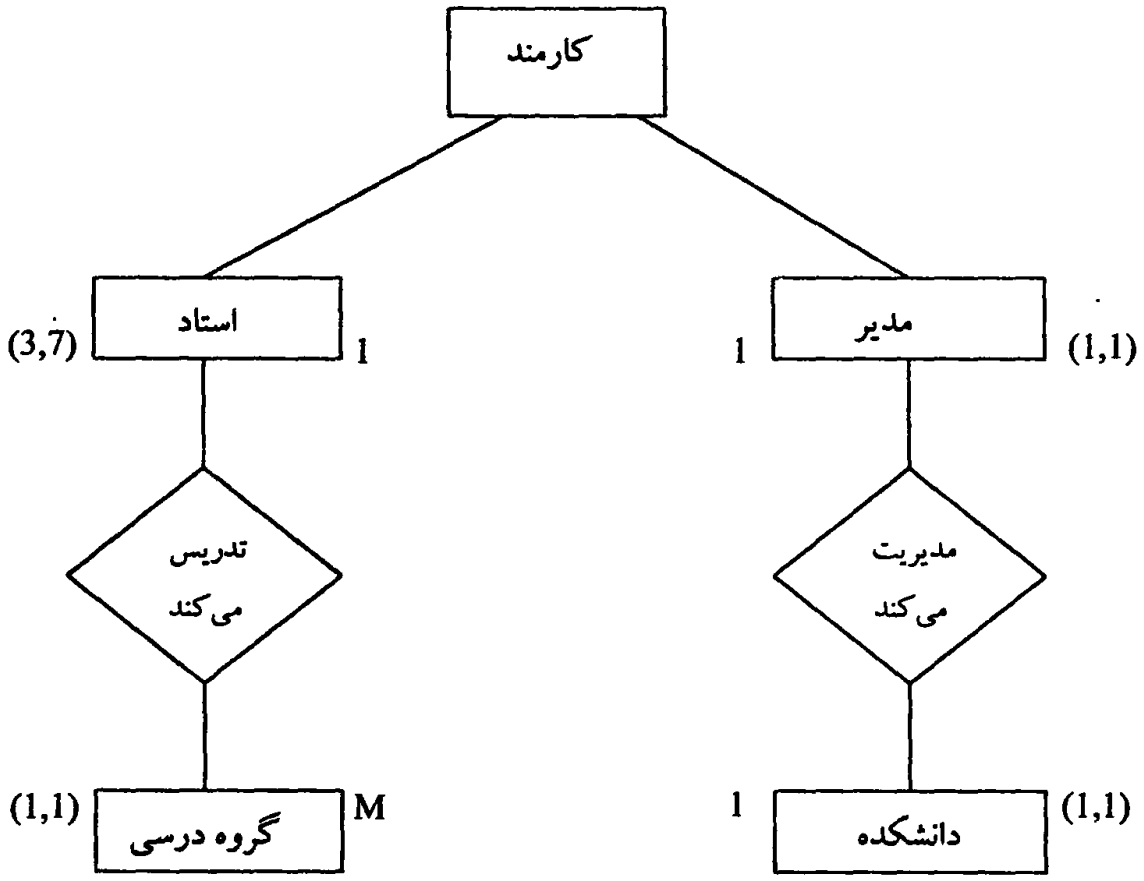
emp#	Pilot-license	Fly_count
100	ATP	150
102	ATP	167
104	COM	210

همانگونه که مشاهده می‌کنید ویژگیهای مشترک میان همهٔ کارمندان (شمارهٔ کارمندی، نام، تاریخ استخدام) در جدول employee و ویژگیهای خاص خلبانان (مدرک خلبانی و تعداد پرواز) در جدول pilot قرار گرفته‌اند.

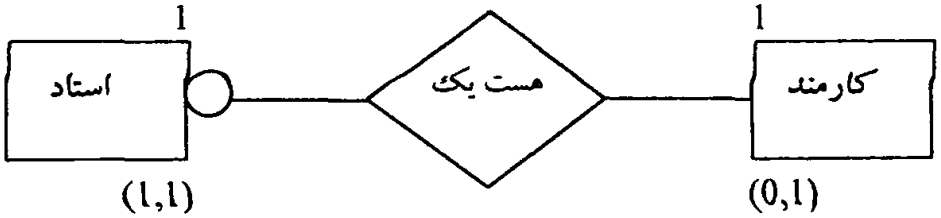
مثال: طبق قوانین یک دانشگاه:

- کارمندان شامل اساتید و مدیران دانشکده‌ها می‌باشند.
- هر استاد حداقل ۳ و حداکثر ۷ گروه درسی را تدریس می‌کند.
- مدیریت هر دانشکده تنها به یک مدیر سپرده می‌شود وهر مدیر تنها مدیریت یک دانشکده را بر عهده دارد.

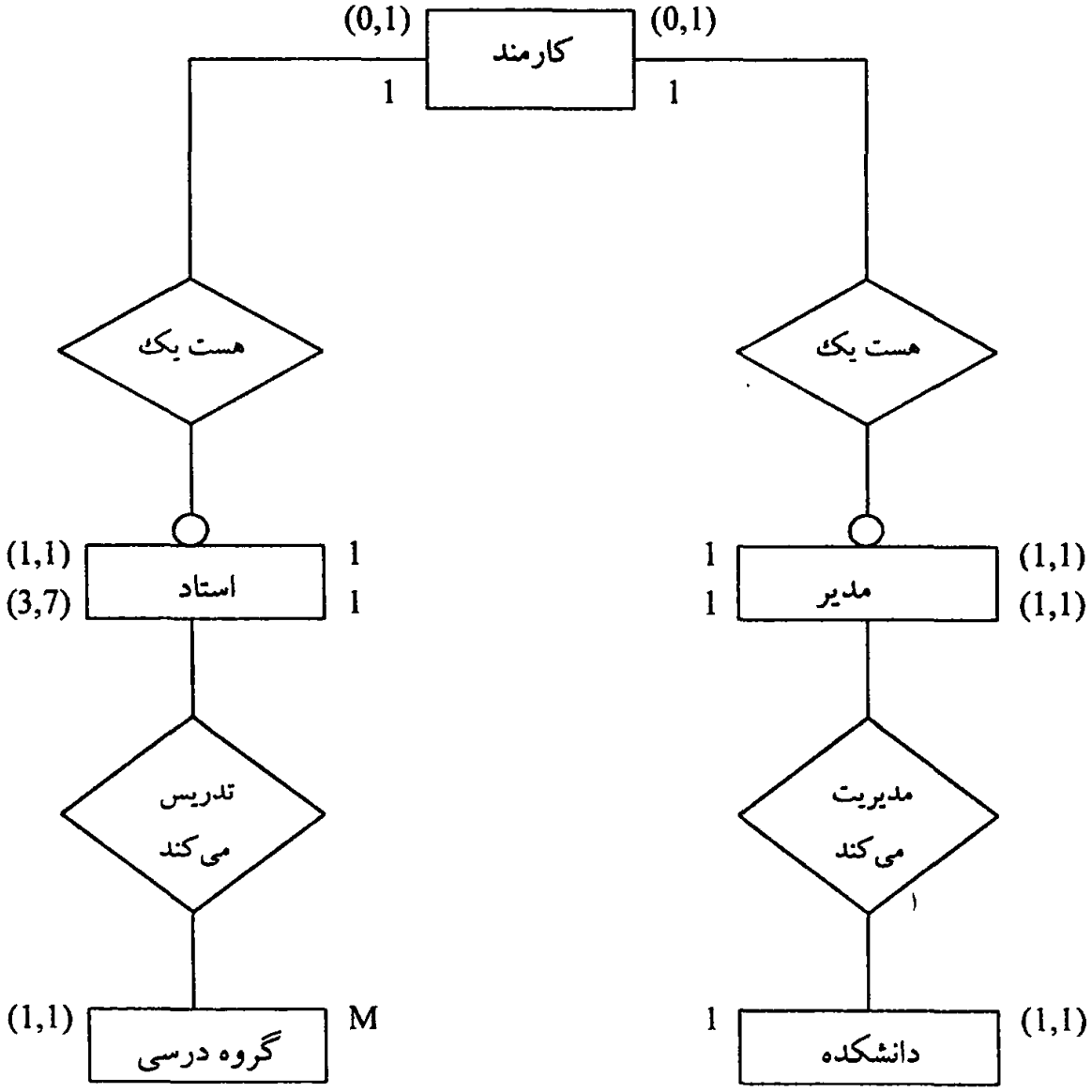
شکل ذیل، نمودار ER این سیستم را نشان می‌دهد:



تذکر : هیچ اجباری برای در نظر گرفتن روابط ابرنوع-زیرنوع وجود ندارد. به عنوان مثال رابطهٔ ابرنوع-زیرنوع کارمند- استاد را می‌توان به شکل ذیل نشان داد:



این رابطه نشان می‌دهد که یک استاد لزوماً یک کارمند است ولی هر کارمند لزوماً استاد نیست. بنابراین، نمودار ER این مثال را می‌توان به شکل ذیل نیز رسم کرد:



٤-۲-٦) نگاشت یک نمودار ER به جداول پایگاه داده‌ها

برای نگاشت یک نمودار ER به جداول پایگاه داده‌ها مراحل ذیل را دنبال کنید:

قدم ۱: برای هر یک از موجودیتهای قوی، یک جدول در نظر بگیرید. برای هر یک از ویژگیهای موجودیت یک ستون در نظر بگیرید و کلید اصلی جدول را مشخص کنید.

قدم ۲: برای هر یک از موجودیتهای ضعیف، یک جدول در نظر بگیرید و کلید اصلی جدول را مشخص کنید. (کلید اصلی موجودیت قوی باید جزئی از کلید اصلی در موجودیت ضعیف باشد)

قدم ۳: برای هر یک از موجودیتهای مرکب، یک جدول در نظر بگیرید و کلید اصلی جدول را مشخص کنید. (کلید اصلی موجودیتهای دو طرف موجودیت مرکب حتماً باید جزء ویژگیهای این جدول باشند)

قدم ٤: برای هر رابطهٔ دوتایی 1:1 :

- چنانچه هر دو طرف رابطه اجباری و یا هر دو طرف اختیاری است، به دلخواه خود کلید اصلی یکی از جداول را به ویژگیهای جدول طرف مقابل اضافه کنید.
- چنانچه یک طرف رابطه اجباری و طرف دیگر اختیاری است، کلید اصلی جدول طرف اجباری را به ویژگیهای جدول طرف اختیاری اضافه کنید.

قدم ٥: برای هر رابطهٔ دوتایی 1:M، کلید اصلی جدول طرف ۱ را به ویژگیهای جدول طرف M اضافه کنید.

قدم ٦: برای هر رابطهٔ یکتایی 1:1 یا 1:M کلید اصلی موجودیت بازگشتی را با یک نام دیگر به ویژگیهای جدول اضافه کنید(به عنوان کلید خارجی)!!!

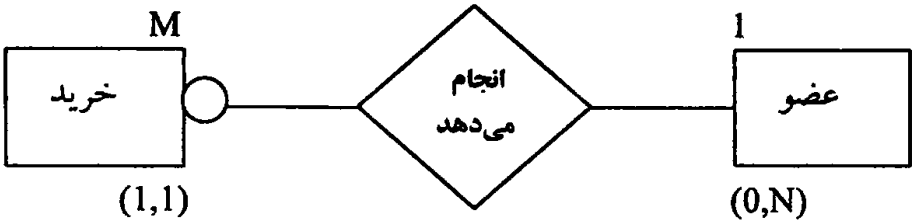
قدم ۷: برای هر رابطهٔ n-تایی با n>2 جدولی در نظر بگیرید که شامل کلیدهای اصلی کلیهٔ موجودیتهای شرکت کننده در رابطه باشد.

قدم ۸: جداولی را که تنها یک ستون دارند و یا کلیهٔ اطلاعات موجود در آنها در جدول دیگری نیز موجود است حذف کنید.

