دولت جمهوری اسلامی افغانستان
ادارۀ تعلیمات تخنیکی و مسلکی
معاونیت امور اکادمیک
ریاست نصاب و تربیه معلم

اساسات دیتابیس

رشته: کمپیوتر ساینس - دیپارتمنت: دیتابیس

صنف ۱۳ - سمستر دوم

سال: ۱۳۹۹ هجری شمسی

شناسنامه کتاب

نام کتاب: اساسات دیتابیس

رشته: کمپیوتر ساینس

تدوین کننده: محمد یاسر حکیمی

همکار تدوین کننده: دیوه خیرخواه بصیری

کمیته نظارت:

- ندیمه سحر رئیس ادارۀ تعلیمات تخنیکی و مسلکی
- عبدالحمید اکبر معاون امور اکادمیک ادارۀ تعلیمات تخنیکی و مسلکی
- حبیب الله فلاح رئیس نصاب و تربیه معلم
- عبدالمتین شریفی آمر انکشاف نصاب تعلیمی، ریاست نصاب و تربیه معلم
- روح الله هوتک آمر طبع و نشر کتب درسی، ریاست نصاب و تربیه معلم
- احمد بشیر هیله‌من مسؤل انکشاف نصاب، پروژه انکشاف مهارت‌های افغانستان
- محمد زمان پویا کارشناس انکشاف نصاب، پروژه انکشاف مهارت‌های افغانستان
- علی خیبر یعقوبی سرپرست مدیریت عمومی تألیف کتب درسی، ریاست نصاب و تربیه معلم

کمیته تصحیح:

- محمد علی شهاب
- مهدی بهار
- محمد امان هوشمند مدیرعمومی بورد تصحیح کتب درسی و آثار علمی

دیزاین: صمد صبا و سیدکاظم کاظمی

سال چاپ: ۱۳۹۹ هجری شمسی

تیراژ: ۱۰۰۰

چاپ: اول

ویب سایت: www.tveta.gov.af

ایمیل: info@tveta.gov.af

سرود ملی

دا وطـــن افغانســـتان دی — دا عـــزت د هـــر افغـــان دی
کـــور د ســـولې کـــور د تـــورې — هـــر بچـــی یـــې قهرمـــان دی
دا وطـــن د ټولـــو کـــور دی — د بلوڅـــــو، د ازبکـــــو
د پښـــتون او هـــزاره وو — د ترکمنـــــو، د تاجکـــــو
ورسره عـــرب، ګوجـــر دي — پامیریـــان، نورســـتانیان
براهـــوي دي، قزلبـــاش دي — هـــم ایمـــاق، هـــم پشـــه‌یان
دا هیـــواد بـــه تـــل ځلیـــږي — لکـــه لمـــر پـــر شـــنه آســـمان
پـــه ســـینه کـــې د آســـیا بـــه — لکـــــه زړه وی جاویـــــدان
نـــوم د حـــق مـــو دی رهبـــر — وایـــو الله‌اکبر وایـــو الله‌اکبر

پیام ادارهٔ تعلیمات تخنیکی و مسلکی

استادان نهایت گرامی و محصلان ارجمند!

تربیت نیروی بشری ماهر، متخصص و کارآمد از عوامل کلیدی و انکارناپذیر در توسعهٔ اقتصادی و اجتماعی هر کشور محسوب می‌گردد و هر نوع سرمایه‌گذاری بزرگ در بخش‌های مختلف اقتصادی نیازمند به پلان‌گذاری و سرمایه‌گذاری در بخش نیروی بشری و توسعهٔ منابع این نیرو می‌باشد. بر مبنای این اصل و بر اساس فرمان شماره ۱۱ مقام عالی ریاست جمهوری اسلامی افغانستان به تاریخ ۱۳۹۷/۲/۱ ادارهٔ تعلیمات تخنیکی و مسلکی از بدنهٔ وزارت معارف مجزا و فصل جدیدی در بخش عرضه خدمات آموزشی در کشور گشوده شد.