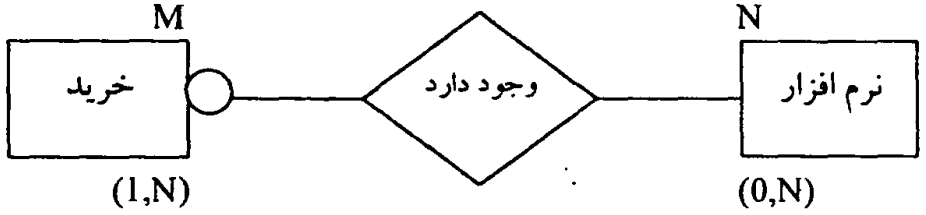
تذکر: اگر چه برای انجام مراحل فوق ترتیب خاصی در نظر گرفتیم، نگاشت یک نمودار ER به جداول پایگاه داده‌ها یک عملیات کاملاً ترتیبی نیست مثلاً ممکن است پس از انجام قدم ۴ یا ۵ مجبور باشید کلیدهای اصلی تعیین شده در قدمهای قبل را تغییر دهید.

مثال ۱: یک فروشگاه الکترونیکی را در نظر بگیرید که محصولات نرم افزاری می‌فروشد. این سایت تعدادی عضو دارد که با وارد کردن نام کاربری و کلمهٔ عبور خود وارد سایت شده، محصولات نرم افزاری موردنظر خود را انتخاب و خریداری می‌کنند. در این سیستم قوانین ذیل حاکم است:

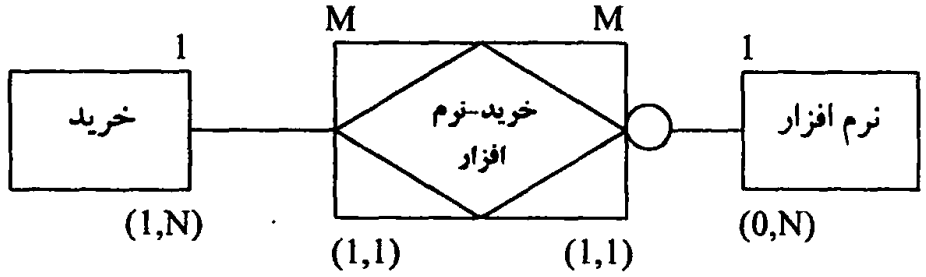
قانون ۱: هر عضو می‌تواند به کرات از سایت خرید کند و یا هیچ خریدی انجام ندهد ولی هر خرید تنها مربوط به یک عضو است.



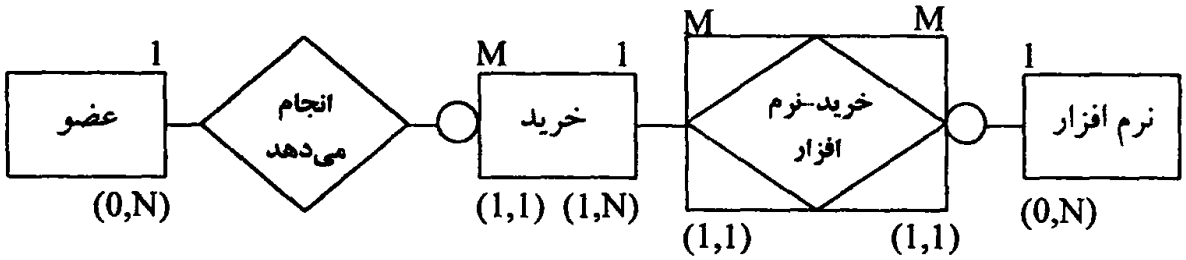
قانون ۲: در هر خرید ممکن است نرم افزارهای متعددی خریداری شود و مسلماً هر نرم‌افزار نیز ممکن است به کرات خریداری شود (مثلاً در خرید شمارهٔ ۱۰ ممکن است دو نسخه از نرم افزار فتوشاپ و یک نسخه از نرم افزار اتوکد خریداری شود. همچنین ممکن است مشتریان زیادی فتوشاپ را خریداری کنند و یا هیچ کس آن را خریداری نکند)



این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:



نمودار کلی:



تعیین ساختار جداول پایگاه داده‌ها:

موجودیت عضو تبدیل به جدول عضو می‌شود. این موجودیت، شامل ویژگیهای نام کاربری و کلمهٔ عبور است. تنها رابطه‌ای که این موجودیت در آن شرکت کرده است، رابطهٔ "انجام می‌دهد" است و چون در طرف ۱ این رابطه قرار دارد، لازم نیست کلید اصلی موجودیت خرید را به ویژگیهای آن اضافه کنیم. کلید اصلی این جدول نام کاربری است:

عضو (نام کاربری- کلمه عبور)

موجودیت خرید تبدیل به جدول خرید می‌شود. موجودیت خرید تنها در رابطهٔ "انجام می‌دهد" شرکت دارد و چون در طرف M این رابطه قرار دارد، باید کلید اصلی موجودیت طرف مقابل

یعنی عضو را به ویژگیهای آن اضافه کنیم. می‌توانیم به هر خرید یک شمارهٔ منحصر بفرد اختصاص دهیم و از این شماره به عنوان کلید اصلی استفاده کنیم:

خرید (شمارهٔ خرید- نام کاربری- تاریخ خرید و ...)

موجودیت نرم افزار تبدیل به جدول نرم افزار می‌شود. از آنجا که این موجودیت در هیچ رابطه‌ای شرکت ندارد، تنها ویژگیهای خاص هر نرم افزار را در آن قرار می‌دهیم. به هر نرم افزار یک شمارهٔ منحصر بفرد اختصاص می‌دهیم و از آن به عنوان کلید اصلی استفاده می‌کنیم:

نرم افزار(شمارهٔ نرم افزار- نام نرم افزار- قیمت – موجودی در انبار)

موجودیت مرکب خرید-نرم افزار نیز تبدیل به یک جدول می‌شود. هم کلید اصلی خرید و هم کلید اصلی نرم افزار باید جزء ویژگیهای این جدول باشند. از ترکیب این دو ویژگی می‌توان به عنوان کلید اصلی جدول استفاده کرد چون امکان تکراری بودن مقادیر آن وجود ندارد:

خرید-نرم افزار(شمارهٔ خرید- شمارهٔ نرم افزار-تعداد خریداری شده)

برای درک بهتر مطلب به نمونهٔ این جداول توجه کنید:

User

userId	password
aftab	Aftab57
mina	crazy
Rasool	Rasool2000

Software

software#	sname	fee	count
100	Adobe photoshop 7.0	2000	25
101	Visual studio.net	8000	30
102	دیکشنری فارسی انگلیسی آریانپور	10000	21

factor

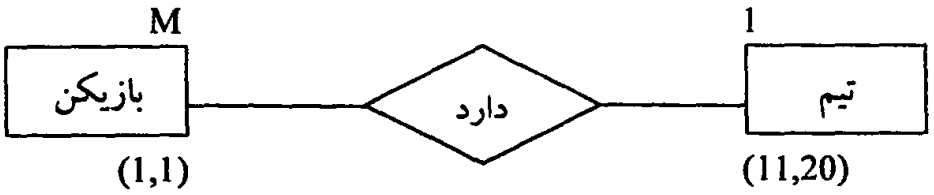
factor#	userId	fdate
1	Aftab	۸۳/۰۵/۰۸
2	Aftab	۸۳/۰۵/۱۰
3	rasool	۸۳/۰۵/۱۰

factor_software

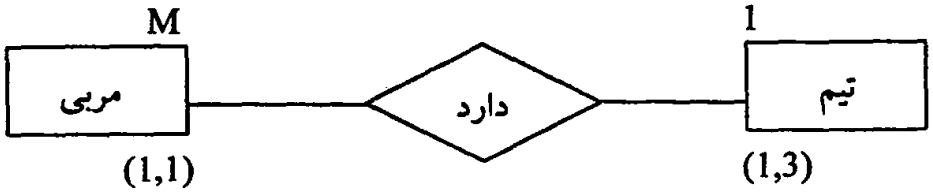
factor#	Software#	qty
1	100	1
1	101	2
1	102	1
2	101	3
3	100	1

مثال ۲: فرض کنید در بازیهای استانی فوتبال قوانین ذیل حاکم باشد:

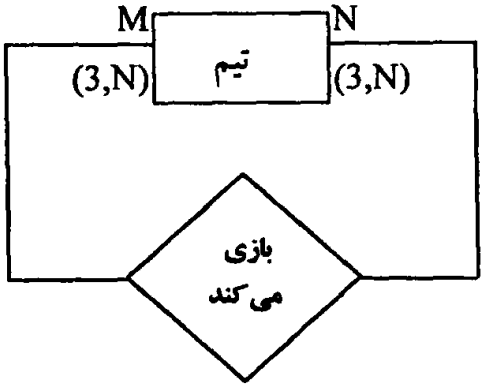
قانون ۱: هر تیم حداقل ۱۱ و حداکثر ۲۰ بازیکن دارد ولی هر بازیکن تنها در یک تیم بازی می‌کند.



قانون ۲: هر تیم حداقل یک و حداکثر ۳ مربی دارد و هر مربی تنها مربی‌گری یک تیم را برعهده دارد.

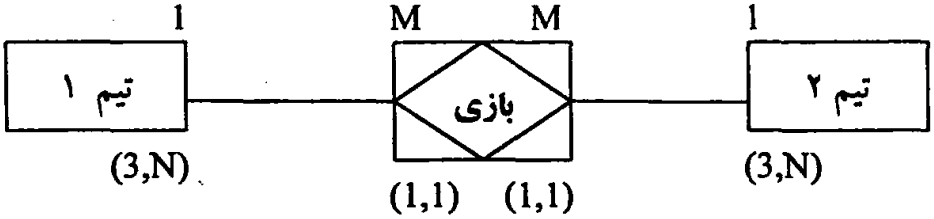


قانون ۳: هر تیم حداقل با سه تیم دیگر بازی می‌کند.

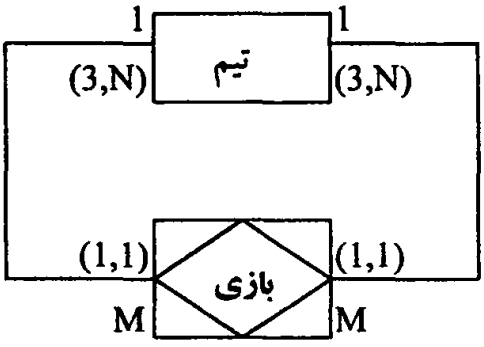


این رابطه M:N است و لازم است به دو رابطهٔ 1:M شکسته شود. برای درک بهتر مطلب، این رابطه را بصورت باز در نظر بگیرید:

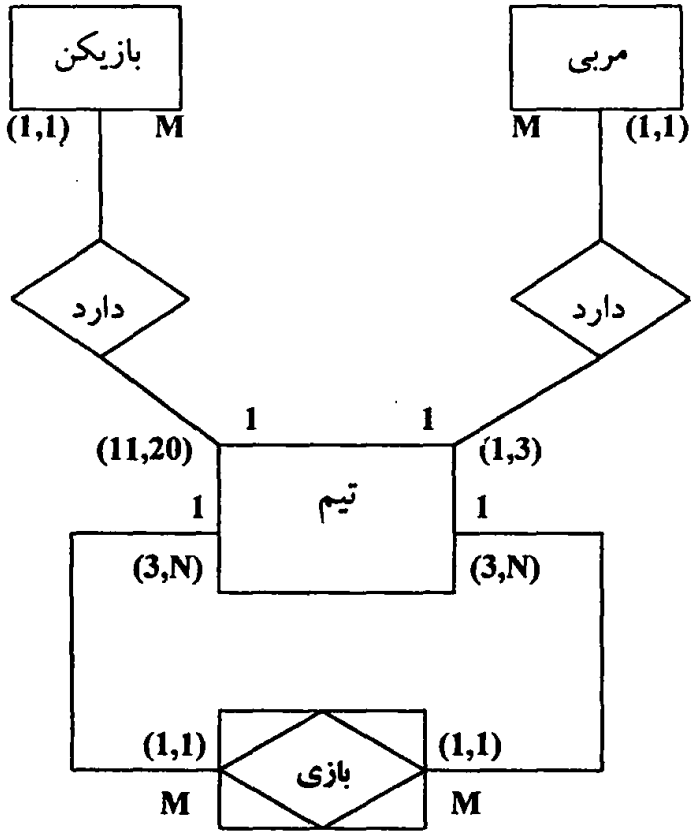
برای تبدیل این رابطهٔ M:N به دو رابطهٔ 1:M باید رابطهٔ "بازی می‌کند" را با یک موجودیت مرکب به نام "بازی" جایگزین کنیم. این موجودیت مرکب باید شامل کلید اصلی تیم ۱ و کلید اصلی تیم ۲ باشد:



این رابطه را بصورت ذیل نمایش می‌دهیم:



نمودار کلی به شکل ذیل خواهد بود:



تعیین ساختار جداول پایگاه داده‌ها:

موجودیت تیم به جدول تیم نگاشت می‌شود. این موجودیت با دو موجودیت رابطه دارد: بازیکن و مربی.