اداره تعلیمات تخنیکی و مسلکی به‌عنوان متولی و مجری آموزش‌های تخنیکی و مسلکی در کشور محسوب می‌شود که در چارچوب استراتژی ۵ ساله خویش دارای چهار اولویت مهم که عبارت‌اند از افزایش دسترسی عادلانه و مساویانه فراگیران آموزش‌های تخنیکی و مسلکی در سطح کشور، بهبود کیفیت در ارائه خدمات آموزشی، یادگیری مادام‌العمر و پیوسته و ارائه آموزش نظری و عملی مهارت‌ها به‌طور شفاف، کم‌هزینه و مؤثر که بتواند نیاز بازار کار و محصلان را در سطح محلی، ملی و بین‌المللی برآورده کند، می‌باشد.

این اداره که فراگیرترین نظام تعلیمی کشور در بخش تعلیمات تخنیکی و مسلکی است، تلاش می‌کند تا در حیطهٔ وظایف و صلاحیت خود زمینهٔ دستیابی به هدف‌های تعیین‌شده را ممکن سازد و جهت رفع نیاز بازار کار، فعالیت‌های خویش را توسعه دهد.

نظام اجتماعی و طرز زندگی در افغانستان مطابق به احکام دین مقدس اسلام و رعایت تمامی قوانین مشروع و معقول انسانی عیار است. ادارهٔ تعلیمات تخنیکی و مسلکی جمهوری اسلامی افغانستان نیز با ایجاد زمینه‌های لازم برای تعلیم و تربیت جوانان و نوجوانان مستعد و علاقه‌مند به حرفه‌آموزی، ارتقای مهارت‌های شغلی در سطوح مختلف مهارتی، تربیت کادرهای مسلکی و حرفوی و ظرفیت‌سازی تخصصی از طریق انکشاف و ایجاد مکاتب و انستیتوت‌های تخنیکی و مسلکی در سطح کشور با رویکرد ارزش‌های اسلامی و اخلاقی فعالیت می‌نماید.

فلهذا جهت نیل به اهداف عالی این اداره که همانا تربیهٔ افراد ماهر و توسعهٔ نیروی بشری در کشور می‌باشد؛ داشتن نصاب تعلیمی بر وفق نیاز بازار کار امر حتمی و ضروری بوده و کتاب درسی یکی از ارکان مهم فرایند آموزش‌های تخنیکی و مسلکی محسوب می‌شود، پس باید همگام با تحولات و پیشرفت‌های علمی نوین و مطابق نیازمندی‌های جامعه و بازار کار تألیف و تدوین گردد و دارای چنان ظرافتی باشد که بتواند آموزه‌های دینی و اخلاقی را توأم با دست‌آوردهای علوم جدید با روش‌های نوین به محصلان انتقال دهد. کتابی را که اکنون در اختیاردارید، بر اساس همین ویژگی‌ها تهیه و تدوین گردیده است.

بدین‌وسیله، صمیمانه آرزومندیم که آموزگاران خوب، متعهد و دلسوز کشور با خلوص نیت، رسالت اسلامی و ملی خویش را ادا نموده و نوجوانان و جوانان کشور را به‌سوی قله‌های رفیع دانش و مهارت‌های مسلکی رهنمایی نمایند و از محصلان گرامی نیز می‌خواهیم که از این کتاب به‌درستی استفاده نموده، در حفظ و نگهداشت آن سعی بلیغ به خرج دهند. همچنان از مؤلفان، استادان، محصلان و اولیای محترم محصلان تقاضا می‌شود نظریات و پیشنهادات خود را در مورد این کتاب از نظر محتوا، ویرایش، چاپ، اشتباهات املایی، انشایی و تایپی عنوانی ادارهٔ تعلیمات تخنیکی و مسلکی کتباً ارسال نموده، امتنان بخشند.

در پایان لازم می‌دانیم در جنب امتنان از مؤلفان، تدوین‌کنندگان، مترجمان، مصححان و تدقیق کنندگان نصاب تعلیمات تخنیکی و مسلکی از تمامی نهادهای ملی و بین‌المللی که در تهیه، تدوین، طبع و توزیع کتب درسی زحمت‌کشیده و همکاری نموده‌اند، قدردانی و تشکر نمایم.