- در رابطه با بازیکن، تیم در طرف ۱ رابطه است پس لازم نیست کلید اصلی بازیکن را به ویژگیهای تیم اضافه کنیم.
- در رابطه با مربی نیز تیم در طرف ۱ رابطه است پس لازم نیست کلید اصلی مربی را به ویژگیهای تیم اضافه کنیم.

تیم(<u>شمارهٔ تیم</u>- نام تیم-...)

موجودیت بازیکن به جدول بازیکن نگاشت می‌شود. این موجودیت با تیم رابطه دارد و در طرف M این رابطه قرار دارد پس باید کلید اصلی تیم را به ویژگیهای آن اضافه کنیم. در هر تیم به هر بازیکن یک شمارهٔ منحصر بفرد داده‌ایم(می‌توانیم بطور کلی به هر یک از بازیکنها صرفنظر از

تیمی که در آن بازی می‌کنند یک شمارهٔ منحصر بفرد اختصاص دهیم. در اینصورت، شمارهٔ بازیکن به تنهایی کلید اصلی خواهد بود) :

بازیکن(شمارهٔ تیم- شمارهٔ بازیکن-نام بازیکن-...)

موجودیت مربی به جدول مربی نگاشت می‌شود. این موجودیت با تیم رابطه دارد و در طرف M این رابطه قرار دارد پس باید کلید اصلی تیم را به ویژگیهای آن اضافه کنیم. به هر مربی یک شمارهٔ منحصر بفرد اختصاص می‌دهیم:

مربی(شمارهٔ مربی- شمارهٔ تیم-نام مربی-...)

موجودیت مرکب بازی به جدول بازی نگاشت می‌شود. به هر بازی یک شمارهٔ منحصر بفرد اختصاص می‌دهیم:

بازی(شمارهٔ بازی- شمارهٔ تیم۱- شمارهٔ تیم۲- تاریخ بازی -...)

برای درک بهتر مطلب به نمونهٔ این جداول توجه کنید:

Team

team#	tname
1	امید
2	جوانان
3	فردا
4	پیروزی

Player

team#	player#	pname
1	1	کیومرث تابش
1	10	رسول صادقی
2	1	امید پهلوان
2	6	سیروس آزاد

Coatch

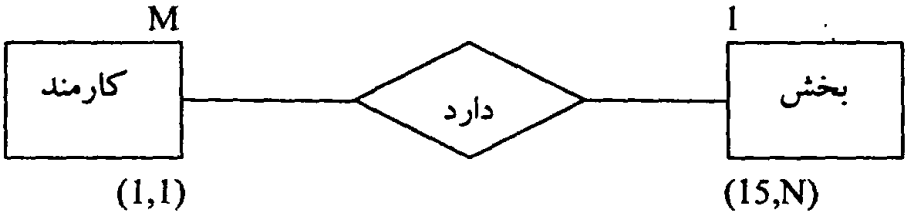
coatch#	team#	Cname
100	1	آرش راد
101	2	بابک کیانی
102	3	تقی فرحی
103	4	رضا راد

Game

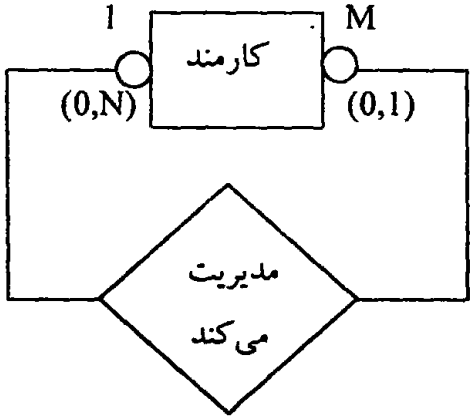
game#	team1#	team2#	gdate
1	1	4	۸۳/۰۵/۰۵
2	1	2	۸۳/۰۵/۰۶
3	2	4	۸۳/۰۵/۱۰
4	3	2	۸۳/۰۵/۱۱

مثال ۳: در یک سازمان قوانین ذیل حاکم است:

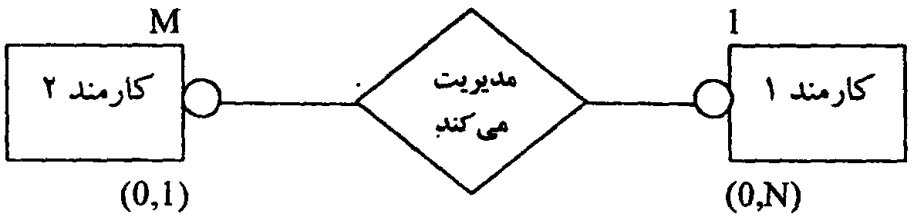
-هر کارمند در یک بخش کار می‌کند ولی در هر بخش حداقل ۱۵ کارمند وجود دارد.



-ممکن است یک کارمند مدیریت چند کارمند دیگر را بر عهده داشته باشد ولی هر کارمند حداکثر یک مدیر دارد.



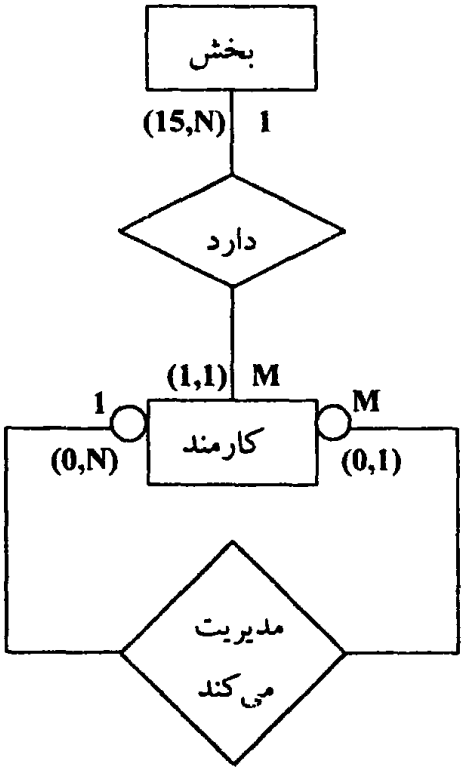
برای درک بهتر مطلب، این رابطه را بصورت باز در نظر بگیرید:



هنگام تبدیل این نمودار به ساختار جداول باید کلید اصلی کارمند ۱ را به ویژگیهای جدول طرف M یعنی کارمند ۲ اضافه کنیم. از آنجا که این رابطه یکتایی است و موجودیتهای کارمند ۱ و کارمند ۲ بر هم منطبق هستند و در نهایت برای موجودیت کارمند تنها یک جدول در نظر

می‌گیریم، در واقع کلید اصلی جدول کارمند را به عنوان کلید خارجی به ویژگیهای همین جدول اضافه می‌کنیم.

نمودار کلی:



تبدیل به ساختار جداول:

موجودیت بخش تبدیل به جدول بخش می‌شود. موجودیت بخش تنها در یک رابطه شرکت دارد و در طرف ۱ این رابطه قرار دارد پس لازم نیست کلید اصلی موجودیت طرف مقابل را به ویژگیهای آن اضافه کنیم. به هر بخش یک شمارهٔ منحصر بفرد می‌دهیم:

بخش(شمارهٔ بخش-نام بخش)

موجودیت کارمند تبدیل به جدول کارمند می‌شود. به هر کارمند یک شمارهٔ منحصر بفرد داده، از آن به عنوان کلید اصلی استفاده می‌کنیم. چون موجودیت کارمند در یک رابطهٔ یکتایی 1:M شرکت دارد، کلید اصلی آن را با یک نام دیگر(شمارهٔ مدیر) به ویژگیهای آن اضافه می‌کنیم:

کارمند(شمارهٔ کارمند-نام کارمند-شمارهٔ مدیر)

برای درک بهتر مطلب به نمونهٔ این جداول توجه کنید:

department

dep#	dname
1	اداری
2	فنی
3	مالی

employee

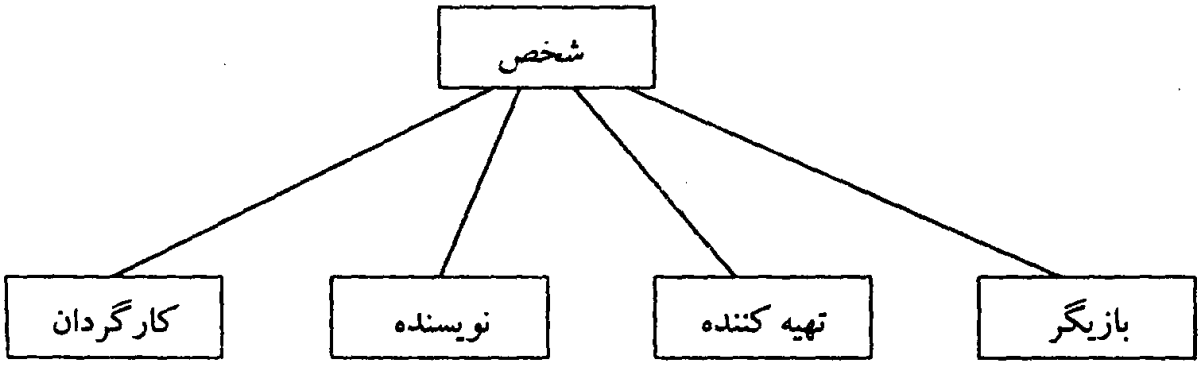
emp#	name	manager#
100	آرش راد	102
101	حسن غفوری	102
102	ایرج فرزاد	104
103	رسول توکلی	102
104	صبا شیرازی	Null

تذکر: در جدول employee، #manager نسبت به جدول employee کلید خارجی است!

تمرینهای حل شده

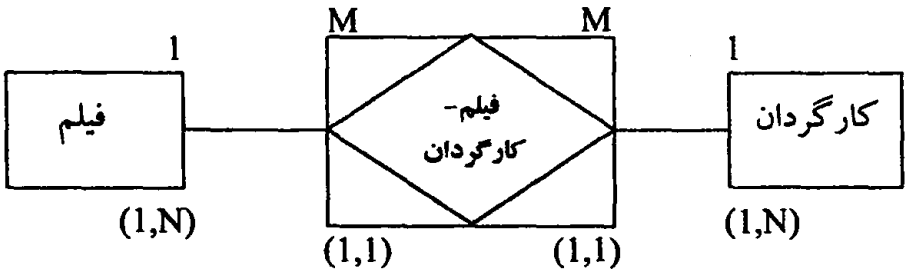
تمرین ۱: فرض کنید بخواهید یک سایت اطلاع‌رسانی در مورد سینما طراحی کنید. با در نظر گرفتن قوانین ذیل نمودار ER سیستم را رسم کرده، پایگاه داده‌های آن را طراحی کنید:

قانون ۱: کلیهٔ اشخاصی که مشخصات آنها در پایگاه داده‌های این سایت نگهداری می‌شود، کارگردان یا بازیگر یا تهیه کننده یا فیلمنامه نویس و یا ترکیبی از موارد فوق می‌باشند.

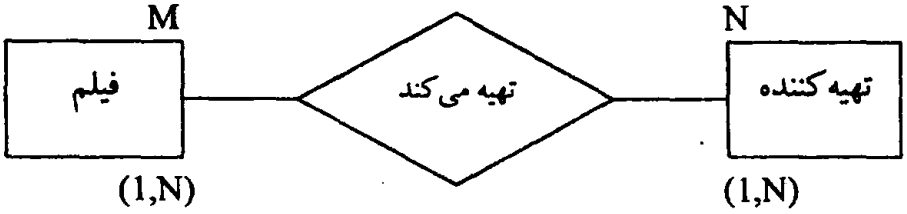


قانون ۲: هر فیلم یک یا چند کارگردان دارد و هر کارگردان ممکن است فیلمهای متعددی را کارگردانی کرده باشد.

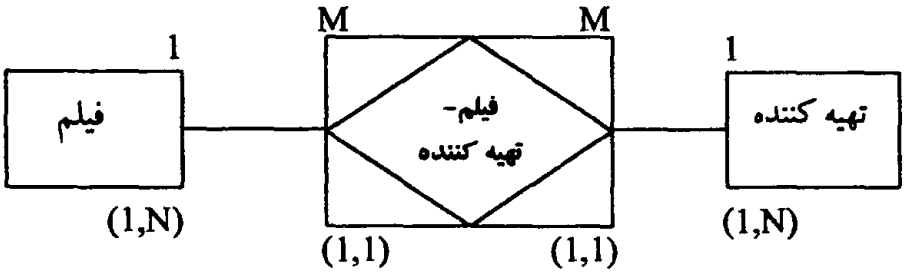
این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:



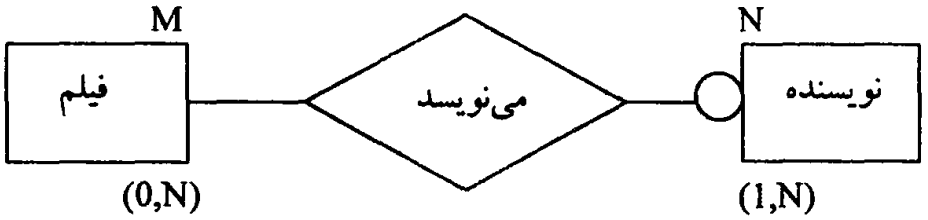
قانون ۳: هر فیلم یک یا چند تهیه کننده دارد و هر تهیه کننده ممکن است فیلمهای متعددی را تهیه کرده باشد.



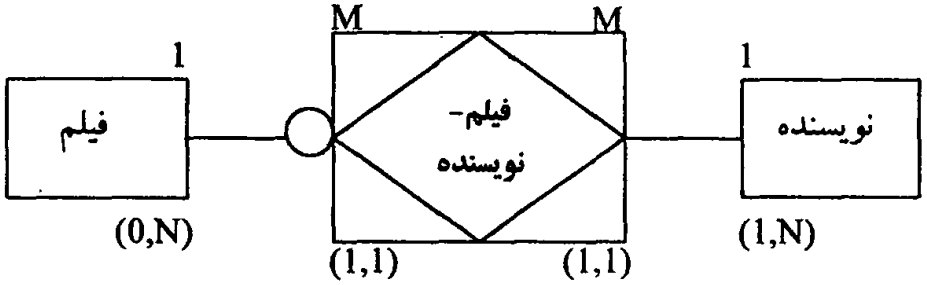
این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:



قانون ٤: هر فیلم ممکن است چند نویسنده داشته باشد و یا نویسنده‌ای نداشته باشد. هر نویسنده ممکن است فیلمنامه‌های متعددی را نوشته باشد.

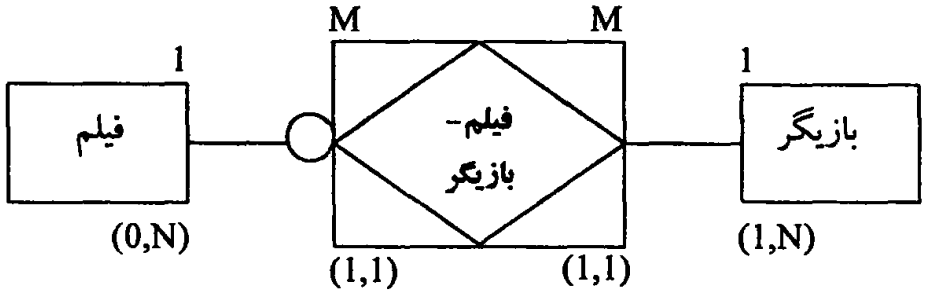


این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:

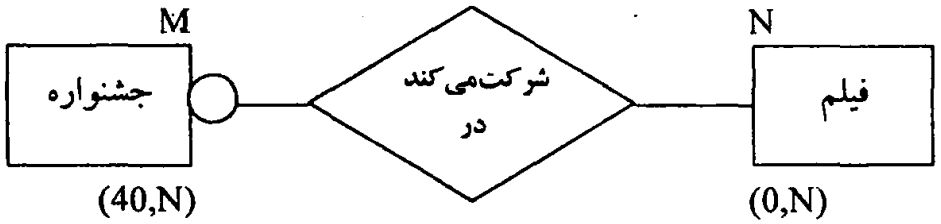


قانون ٥: هر فیلم ممکن است چند بازیگر داشته باشد و یا هیچ بازیگری نداشته باشد (مثل فیلمهای مستند). هر بازیگر ممکن است در فیلمهای متعددی بازی کرده باشد.

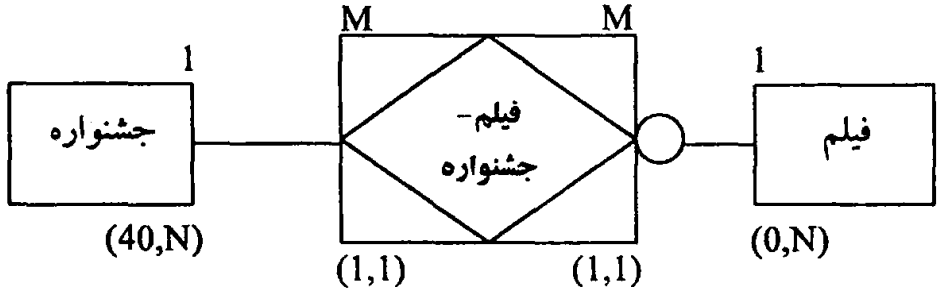
این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:



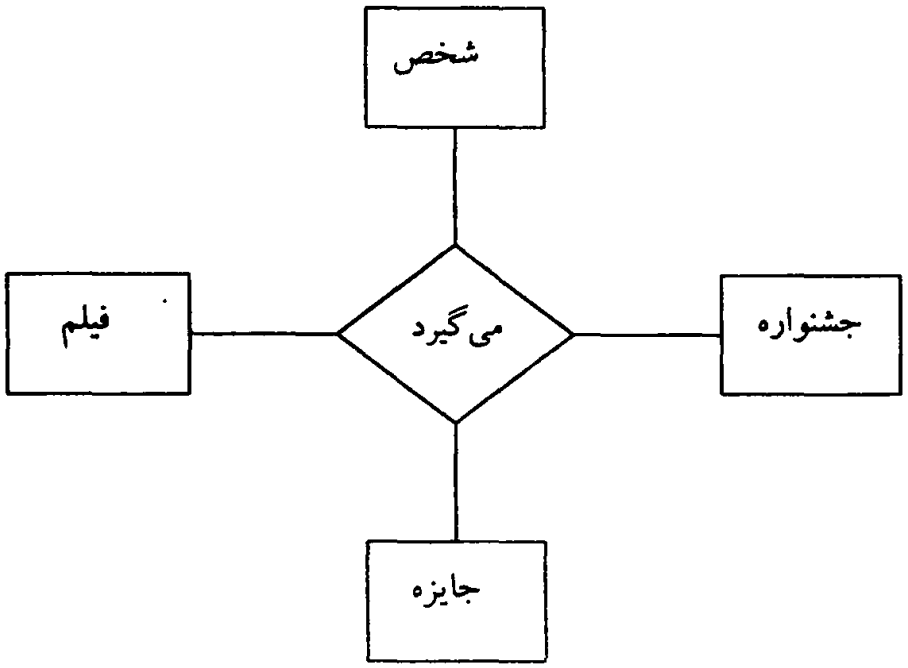
قانون ٦: هر فیلم ممکن است در جشنواره‌های مختلف شرکت کند و یا در هیچ جشنواره‌ای شرکت نکند. در هر جشنواره حداقل ۴۰ فیلم شرکت می‌کنند.



این رابطهٔ M:N را به دو رابطهٔ 1:M می‌شکنیم:

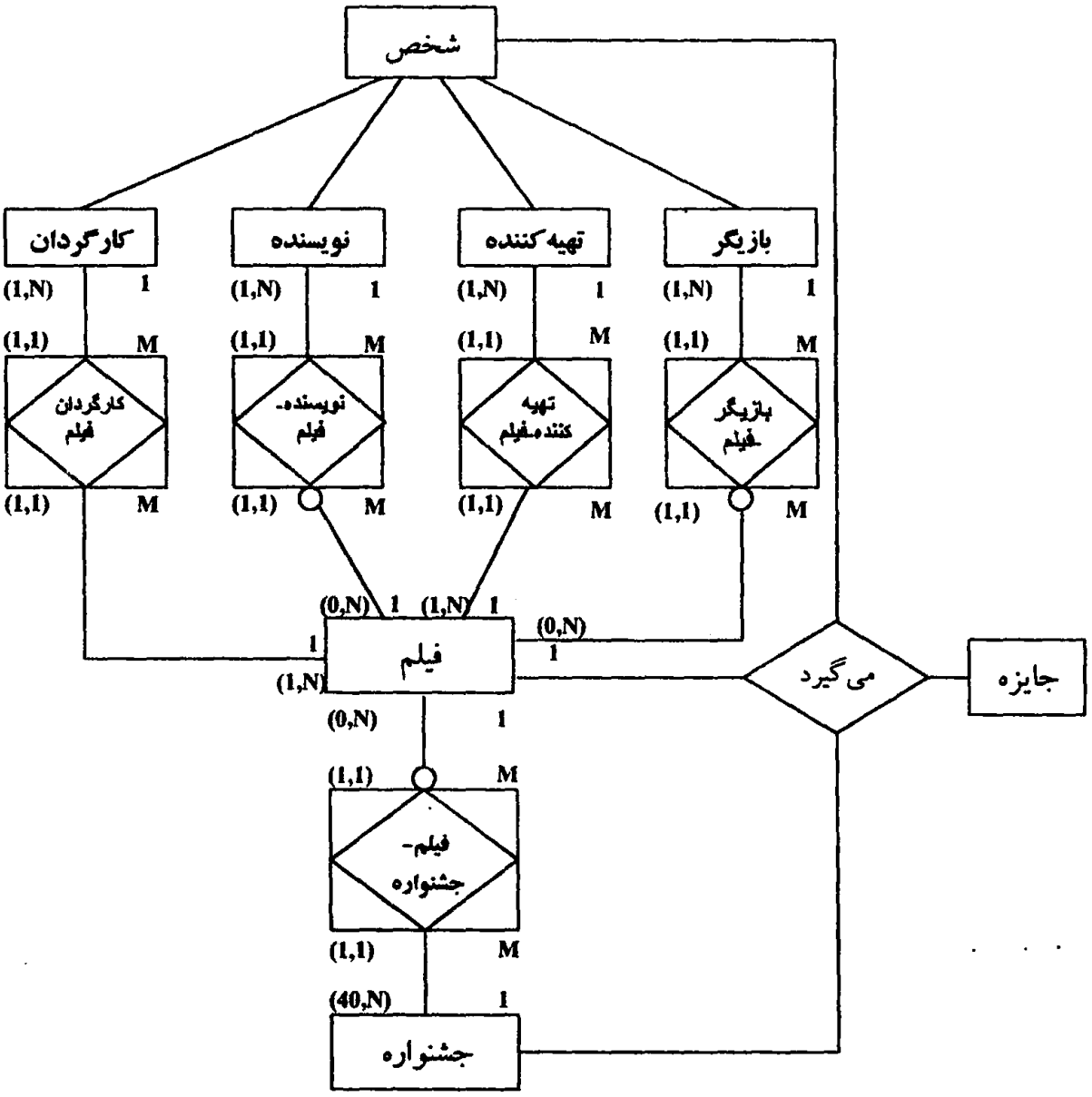


قانون ۷: در هر جشنواره ممکن است به هر یک از اشخاص برای نقشی که در یک فیلم داشته‌اند نوعی جایزه تعلق گیرد. مثلاً ممکن است در جشنوارهٔ سینمای ایران دورهٔ چهاردهم شخص X برای بازی در فیلم Y برنده یا کاندیدای جایزهٔ بهترین بازی در نقش دوم زن شود.