ندیمه سحر

رئیس ادارهٔ تعلیمات تخنیکی و مسلکی جمهوری اسلامی افغانستان

فهرست

عنوان | صفحه

مقدمه

کمپیوتر اساساً یک نوع ماشین الکترونیکی است که معلومات را در خود ذخیره می‌کند و با استفاده از این معلومات ذخیره‌شـــده، کارهای بی‌نهایت عمده را در اندک زمان بر طبق هدایاتی که به او داده می‌شـــود، به انجام می‌رساند. باید دانست که این دستگاه عام‌المنفعهٔ تخنیکی یا به اصطلاح کمپیوتر، یک ماشین ساده نیست، بلکه نتیجهٔ سال‌ها و قرن‌ها پژوهش‌های علمی می‌باشد که دانش‌مندان تحقیقات و پژوهش‌های خود را یکی بعد از دیگری به دسترس دانش‌مندان و پژوهش‌گران نسل‌های مابعد خویش قرار داده‌اند تا بالاخره این اعجوبهٔ عالم را به میان آورده اند.

دانشـــی که همهٔ این مطالعات و تحقیقات علمی را انجام داده و هنوز هم مطالعات خویش را در راه پیشرفت و انکشاف بیش‌تر این اعجوبهٔ معاصر ادامه می‌دهد و حتا عصری به نام عصر کمپیوتر را برای خویش اختصاص داده است، به نام دانش کمپیوتر یا Computer Science یاد می‌شود.

این علم دلچسپ و عام‌المنفعه که هر نوع موضوعات تخنیکی و اوپراتیفی پیشرفت‌ها و انکشافات معاصر را به سرعت سرسام‌آوری تحلیل، مطالعه و تدقیق می‌نماید، از سه خصوصیت ممتاز برخوردار است:

۱. این دانش به سرعت انکشاف نموده و انکشاف آن به سرعت ادامه دارد.
۲. صورت استفاده از این دانش ساحهٔ بی‌نهایت وسیع را در تمام امور زندگی معاصر احتواء می‌نماید.
۳. کسب و آموزش این دانش برای هرنوع استعداد فکری و دارندگان عقل سلیم میسر است.

ولی باید متوجه بود که آموزش این دانش کار پیگیر و مطالعهٔ دوامدار را توصـیه می‌نماید. به طور خاص، استفاده و دانستن تکنالوژی دیتابیس، امروز یک جزء مهم اکثر امور زندگی را تشکیل می‌دهد. دیتابیس‌ها در بسیاری موارد استفاده می‌شوند؛ به طور مثال، در قسمت تجارت از طریق انترنت(E-Commerce)، پیدانمودن اشـــیای مورد نیاز، پیدانمودن کتاب‌ها در کتابخانه‌ها به صـــورت Online و یا Offline، پیدانمودن وظیفه، اشخاص و دیگر ضروریات از طریق شبکه‌های محلی، ملی و بین‌المللی، دیتابیس‌ها به طور وسیع استفاده می‌شوند.

دیتابیس‌ها توسط هزاران شرکت بزرگ و کوچک و ملیون‌ها فرد به سطح بین‌المللی استفاده می شوند. یک سروی، تعداد دیتابیس‌های فعال در سال 2002 به سطح تمام جهان را اضافه از 10 ملیون نشان می‌دهد.

در این نوت، علم دیتابیس به صورت عمومی توضیح و جهت تدریس آماده شده است و هم‌چنان کوشش زیاد به عمل آمده اســـت تا مفهوم دیتابیس، تکنالوژی دیتابیس و تخنیک‌هایی که توســـط آن‌ها دیزاین و استفادهٔ دیتابیس صورت می‌گیرد، توضیح شود.

هدف کلی کتاب

آشنایی با دیتابیس، مودل ها، جدول ها، ارتباطات، نارمل سازی و ER Diagram.

فصل اول

معرفی دیتابیس (Database Concept)

هدف کلی: آشنایی محصلان با دیتابیس

اهداف آموزشی: در پایان این فصل، محصلان قادر خواهند بود تا:

۱. دیتابیس را تعریف کنند.
۲. موارد استفادهٔ دیتابیس را بیان دارند.
۳. دیتابیس سیستم را تشریح کنند.
۴. سیستم مدیریت دیتابیس را تشریح کنند.
۵. سیستم مدیریت فایل و سیستم مدیریت دیتابیس را از هم تفکیک کرده بتوانند.

در این فصل معرفی دیتابیس، سیستم دیتابیس، سیستم مدیریت دیتابیس و مقایسهٔ سیستم مدیریت فایل و سیستم مدیریت دیتابیس را مورد بحث قرار می‌دهیم تا هدف و طرز استفاده از دیتابیس واضح گردد.