تذکر: این رابطه ۴ تایی است. درجه و کاردینالیتی روابط n-تایی با n>2 از قوانین پیچیده‌ای تبعیت می‌کنند که خارج از حد بحث این کتاب است. بهمین دلیل از ذکر جزئیات این رابطه صرفنظر شده است.

نمودار کلی:



ساختار جداول پایگاه داده‌ها:

موجودیت شخص تبدیل به جدول شخص می‌شود. به هر شخص اعم از کارگردان، فیلمنامه نویس، تهیه کننده یا بازیگر صرفنظر از سمت یا سمتهایی که دارد یک شمارهٔ منحصر بفرد می‌دهیم:

شخص (شمارهٔ شخص- نام شخص- جنسیت- زندگینامه)

موجودیت کارگردان تبدیل به جدول کارگردان می‌شود:

کارگردان (شمارهٔ شخص)

فرض کنید برای کارگردانان تنها ویژگیهایی که در جدول شخص در نظر گرفته‌ایم مورد نظر باشد. در اینصورت، چون جدول کارگردان تنها یک ویژگی دارد در پایان کار آن را حذف می‌کنیم.

موجودیت نویسنده تبدیل به جدول نویسنده می‌شود:

نویسنده (شمارهٔ شخص)

فرض کنید برای نویسندگان تنها ویژگیهایی که در جدول شخص در نظر گرفته‌ایم مورد نظر باشد. در اینصورت، چون جدول نویسنده تنها یک ویژگی دارد در پایان کار آن را حذف می‌کنیم.

موجودیت تهیه کننده تبدیل به جدول تهیه کننده می‌شود:

تهیه کننده (شمارهٔ شخص)

فرض کنید برای تهیه کنندگان تنها ویژگیهایی که در جدول شخص در نظر گرفته‌ایم مورد نظر باشد. در اینصورت، چون جدول تهیه کننده تنها یک ویژگی دارد در پایان کار آن را حذف می‌کنیم.

موجودیت بازیگر تبدیل به جدول بازیگر می‌شود:

بازیگر (شمارهٔ شخص)

فرض کنید برای بازیگران تنها ویژگیهایی که در جدول شخص در نظر گرفته‌ایم مورد نظر باشد. در اینصورت، چون جدول بازیگر تنها یک ویژگی دارد در پایان کار آن را حذف می‌کنیم.

موجودیت فیلم تبدیل به جدول فیلم می‌شود. به هر فیلم یک شمارهٔ منحصر بفرد می‌دهیم:

فیلم (شمارهٔ فیلم- نام فیلم- موضوع فیلم- سال ساخت)

موجودیت مرکب فیلم-کارگردان تبدیل به جدول فیلم-کارگردان می‌شود:

فیلم- کارگردان (شمارهٔ فیلم- شمارهٔ شخص)

موجودیت مرکب فیلم-نویسنده تبدیل به جدول فیلم-نویسنده می‌شود:

فیلم- نویسنده (شمارهٔ فیلم- شمارهٔ شخص)

موجودیت مرکب فیلم-تهیه کننده تبدیل به جدول فیلم-تهیه کننده می‌شود:

فیلم- تهیه کننده (شمارهٔ فیلم- شمارهٔ شخص)

موجودیت مرکب فیلم-بازیگر تبدیل به جدول فیلم-بازیگر می‌شود:

فیلم- بازیگر (شمارهٔ فیلم- شمارهٔ شخص- نقش)

موجودیت جشنواره تبدیل به جدول جشنواره می‌شود:

جشنواره (شمارهٔ جشنواره-نام جشنواره-محل برگزاری-تاریخ برگزاری)

موجودیت مرکب فیلم-جشنواره تبدیل به جدول فیلم-جشنواره می‌شود:

فیلم- جشنواره (شمارهٔ فیلم- شمارهٔ جشنواره)

موجودیت جایزه تبدیل به جدول جایزه می‌شود:

جایزه (شمارهٔ جایزه- نام جایزه)

رابطهٔ ۴-تایی "می‌گیرد" به جدولی تبدیل می‌شود که شامل کلیدهای اصلی هر ۴ موجودیت مربوطه است. توجه کنید ممکن است چند شخص که در یک فیلم نقش داشته‌اند جایزه‌های مختلف و یا مشابه بگیرند مثلاً ممکن است یک شخص برای بازی در یک فیلم جایزهٔ بهترین بازیگر نقش اول مرد را بگیرد و شخص دیگری برای کارگردانی همان فیلم جایزهٔ بهترین کارگردانی را بگیرد و یا یک فیلم دو کارگردان داشته باشد و هر دو جایزهٔ بهترین کارگردانی را بگیرند. همچنین، ممکن است در یک جشنواره برای یک فیلم جایزه‌های مختلفی به یک شخص تعلق گیرد مثلاً ممکن است به شخصی که کارگردانی و نویسندگی یک فیلم را بر عهده داشته است، هم جایزهٔ بهترین کارگردانی و هم جایزهٔ بهترین فیلمنامه تعلق گیرد. پس کلید اصلی این جدول باید شامل هر ۴ ویژگی باشد:

اعطاء جایزه(شمارهٔ جشنواره-شمارهٔ فیلم-شمارهٔ جایزه-شمارهٔ شخص-نوع هدیه- نتیجه)

برای درک بهتر مطلب به نمونهٔ این جداول توجه کنید:

film

film#	fname	year	subject
1	عروس آتش	78	اجتماعی
2	کما	83	اجتماعی-خانوادگی
3	قرمز	77	خانوادگی

people

id	name	sex	biography
100	فریدون جیرانی	m	
101	علی حاتمی	m	علی حاتمی‌در سال ۱۳۲۳ در تهران متولد شد. در اواخر دورهٔ متوسطه به ادارهٔ هنرهای دراماتیک رفت ...
102	خسرو سینایی	m	خسرو سینائی در سال ۱۳۱۹ متولد شد. او فارغ التحصیل آکادمی‌موسیقی و هنرهای نمایشی وین در رشتهٔ کارگردانی ...
103	حمید فرخ نژاد	m	
104	پیمان عمادی	m	
105	هدیه تهرانی	z	هدیه تهرانی متولد ۱۳۵۱ در تهران است.خانم تهرانی با بازی در فیلم سلطان آقای کیمیائی وارد عرصه بازیگری شد...
106	محمد رضا فروتن	m	محمدرضا فروتن، فارغ‌التحصیل روانشناسی بالینی ازدانشگاه آزاد، با آغاز موج جوانگرایی در سینمای ایران در نیمه دوم دهه ۱۳۷۹ ...
107	غزل صارمی	z	
108	آرش معیریان	m	آرش معیریان در سال ۱۳۵۰ در تهران متولد شد. او فوق دیپلم برنامه‌نویسی کامپیوتر، لیسانس کارگردانی سینما از دانشگاه...
109	سعید پورصمیمی	m	
110	عبدالله علیخانی	m	
111	حسن فرح بخش	m	
112	سلیمه رنگزن	z	

film_director

film#	id
1	102
2	108
3	100

film_writer

film#	id
1	102
1	103
2	104
3	100

film_actor

film#	id	role
1	107	اول
1	109	دوم
1	112	دوم
3	105	اول
3	106	اول

film_provider

film#	id
1	111
1	112

award

award#	Aname
k	بهترین کارگردانی
f	بهترین فیلمنامه
z1	بهترین بازی در نقش اول زن
z2	بهترین بازی در نقش دوم زن
m1	بهترین بازی در نقش اول مرد
m2	بهترین بازی در نقش دوم مرد

festival

fes#	fname	year	place
1	فجر-دوره هجدهم	78	ایران
2	جشن خانه سینما-دوره چهارم	79	ایران
3	جشن خانه سینما-دوره سوم	78	ایران

festival_film

fes#	film#
1	1
1	3
3	1
3	3

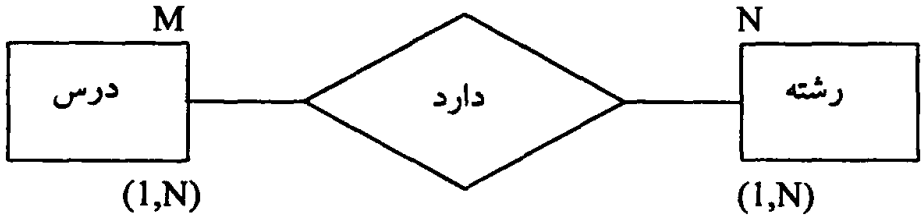
grant

fes#	film#	award#	id	kind	result
1	1	z2	112	دیپلم افتخار	برگزیده
1	1	f	102	سیمرغ بلورین	برگزیده
1	1	f	103	سیمرغ بلورین	برگزیده
2	1	z2	112	دیپلم افتخار	برگزیده
2	1	f	102	Null	کاندیدا
2	1	f	103	Null	کاندیدا
2	3	m1	106	Null	کاندیدا
3	3	m1	106	Null	کاندیدا

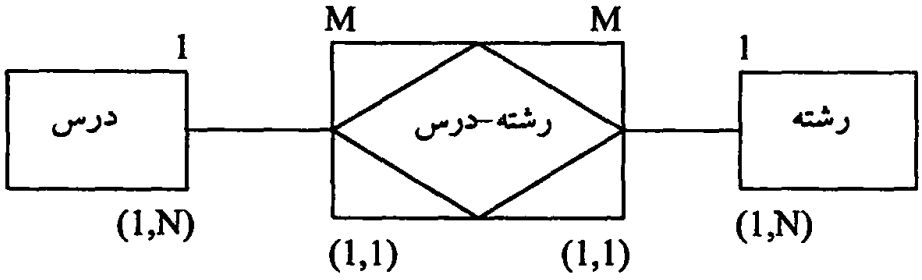
تمرین ۲: با این فرض که قوانین ذیل در یک دانشگاه حاکم باشد نمودار ER آن را رسم کرده، پایگاه داده‌های آن را طراحی کنید.

قانون ۱: هر دانشجو تنها در یک رشته تحصیل می‌کند. در هر رشته دانشجوهای متعددی تحصیل می‌کنند.

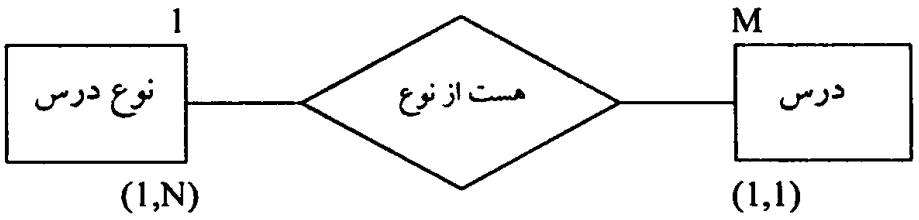
قانون ۲: هر رشته چندین درس دارد و هر درس ممکن است مربوط به چند رشتهٔ تحصیلی باشد.



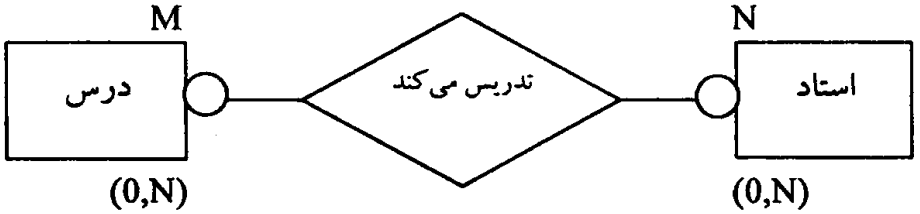
این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:



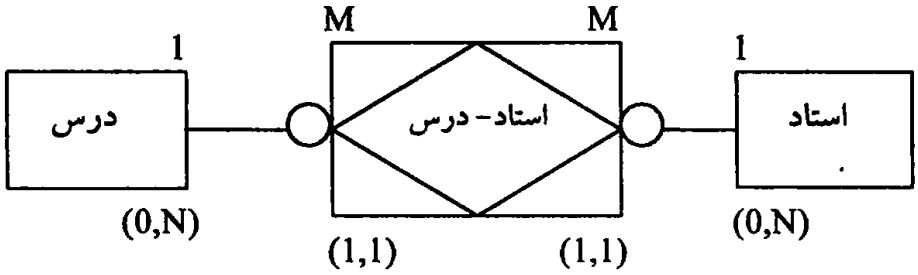
قانون ۳: برای هر نوع درس (نظری- عملی ...) ممکن است چندین درس وجود داشته باشد ولی هر درس تنها مربوط به یک نوع درس است.



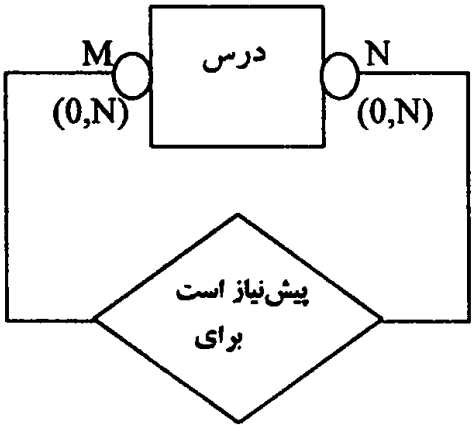
قانون ٤: در هر ترم ممکن است هر درس توسط چندین استاد ارائه شود و یا اصلاً ارائه نشود. هر استاد ممکن است در طول ترم چندین درس را تدریس کند و یا هیچ درسی را تدریس نکند.



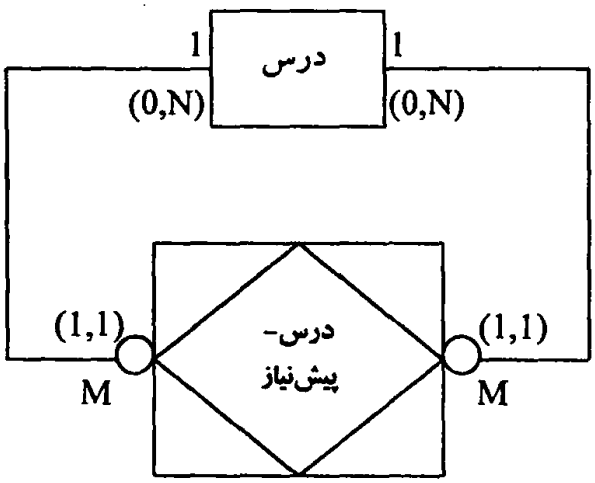
این رابطهٔ M:N را به دو رابطه 1:M می‌شکنیم:



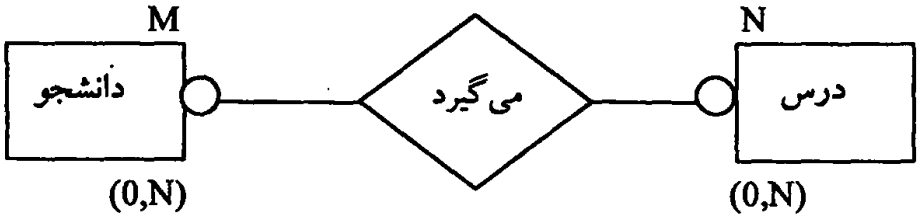
قانون ٥: هر درس ممکن است چند پیش‌نیاز داشته باشد و یا پیش‌نیازی نداشته باشد.



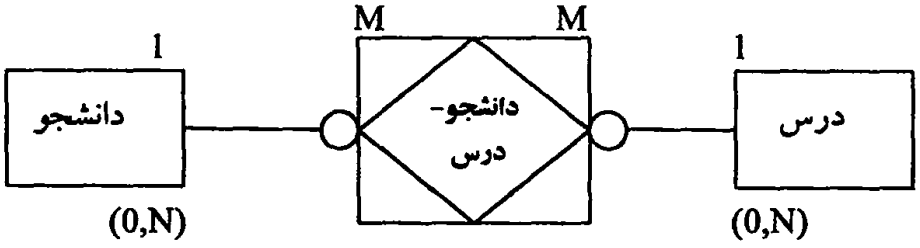
این رابطهٔ M:N را به دو رابطهٔ 1:M تبدیل می کنیم:



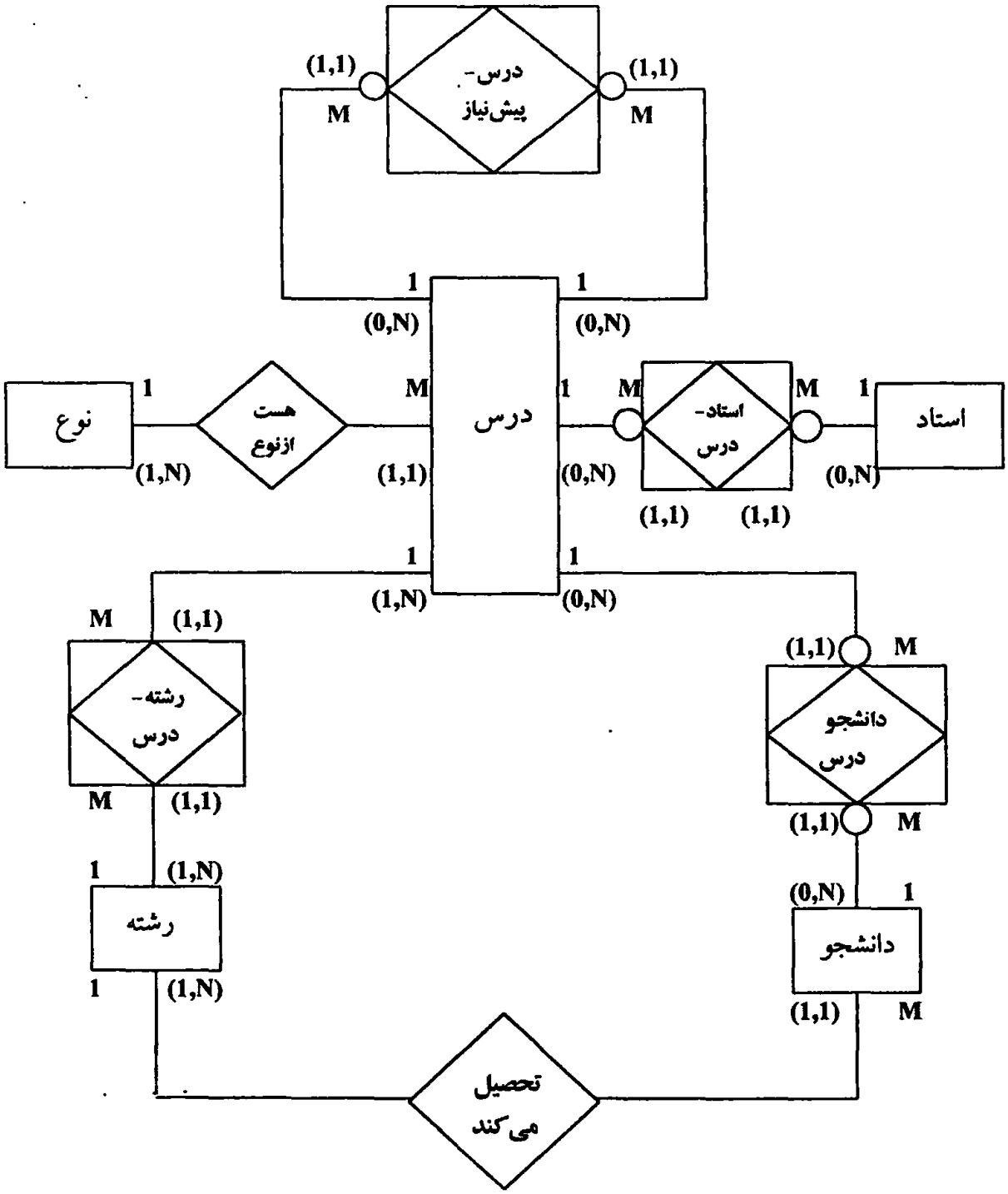
قانون ٦: هر دانشجو ممکن است درسهای متعددی را بگیرد. هر درس ممکن است توسط چندین دانشجو اخذ شود.



این رابطهٔ M:N را به دو رابطهٔ 1:M می‌شکنیم:



نمودار کلی به شکل ذیل خواهد بود:



ساختار جداول پایگاه داده‌ها:

دانشجو (شماره دانشجویی- نام دانشجو- سال ورود- کد رشته)

رشته (کد رشته- نام رشته)

نوع درس (کد نوع- نام نوع-قیمت واحد)

درس (شمارهٔ درس- نام درس- تعداد واحد- کد نوع درس)

رشته-درس (کد رشته- شمارهٔ درس)

استاد (شمارهٔ استاد- نام استاد- مدرک)

استاد-درس (شمارهٔ استاد- شمارهٔ درس- ترم)

دانشجو-درس (شماره دانشجویی- شمارهٔ درس- ترم- نمره)

درس-پیش‌نیاز (شمارهٔ درس- شمارهٔ درس پیش‌نیاز)

برای درک بهتر مطلب به نمونهٔ جداول این پایگاه داده‌ها توجه کنید:

field

field#	fieldName
1	مهندسی کامپیوتر
2	مهندسی الکترونیک
3	ریاضی محض

type

type#	typeName	fee
1	نظری	5000
2	عملی	20000
3	آزمایشگاه	30000

st

st#	sname	startYear	field#
8001	آرش راد	80	1
8002	عسل شاملو	80	3
8003	سیاوش آزاد	80	1
8101	ساغر راد	81	1
8102	علی نیکی	81	1
8111	فرامرز نیکی	81	1
8112	علی رضایی	81	2

course

crs#	cname	unit	type#
1100	ادبیات	2	1
1105	تربیت بدنی	1	2
1400	برنامه‌سازی ۱	3	1
1402	برنامه‌سازی ۲	3	1
1403	ذخیره‌وبازیابی	3	1
1407	پایگاه داده‌ها	3	1
1500	ریاضی ۱	3	1
1600	زبان پیش‌نیاز	2	1

course_field

crs#	field#
1100	1
1100	2
1100	3
1105	1
1105	2
1105	3
1400	1
1400	2
1400	3
1402	1
1403	1
1407	1
1500	1
1500	2
1500	3

st_course

st#	crs#	term	grade
8001	1400	811	9
8001	1400	812	13
8001	1402	811	13
8101	1400	821	19
8101	1500	821	14
8111	1400	811	12
8111	1402	811	20

pre

crs#	preCrs#
1400	1500
1402	1400
1407	1402
1407	1403

professor

prof#	pname	degree
101	فرانک شامی	L
102	علی پیامی	F
106	آزاده نیکو	F
107	سیامک فرد	D
108	علی نادرنژاد	F

prof_course

prof#	course#	term
101	1400	801
101	1400	802
101	1401	801
102	1400	801
108	1500	811

تمرینهای فصل

تذکر: به منظور راهنمایی دانشجویان، پاسخ نهایی برخی از تمرینهای این فصل در صورت سوالات فصل ۴ گنجانده شده است.