۱.۱ تاریخچهٔ دیتابیس

مفهوم دیتابیس از دههٔ ۱۹۶۰ برای کاهش مشکلات فزاینده در طراحی، ساخت و نگهداشت سیستم‌های اطلاعاتی معمولاً با تعداد زیادی استفاده‌کننده و هم‌زمان با تعداد زیادی اطلاعات ایجاد شده‌است. این مفهوم به همراه مفهوم سیستم‌های مدیریت دیتابیس که استفادهٔ مؤثر و کارا به دیتابیس را ممکن می‌سازد، رشد کرده است.

اولین استفادهٔ اصطلاح دیتابیس به سال ۱۹۶۳ باز می‌گردد؛ یعنی زمانی‌که شرکت System Development Corporation مسئولیت اجرای یک طرح به نام «توسعه و مدیریت محاسباتی یک دیتابیس مرکزی» را بر عهده گرفت. دیتابیس به عنوان یک واژهٔ واحد در اوایل دههٔ ۷۰ در اروپا و در اواخر دههٔ ۷۰ در اخبار معتبر آمریکایی به کار رفت.

اولین سیستم مدیریت دیتابیس در دههٔ ۶۰ گسترش یافت. از پیش‌گامان این شاخه چارلز بکمن می‌باشد. مقالات بکمن این را نشان داد که فرضیات او کاربرد بسیار مؤثرتری برای دست‌رسی به وسایل ذخیره‌سازی را مهیا می‌کند. در آن زمان‌ها پروسس اطلاعات بر پایهٔ کارت‌های مقناطیسی و نوارهای مقناطیسی بود که پروسس سریع اطلاعات را مهیا می‌کرد. دو نوع مودل اطلاعاتی در آن زمان ایجاد شد: یکی CODASYL بود که موجب توسعهٔ مودل شبکه‌یی گردید که آن هم ریشه در نظریات بکمن داشت و دوم مودل سلسله مراتبی که توسط North American Rockwell ایجاد شد و بعداً با اقتباس از آن شرکت IBM محصول IMS را تولید نمود.

مودل رابطه‌یی توسط E. F. Codd در سال ۱۹۷۰ ارائه شد. او مودل‌های موجود را مورد انتقاد قرار می‌داد. برای مدتی نسبتاً طولانی این مودل در مجامع علمی مورد تایید بود. اولین محصول موفق برای کامپیوترهای خورد dBASE بود که برای سیستم‌عامل‌هایCP/M و PC-DOS/MS-DOSساخته شد. در جریان سال ۱۹۸۰ پژوهش بر روی دیتابیس مودل توزیع‌شده و ماشین‌های دیتابیس (Database Machine) متمرکز شد، اما تأثیر کمی بر بازار گذاشت. در سال ۱۹۹۰ توجه به طرف مودل شئ‌گرا جلب گردید. این مودل جهت کنترل اطلاعات مرکب به سادگی بر روی دیتابیس خاص، مهندسی اطلاعات (شامل مهندسی نرم‌افزار منابع) و اطلاعات با نوعیت‌های مختلف کار می‌کرد.

در سال ۲۰۰۰ میلادی نوآوری تازه‌یی رخ داد و مودل اکس ام ال (XML) به وجود آمد. هدف این مودل از بین‌بردن تفاوت بین مستندات و اطلاعات است و کمک می‌کند که منابع اطلاعاتی در کنار هم قرار گیرند.

۱.۲ دیتابیس چیست

دیتابیس (Database) به مجموعه‌یی از اطلاعات با ساختار منظم گفته می‌شود. این اطلاعات معمولاً به شکلی که برای کمپیوتر قابل خواندن و قابل دسترسی باشند، ذخیره می‌شوند. به عبارت دیگر، مجموعه‌یی اطلاعات تعریف‌شده یا سازمان‌دهی‌شده که به صورت منطقی با هم مرتبط باشند، به نام دیتابیس یاد می‌شود.