۱-فرض کنید بخواهید پایگاه داده‌های بیمارستانی با قوانین ذیل را طراحی کنید. نمودار ER سیستم را رسم کرده، ساختار جداول آن را استخراج کنید:

قانون ۱: هر بخش تعدادی خدمه دارد ولی هریک از خدمه تنها در یک بخش کار می‌کنند.

قانون ۲: هر بخش تعدادی پرستار دارد ولی هر پرستار تنها در یک بخش کار می‌کند.

قانون ۳: هر بخش تعدادی پزشک دارد ولی هر پزشک تنها در یک بخش کار می‌کند.

قانون ۴: هر پزشک حداقل در یک زمینه تخصص دارد و در هر زمینهٔ تخصص ممکن است پزشکان زیادی وجود داشته باشند.

قانون ۵: در هر بخش، تنها یکی از پرستارها مسئول بخش است.

قانون ۶: در هر شیفت از یک تاریخ تعدادی پرستار در بیمارستان حضور دارند.

قانون ۷: در هر شیفت از یک تاریخ تعدادی پزشک در بیمارستان حضور دارند.

قانون ۸: هر بخش تعدادی اتاق و هر اتاق تعدادی تخت دارد.

قانون ۹: هر بیمار ممکن است چند بار در بیمارستان بستری شود.

قانون ۱۰: هر بیمار در هر دورهٔ اقامت یک تخت در اختیار دارد.

قانون ۱۱: هر بیمار در هر دورهٔ اقامت تنها تحت مراقبت یک پزشک است ولی یک پزشک در هر لحظه ممکن است معالجهٔ چندین بیمار را بر عهده داشته باشد.

قانون ۱۲: هر بیمار تنها تحت پوشش یک نوع بیمه می‌باشد ولی مسلماً ممکن است بیماران زیادی تحت پوشش یک نوع بیمه باشند.

قانون ۱۳: پس از هر دورهٔ اقامت بیمار در بیمارستان، یک صورتحساب صادر می‌شود.(برای هر اقامت تنها یک صورتحساب صادر می‌شود و هر صورتحساب تنها مربوط به یک دورهٔ اقامت بیمار است.)

قانون ۱۴: صورتحساب، شامل کلیهٔ هزینه‌های مربوط به داروها و وسایل و تجهیزات مورد استفاده برای تشخیص بیماری و یا درمان بیمار در طول مدت زمان اقامت وی در بیمارستان می‌باشد.

۲-فرض کنید بخواهید سیستم پرسنلی و مدیریت پروژهٔ یک سازمان را طراحی کنید. با فرض حاکمیت قوانین ذیل در این سازمان نمودار ER این سیستم را رسم کرده، ساختار جداول پایگاه داده‌های آن را استخراج کنید.

قانون ۱: کارمندان این سازمان به دو دستهٔ استخدام رسمی و استخدام قراردادی تقسیم می‌شوند. به کارمندان استخدام رسمی حقوق ثابت تعلق می‌گیرد ولی حقوق کارمندان قراردادی بر اساس مبلغ ساعتی توافق شده در آخرین قرارداد منعقد شده با هر یک از آنها محاسبه می‌شود. (با هر کارمند قراردادی ممکن است یک یا چند قرارداد منعقد شود.)

قانون ۲: هر کارمند در یک یا چند زمینه تخصص دارد. در هر زمینهٔ کاری ممکن است چند کارمند وجود داشته باشند.

قانون ۳: دراین سازمان در هر لحظه یک یا چند پروژه در حال اجرا است. هر کارمند ممکن است روی چند پروژه کار کند و در هر پروژه چندین کارمند فعالیت دارند.

قانون ۴: در هر پروژه یکی از کارمندان مدیر پروژه است. هر کارمند می‌تواند حداکثر مدیر یک پروژه باشد.

قانون ۵: هر پروژه به تعدادی وظیفه[1] شکسته می‌شود. هر وظیفه حداقل توسط یک و حداکثر توسط سه کارمند انجام می‌شود.

قانون ۶: هر یک از کارمندان موظفند هر روز برای هر یک از وظایفی که روی آنها کار می‌کنند یک گزارش پیشرفت کار مجزا به سیستم تحویل دهند.

۳-با فرض داشتن قوانین ذیل در یک دانشگاه نمودار ER سیستم را رسم کرده، ساختار جداول آن را استخراج کنید:

قانون ۱: هر دانشکده چندین گروه آموزشی دارد.

قانون ۲: هر گروه تعدادی عضو هیات عامی دارد ولی هر یک از اساتید تنها عضو هیات علمی یک گروه هستند. در هر گروه تنها یکی از اعضاء هیات علمی مدیر گروه نیز هست.

قانون ۳: درطول هر ترم در هر گروه آموزشی تعدادی درس ارائه می‌شود و هر درس ممکن است در چند گروه آموزشی ولی در هر گروه آموزشی تنها توسط یک استاد ارائه شود.

قانون ۴: هر درس ممکن است چند پیش‌نیاز داشته باشد و یا پیش‌نیازی نداشته باشد.

قانون ۵: هر درس ممکن است چند هم نیاز داشته باشد و یا هم نیازی نداشته باشد .

۴-فرض کنید بخواهید یک سایت اطلاع‌رسانی اینترنتی در بارهٔ تاریخچهٔ بازیهای جام جهانی فوتبال طراحی کنید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: در هر جام تعدادی تیم شرکت می‌کنند. هر تیم ممکن است در جامهای مختلف شرکت کند.

قانون ۲: در هر جام هر تیم حداقل ۲۰ بازیکن دارد. در هر جام، هر بازیکن تنها در یک تیم بازی می‌کند.

---

1 - Task

قانون ۳: هر تیم در هر جام یک یا چند مربی دارد. در هر جام هر مربی تنها مربی‌گری یک تیم را برعهده دارد.

قانون ۴: در هرجام هر تیم ممکن است با چند تیم مختلف بازی کند.

قانون ۵: درهر مسابقه چند داور بر مسابقه نظارت دارند. هر داور ممکن است در چندین مسابقه داوری کند.

قانون ۶: در هر بازی ممکن است هر بازیکن چند گل را به ثمر برساند ولی هر گل تنها توسط یک بازیکن به ثمر می‌رسد.

قانون ۷: ممکن است یک بازیکن در یک بازی خطاهای زیادی را مرتکب شود. هر خطا توسط یک بازیکن صورت می‌گیرد.

قانون ۸: کلیهٔ اشخاصی که اطلاعات آنها در این سیستم ذخیره شده است، بازیکن، مربی، داور و یا ترکیبی از این موارد هستند.

۵- سیستم اخذ پروژهٔ فارغ التحصیلی و کارورزی توسط دانشجویان را در نظر بگیرید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر پروژهٔ فارغ التحصیلی ممکن است توسط یک یا دو دانشجو ارائه شود ولی هر دانشجو تنها یک پروژهٔ فارغ التحصیلی ارائه می‌دهد.

قانون ۲: برای هر پروژهٔ فارغ التحصیلی تنها یک استاد راهنما تعیین می‌شود. هر استاد حداکثر می‌تواند راهنمایی ۵ پروژهٔ فارغ التحصیلی را برعهده داشته باشد.

قانون ۳: هر پروژهٔ کارورزی تنها توسط یک دانشجو ارائه می‌شود و هر دانشجو تنها یک پروژهٔ کارورزی ارائه می‌دهد.

۶- فرض کنید بخواهید برای یک باشگاه که کلاسهای مختلف ورزشی برای رشته‌های گوناگون برگزار می‌کند، سیستمی طراحی کنید. با فرض داشتن قوانین ذیل، نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر کلاس در یکی از سالنها برگزار می‌شود. در هر سالن در یک زمان مشخص (از ساعت تا ساعت) ممکن است بیش از یک کلاس برگزار شود.

قانون ۲: هر کلاس یک مربی دارد. هر مربی می‌تواند مربی یک یا چند کلاس باشد.

قانون ۳: هر کلاس چند عضو دارد. هر عضو ممکن است در کلاسهای مختلف ثبت نام کند.

۷- فرض کنید بخواهید برای یک سازمان که برای کمک‌رسانی به آسیب دیدگان حوادث طبیعی تیمهای مختلف را آموزش می‌دهد، سیستمی طراحی کنید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر تیم حداقل ۵ عضو دارد. هر شخص تنها عضو یکی از تیمها است.

قانون ۲: هر تیم تنها در یک زمینه تخصص دارد ولی در هر زمینه ممکن است چندین تیم مختلف وجود داشته باشند.

قانون ۳: در هر تیم، یکی از اعضا سرگروه است.

قانون ۴: هر تیم ممکن است درامدادرسانی به آسیب دیدگان حوادث مختلف فعالیت داشته باشد. پس از هر حادثه ممکن است چندین تیم به کمک آسیب دیدگان بروند.

قانون ۵: برای کمک رسانی به آسیب دیدگان هر حادثه، هر تیم ممکن است با تیمهای دیگر همکاری داشته و یا با هیچ یک از تیمهای دیگر همکاری نداشته باشد.

۸- فرض کنید بخواهید یک وب سایت برای اطلاع‌رسانی وفروش CD موسیقی طراحی کنید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر خواننده یا گروه ممکن است آلبومهای متعددی داشته باشد ولی هر آلبوم تنها متعلق به یک خواننده یا گروه است.

قانون ۲: هر آلبوم تعدادی آهنگ دارد.

قانون ۳: هر آهنگ یک یا چند آهنگساز دارد و هر آهنگساز ممکن است آهنگهای زیادی بسازد.

قانون ۴: هر آهنگ یک یا چند نویسندهٔ متن دارد و هر نویسنده ممکن است متن آهنگهای مختلفی را بنویسد.

قانون ۵: : هر آهنگ یک یا چند تنظیم کننده دارد و هر تنظیم کننده ممکن است آهنگهای مختلفی را تنظیم کند.

قانون ۶: هر شخص ممکن است آهنگساز، خواننده، تنظیم کننده، نویسنده و یا ترکیبی از این موارد باشد.

قانون ۷: هر خواننده یا گروه از یک یا چند سبک موسیقی پیروی می‌کند. برای هر سبک موسیقی، خوانندگان و یا گروههایی وجود دارند.

۹- فرض کنید بخواهید یک سایت اینترنتی برای اطلاع‌رسانی و فروش بلیط کنسرتهای مختلف طراحی کنید بطوریکه اعضا سایت با وارد کردن شناسه و کلمهٔ عبور مخصوص خود وارد سایت شده، بلیط کنسرتهای مختلف را خریداري کنند. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: در هر کنسرت تعدادی خواننده یا گروه شرکت دارند و هر خواننده یا گروه ممکن است در کنسرتهای مختلف شرکت کند.

قانون ۲: در هر کنسرت تعدادی نوازنده حضور دارند و هر نوازنده ممکن است در کنسرتهای مختلف شرکت کند.

قانون ۳: هر آهنگ یک یا چند آهنگساز دارد و هر آهنگساز ممکن است آهنگهای زیادی بسازد.

قانون ۴: هر آهنگ یک یا چند نویسندهٔ متن دارد و هر نویسنده ممکن است متن آهنگهای مختلفی را بنویسد.

قانون ۵: : هر آهنگ یک یا چند تنظیم کننده دارد و هر تنظیم کننده ممکن است آهنگهای مختلفی را تنظیم کند.

قانون ۶: هر شخص ممکن است آهنگساز، خواننده، تنظیم کننده، نویسنده و یا ترکیبی از این موارد باشد.

قانون ۷: هر خواننده یا گروه از یک یا چند سبک موسیقی پیروی می‌کند. برای هر سبک موسیقی، خوانندگان و یا گروههایی وجود دارند.

۹- فرض کنید بخواهید یک سایت اینترنتی برای اطلاع‌رسانی و فروش بلیط کنسرتهای مختلف طراحی کنید بطوریکه اعضا سایت با وارد کردن شناسه و کلمهٔ عبور مخصوص خود وارد سایت شده، بلیط کنسرتهای مختلف را خریداری کنند. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: در هر کنسرت تعدادی خواننده یا گروه شرکت دارند و هر خواننده یا گروه ممکن است در کنسرتهای مختلف شرکت کند.

قانون ۲: در هر کنسرت تعدادی نوازنده حضور دارند و هر نوازنده ممکن است در کنسرتهای مختلف شرکت کند.

قانون ۳: در هر کنسرت، هر نوازنده ممکن است یک یا چند آلت موسیقی را بنوازد و برای هر آلت موسیقی ممکن است چند نوازنده وجود داشته باشند.

قانون ۴: هر کنسرت در یک محل برگزار می‌شود. در هر محل ممکن است در زمانهای مختلف کنسرتهای مختلف اجرا شود.

قانون ۵: در هر خرید کاربر می‌تواند بلیط کنسرتهای مختلف را (به تعداد دلخواه) خریداری کند.

۱۰- فرض کنید بخواهید یک سایت اینترنتی برای یک فروشگاه اسباب‌بازی طراحی کنید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: برای هر گروه سنی ممکن است اسباب‌بازیهای زیادی وجود داشته باشند. همچنین، ممکن است برخی اسباب‌بازیها برای چند گروه سنی مناسب باشند.

قانون ۲: برای هر نوع اسباب‌بازی (ورزشی، فکری، تفریحی...) ممکن است چندین اسباب‌بازی وجود داشته باشد. همچنین، ممکن است برخی از اسباب‌بازیها در چند نوع مختلف طبقه‌بندی شوند.

قانون ۳: هر اسباب‌بازی توسط یک شرکت تولیدی ساخته می‌شود. هر شرکت ممکن است اسباب‌بازیهای زیادی بسازد.

قانون ۴: در هر خرید ممکن است یک یا چند اسباب‌بازی خریداری شود.(فرض کنید مشتریان سایت ناشناس هستند. به عبارت دیگر لازم نیست برای مشتری موجودیتی در نظر بگیرید.)

۱۱-فرض کنید بخواهید یک سایت اینترنتی گردهمایی[1] برای برنامه‌نویسان طراحی کنید بطوریکه هر یک از اعضاء سایت بتوانند مشکلات و سوالات برنامه‌نویسی خود را با سایر اعضاء سایت در میان گذاشته، پاسخهای آنها را دریافت کنند. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر عضو می‌تواند سوالات زیادی را مطرح کند و یا هیچ سوالی مطرح نکند.

قانون ۲: هر عضو می‌تواند به هر سوال مطرح شده چند پاسخ بدهد و یا هیچ پاسخی ندهد.

۱۲-فرض کنید بخواهید یک سایت اینترنتی برای برگزاری آزمونهای مختلف طراحی کنید بطوریکه اعضاء سایت با وارد کردن نام کاربری و کلمهٔ عبور خود وارد سایت شده، آزمون مورد نظر خود را انتخاب کرده، به سوالات چند گزینه‌ای آن پاسخ دهند و امتیاز بگیرند. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: در هر آزمون تعدادی سوال مطرح می‌شود.

قانون ۲: برای هر سوال ممکن است چندین گزینه وجود داشته باشد ولی تنها یکی از گزینه‌ها صحیح است.

قانون ۳: هر کاربر می‌تواند در چند آزمون و در هر آزمون حداکثر یک بار شرکت کند و یا در هیچ آزمونی شرکت نکند. در هر آزمون کاربران زیادی می‌توانند شرکت کنند.

قانون ۴: برای هر سوال کاربر مجاز است تنها یکی از گزینه‌ها را انتخاب کند.

---

1 -Forum

۱۳- فرض کنید بخواهید برای یک مرکز پژوهشی با تعدادی عضو سیستمی طراحی کنید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر پژوهشگر ممکن است مقالات زیادی ارائه کند. هر مقاله حداکثر توسط سه پژوهشگر ارائه می‌شود.

قانون ۲: هر مقاله در مورد یک موضوع است ولی برای هر موضوع ممکن است مقالات زیادی ارائه شود.

قانون ۳: هر یک از اعضا می‌توانند حداکثر یکبار نظر خود را در مورد یک مقاله اعلام کنند.

۱۴- یک بنگاه مسکن را در نظر بگیرید. وظیفهٔ این بنگاه معرفی صاحبان املاک به افرادی است که متقاضی اجاره یا خرید یا رهن ملک هستند. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: در هر منطقه تعدادی ملک آمادهٔ فروش یا اجاره وجود دارد.

قانون ۲: برخی از افراد ممکن است چندین ملک داشته باشند (مانند افراد بساز بفروش!) و هر ملک ممکن است چندین مالک داشته باشد.

قانون ۳: برای فروش، اجاره یا رهن هر ملک یک قرارداد تنظیم می‌شود.(ممکن است یک ملک چندین بار فروخته شده و یا رهن یا اجاره داده شود)

۱۵- یک تولیدی لباس را در نظر بگیرید. با فرض داشتن قوانین ذیل نمودار ER سیستم را رسم کرده، ساختار جداول پایگاه داده‌ها را استخراج کنید:

قانون ۱: هر خیاط در دوخت یک یا چند نوع لباس مهارت دارد. برای دوخت هر نوع لباس حداقل ۵ خیاط وجود دارند.

قانون ۲: برای دوخت هر نوع لباس از یک یا چند نوع پارچهٔ مخصوص استفاده می‌شود. از یک نوع پارچه ممکن است در دوخت یک یا چند نوع لباس استفاده شود.

قانون ۳: هر نوع پارچه تنها از یک تولید کنندهٔ پارچه خریداری می‌شود ولی ممکن است از یک تولید کننده چندین نوع پارچه خریداری شود.

قانون ۴: در هر خرید از یک تولید کنندهٔ پارچه ممکن است انواع مختلف پارچه‌ها خریداری شود.

قانون ۵: لباسهای تولیدی به فروشگاههای مختلف فروخته می‌شوند. در هر فروش ممکن است انواع مختلف لباسها به یک فروشگاه فروخته شوند.

۱۶-یک فروشگاه زنجیره‌ای را در نظر بگیرید که شامل فروشگاههای مختلف در یک شهر است. اطلاعات کلیهٔ فروشگاهها به صورت مشترک در یک پایگاه داده‌ها ذخیره می‌شود.

موجودیتهای اصلی این سیستم بشرح ذیل می‌باشند:

کارمندان:

هر یک از فروشگاهها دارای یک مدیر و تعدادی کارمند و کارگر می‌باشند. هر یک از کارمندانی که به سیستم نرم افزاری فروش دسترسی دارند، دارای یک شناسهٔ منحصر بفرد و یک کلمهٔ عبور می‌باشند.

مشتریان:

مشتریان این فروشگاهها ناشناس هستند و هیچ اطلاعاتی در مورد آنها در سیستم ثبت نمی‌شود.

کالاها و تامین کنندگان:

این فروشگاه زنجیره‌ای یک انبار مرکزی دارد. کلیهٔ کالاهای مورد نیاز از فروشندگان مربوطه خریداری شده، در این انبار ذخیره شده و سپس به فروشگاههای مختلف ارسال می‌شوند.

کالاها بگونه‌ای کدبندی شده‌اند که هر کالا با یک مارک بخصوص دارای یک کد منحصر بفرد باشد. این کد در کلیهٔ فروشگاهها یکسان است مثلاً کد مایع ظرفشویی جام در کلیهٔ فروشگاهها ۵۰۵ است. همچنین، هر کالا با یک مارک بخصوص تنها از یک فروشنده خریداری می‌شود مثلاً مایع ظرفشویی جام تنها از آقای آرش راد خریداری می‌شود.

برای حفظ سادگی مسئله از نحوهٔ محاسبهٔ قیمت کالاها صرفنظر کرده و فرض می‌کنیم قیمت فروش کالاها در جدول مربوطه مشخص است. تنها به این نکته توجه کنید که قیمت کالاها بر حسب واحدهای مختلف متفاوت است مثلاً قیمت یک خودکار بیک ۱۰۰ تومان است در حالیکه قیمت یک جعبه خودکار بیک که شامل ۲۰ خودکار بیک است، ۱۸۰۰ تومان و قیمت یک کارتن خودکار بیک که شامل ۱۰ جعبه خودکار بیک است، ۱۵۰۰۰ تومان می‌باشد.

برای هر کالا، یک مرز حداقل موجودی در نظر گرفته شده است. هنگامیکه موجودی کالای مورد نظر کمتر از این مرز باشد، کالای مورد نظر سفارش داده شده و خریداری می‌شود مثلاً اگر حداقل موجودی برای مایع ظرفشویی جام ۱۰۰۰ عدد باشد، هنگامیکه موجودی آن در انبار مرکزی به کمتر از ۱۰۰۰ عدد برسد، بایستی نسبت به خرید مایع ظرفشویی جام اقدام شود.

فاکتورها:

در هر فروشگاه تعدادی اپراتور کار می‌کنند. هر مشتری پس از انتخاب کالاهای مورد نظر خود به یکی از اپراتورها(صندوقها) مراجعه می‌کند. اپراتور، کد، واحد و تعداد کالاهای خریداری شده را وارد سیستم کرده، سیستم بطور خودکار یک فاکتور شامل ریز قیمت کالاها و کل مبلغ فاکتور صادر می‌کند.

مشتری می‌تواند وجه فاکتور را بصورت نقدی، چک یا کارت اعتباری بپردازد.(برای حفظ سادگی مسئله، از مسائل حسابداری نیز صرفنظر می‌کنیم.)

در هر فاکتور، تاریخ و ساعت صدور فاکتور و همچنین، شناسهٔ اپراتوری که فاکتور را صادر کرده است ثبت می‌شود، به این ترتیب، هر گونه اشکال احتمالی در فاکتورها قابل پی‌گیری خواهد بود.

در هر فروشگاه، هر فاکتور دارای یک شمارهٔ منحصر بفرد است. شمارهٔ فاکتورها در هر فروشگاه مستقل از فروشگاههای دیگر است و از ۱ شروع شده، یکی یکی به آن اضافه می‌شود.

برخی از گزارشاتی که قرار است از این سیستم گرفته شوند، به شرح ذیل می‌باشند:

-نام کالاهایی که باید سفارش داده شوند.

-نام و مقدار کالاهایی که در فاصلهٔ زمانی مشخص از یک فروشنده خریداری شده‌اند مثلاً نام و مقدار کالاهایی که از تاریخ ۸۲/۴/۴ تا تاریخ ۸۲/۵/۵ از آرش راد خریداری شده‌اند.

-قیمت کل یک فاکتور بخصوص مثلاً قیمت کل فاکتور شمارهٔ ۱۰۰۱ در فروشگاه ۲

-کل مبلغ فروش یک کالای بخصوص در یک فروشگاه بین دو تاریخ مشخص مثلاً کل مبلغ فروش پودر برف در فروشگاه شمارهٔ ۱ از تاریخ ۸۲/۴/۴ تا ۸۲/۵/۵

- نام اپراتوری که یک فاکتور بخصوص از یک فروشگاه را صادر کرده است مثلاً نام اپراتوری که در فروشگاه شمارهٔ ۱ فاکتور شمارهٔ ۱۰۰۱ را صادر کرده است.

...

با توجه به آنچه تا کنون آموخته‌اید پایگاه داده‌های این سیستم را طراحی کنید. چنانچه لازم می‌دانید فرضهایی را خود به صورت مسئله اضافه کنید.