به هر چیزی (شئ، شخص، محل و...) که می‌خواهیم در یک سیستم راجع به آن اطلاعاتی را جمع‌آوری، پروسس و نگهداری نمائیم، یک جدول (Entity) گفته می‌شود. تعریف فوق، برداشت اولیه از جدول می‌باشد. مجموعه جدول‌های یک سیستم، ساختار اطلاعاتی آن سیستم را مشخص می‌کند. هر جدول شامل اجزایی است که آن جدول را توصیف می‌کند که به آن‌ها مشخصات و یا (Column) گفته می‌شود. هر جدول بسته به این‌که در سیستم مورد مطالعه چه میزان اطلاعات راجع به آن می‌خواهیم داشته باشیم، شامل حد اقل یک و یا چند مشخصه خواهد بود. از آن‌جا که هر جدول راجع به یک موضوع خاص می‌باشد، بنابراین یک ارتباط منطقی بین تمام مشخصات جدول وجود خواهد داشت. در واقع، تمام مشخصات یک جدول توصیف‌کنندهٔ آن جدول خواهد بود. برای روشن‌شدن موضوع، لازم است که به نمونهٔ مثال ذیل توجه نمایید:

جدول مشتری شامل مشخصات چون نام مشتری، آدرس مشتری، تلفن مشتری و... است.

جدول سفارش شامل مشخصات شمارهٔ سفارش، تاریخ سفارش، نام مشتری، اجناس سفارش‌شده، تعداد اجناس سفارش‌شده و... است.

شکل ۱-۱: نمایش دیتابیس

۱.۳ موارد استفادهٔ دیتابیس

ممکن است برای‌تان سوال پیش بیاید که اصلاً چرا باید از دیتابیس (Database) استفاده کنیم؟ سوال بسیار خوبی است. ممکن است شما بتوانید مجموعه‌یی از رستورانت‌های اطراف‌تان، شاگردان یک مکتب و... را بدون استفاده از دیتابیس و جدول‌ها، نگهداری کنید، ولی هنگامی‌که تعداد اقلام شما (رستورانت یا شاگردان)

زیاد می‌شود، دست‌رسی و جست‌وجو در این اطلاعات بسیار سخت و گاهی غیر ممکن می‌گردد. پس دیتابیس (Database) با استفاده از ساختاربندی منظمی که به اطلاعات ما می‌دهد، باعث می‌شود که اطلاعات ما در بلندمدت، بسیار منظم و یک‌پارچه باشند و دست‌رسی به آن‌ها نیز بسیار ساده گردد. بنابراین، در پاسخ به این سوال که «چرا باید از دیتابیس استفاده کنیم؟» می‌توانیم بگوییم که دیتابیس (Database) اطلاعات ما را در جدول‌ها قرار می‌دهد. این جدول‌ها، نگهداری اطلاعات را برای ما بسیار ساده و منظم می‌کنند و یک‌پارچگی بسیار خوبی به آن‌ها می‌دهند؛ برای مثال، می‌توان به یک کتابچهٔ تلفن که با نظم خاصی نوشته شده است و دارای سطر و ستون‌های یک‌پارچه می‌باشد، یک دیتابیس گفت.

تا این‌جا، مثال‌ها همه در دنیای واقعی بود. حالا می‌خواهیم دربارهٔ دنیای نرم‌افزارها و کاربرد دیتابیس صحبت کنیم.

یکی از مهم‌ترین جاهایی‌که دیتابیس(Database)ها به کار می‌روند، نرم‌افزارها هستند. می‌توانیم بگوییم که تقریباً تمام نرم‌افزارها، به وسیلهٔ دیتابیس (Database) قدرت واقعی خود را پیدا می‌کنند و بدون آن هیچ کار خاصی نمی‌توانند انجام دهند؛ برای مثال:

یوتیوب، یک دیتابیس عظیم از فایل‌های ویدئویی دارد.
یک سیستم مدیریت یک کتابخانه، نیاز به دیتابیسی ازکتاب‌ها و افرادی‌که در آن ثبت نام کرده‌اند، دارد.
همین‌طور یک نرم‌افزار مدیریت مکتب، نیاز به دیتابیسی از شاگردان آن مکتب دارد.

تمام مثال‌های فوق، قدرت اصلی خود را از دیتابیس‌ها می‌گیرند. پس کار دیتابیس (Database) در نرم‌افزارها و صفحات اینترنتی این است که مجموعه‌یی از اطلاعات مورد نیاز را در خود ذخیره می‌کنند، سپس آن اطلاعات را به نرم‌افزار تحویل می‌دهند و آن نرم‌افزار یک مجموعه عملیات و پروسس‌هایی با آن اطلاعات انجام می‌دهد.

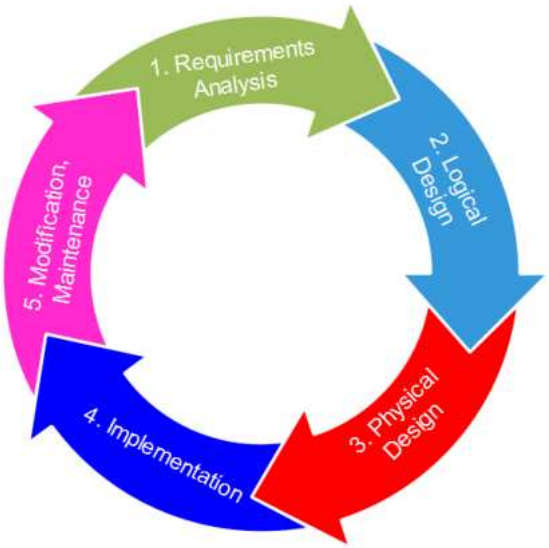
شکل ۱-۲: استفادهٔ دیتابیس

۱.۴ مراحل ساخت دیتابیس

برای طراحی دیتابیس مراحل ذیل را باید انجام داد:

۱. تجزیه و تحلیل نیازمندی‌ها (Requirement Analysis)
۲. طراحی مفهومی (Logical Design)
۳. طراحی فزیکی (Physical Design)
۴. پیاده‌سازی (Implementation)
۵. تعمیر و نگهداری (Modification Maintenan)

شکل ۱-۳ مراحل ساخت دیتابیس را نشان می‌دهد.



شکل ۱-۳: مراحل ساخت دیتابیس

تجزیه و تحلیل نیازمندی‌ها (Requirements of Analysis)

طی این مرحله، جامعیت و عملکرد مورد نظر کاربرد دیتابیس ها مشخص می گردند. بدین ترتیب، تجزیه و تحلیل نیازمندی‌ها باید جنبه‌های دوگانهٔ کاربرد مورد نظر را مورد توجه قرار دهد؛ یکی نیازمندی‌های اطلاعاتی تنظیمی که دیتابیس‌ها برای آن طراحی و ساخته می شود و دیگری نیازمندی‌های پروسس اطلاعات، یعنی نیازمندی برنامه‌هایی که با دیتابیس‌ها در تعامل هستند. این کار به طور معمول با انجام مصاحبه با استفاده‌کنندهٔ نهایی مورد نظر و برنامه‌نویس سیستم که از دیتابیس‌های تحت طراحی استفاده خواهند نمود، انجام می‌گیرد. تهیهٔ مستندی به زبان رسمی و یا نیمه‌رسمی که به عنوان اساس برای مراحل بعدی مورد استفاده قرار می‌گیرد.

طراحی منطقی (Logical Design)

طراحی منطقی مستقل از سیستم می باشد. در این قسمت در مورد تعین موضوع و صفات آن تصمیم گرفته می‌شود که پس از آن‌که طراحی مفهومی تکمیل گردید، طراحی پیاده‌سازی یا طراحی منطقی می‌تواند انجام گیرد. در این مرحله، طراحی مفهومی به مودل پیاده‌سازی سیستم دیتابیس‌های مورد استفاده تبدیل می‌گردد.

طراحی فزیکی (Physical Design)

در این مرحله طراحی مفهومی به مجموعه‌یی از دستورات SQL برای ایجاد دیتابیس تبدیل گردیده و نهادها (Entity)[1] تبدیل به جدول‌ها می‌گردد.

پیاده‌سازی(Implementation)
در این مرحله نصب DBMS و ایجاد دیتابیس به صورت عملی صورت می‌گیرد.
اصلاح و نگهداری (Modification and Maintenance)

از زمان اختراع مراحل ساخت دیتابیس تا به امروز مسألهٔ خرابی دیتابیس و رفع خرابی (اصلاح) گریبان‌گیر بشریت بوده است. در ضمن، همهٔ دست‌ساخته‌های بشر نیازمند فعالیت‌هایی بوده اند تا بتواند در صورت بروز خرابی، مشکل را رفع (اصلاح) نموده، ضمن سرکشی و انجام فعالیت‌های اغلب ساده، آن‌ها را در مقابل فرسایش و خرابی‌های ناشی از استفاده حفاظت نمایند. برای رسیدن به اهداف فوق، نیاز به استفاده از یک مجموعه امکانات از قبیل ابزار، تجهیزات، قطعات، منابع انسانی، فعالیت‌های اجرائی، فعالیت‌های نظارتی، مدارک، اطلاعات و...داریم.

۱.۵ سیستم دیتابیس (Database System)

سیستم دیتابیس یک سیستم کامپیوتری برای نگهداری و ثبت اطلاعات است؛ یعنی یک سیستم کامپیوتری که هدف آن ذخیرهٔ اطلاعات می‌باشد و استفاده‌کننده‌ها می‌توانند آن اطلاعات را بازیابی کنند.

یک سیستم دیتابیس از چهار مولفه تشکیل شده است: اطلاعات، سخت‌افزار، نرم‌افزار و استفاده‌کننده‌ها، که هر کدام از این چهار مولفه را در این‌جا به طور مختصر مورد بررسی قرار می‌دهیم.

۱.۵.۱ اطلاعات (Data)

اطلاعات ذخیره‌شده در دیتابیس سیستم، جزئی اساسی و مهم این سیستم می‌باشد. هر چیزی که برای یک فرد یا سازمان با ارزش باشد، اطلاعات نامیده می‌شود. بر اساس استفاده‌کننده‌های دیتابیس (Database

[1] Entity1 عبارت از موضوع یا Topic مشخص در دیتابیس که دارای صفات می‌باشد.

Users)، سیستم‌های دیتابیس می تواند یا یک استفاده‌کننده (Single User) یا چند استفاده‌کننده (Multi User) داشته باشد.

سیستم (Single User)

سیستمی است که در هر لحظه فقط یک نفر می‌تواند از دیتابیس استفاده کند، ولی در (Multi User) در هر لحظه چند نفر می‌توانند از دیتابیس استفاده نمایند. هدف اصلی استفادهٔ سیستم‌های (Multi User) این است که به استفاده‌کننده‌ها اجازه می‌دهد که از آن به عنوان یک سیستم (Single User) استفاده نماید. به طور کلی دیتابیس، به خصوص در سیستم‌های بزرگ هم به صورت مجتمع (Integration) و هم به صورت اشتراکی (Sharing) هستند.

جامعیت (Integration)

دیتابیس مجموعه‌یی از فایل‌های مجزاء است که بخشی از اطلاعات اضافی (Redundancy)، در دیدگاهی مبتنی بر جامعیت از بین آن‌ها حذف شده است. از دیدگاه‌های دیگر، جامعیت به معنای صحت اطلاعات، پروسس‌ها و پیروی از مقررات سیستم می‌باشد؛ به عنوان مثال، موجودی واقعی حساب‌های بانکی نباید منفی باشد و یا شخص نتواند بیش از موجودی خود از حسابش برداشت کند. از دیدگاه‌های دیگر، نوعی جامعیت به نام (Consistency) هم‌خوانی معروف است و به معنای این است که اقلام اطلاعات در تمام سیستم باهم در تضاد نباشد؛ به عنوان مثال، در نسخه‌یی از فایل بانک، موجودی شخص با نسخهٔ دیگر فایل که معادل آن است، متفاوت نباشد.

اشتراک (Sharing)

اطلاعات موجود در دیتابیس می‌توانند بین استفاده‌کننده‌های مختلف به اشتراک گذاشته شوند و استفاده‌کننده‌ها قادر باشند تا عملیات مختلفی را روی آن‌ها انجام دهند. واضح است که اشتراک حاصل از جامعیت می‌باشد. با جامعیت‌بخشیدن اطلاعات می‌توان از اطلاعات به طور اشتراکی استفاده کرد که در دیتابیس مشکلی پدید نیاید.

هر استفاده‌کننده فقط بخشی از اطلاعات دیتابیس را مورد استفاده قرار می‌دهد و توسط دیدگاه‌های مختلف با دیتابیس در تعامل است. اگر دیتابیس مشترک نباشد، دیتابیس شخصی یا استفاده‌کنندهٔ ویژه نامیده می‌شود.

۱.۵.۲ سخت‌افزار(Hardware)

قطعات سخت‌افزاری سیستم شامل موارد زیر است:

حافظه‌های جانبی (معمولاً دیسک‌ها) که برای ذخیرهٔ اطلاعات به کار می‌روند، وسایل I/O، پورت‌های I/O و غیره می‌باشند و هم‌چنان پروسسورهای سخت‌افزار و حافظه‌های اصلی آن‌ها اند که برای پشتیبانی از اجرای نرم‌افزار این سیستم به کار می‌روند.

۱.۵.۳ نرم‌افزار(Software)

بین دیتابیس فیزیکی (مثلاً اطلاعات ذخیره‌شده) و استفاده‌کننده‌های سیستم، لایه‌یی از نرم‌افزار وجود دارد که سیستم مدیریت دیتابیس یا (DBMS)نام دارد. امکاناتی از قبیل اضافه و حذف فایل‌ها (یا جدول‌ها)، بازیابی اطلاعات از آن‌ها و به‌هنگام‌رسانی اطلاعات این فایل‌ها و جدول‌ها از طریق سیستم مدیریت دیتابیس ارائه می‌شود. یکی از کارهایی که سیستم مدیریت دیتابیس انجام می‌دهد این است که استفاده‌کننده‌ها را از مواجه‌شدن با جزئیات سخت‌افزاری برحذر می‌دارد.

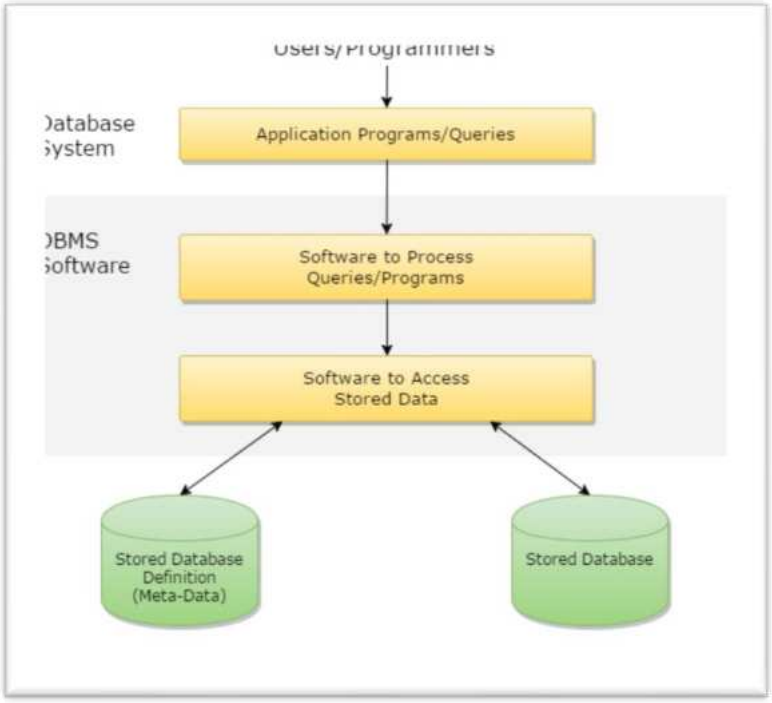
۱.۵.۴ استفاده‌کننده‌ها (Users)

استفاده‌کننده‌های دیتابیس را می‌توان به سه دسته تقسیم کرد:

دستهٔ اول برنامه‌نویسان کاربردی هستند که مسئول نوشتن برنامه‌های کاربردی دیتابیس اند که به زبان‌های مختلف برنامه‌نویسی می‌کنند. این برنامه‌ها با ارسال درخواست مناسب از طریقDBMS به دیتابیس دست‌رسی دارند.

دستهٔ دوم استفاده‌کننده‌های نهایی (End Users) هستند که از طریق ایستگاه‌های کاری به سیستم دست‌رسی دارند. هر استفاده‌کننده می‌تواند از طریق یکی از برنامه‌های کاربردی Online به دیتابیس دست‌یابی داشته باشند. این استفاده‌کننده‌ها هم‌چنین از طریق واسطی (GUI) که به عنوان بخشی از نرم‌افزار دیتابیس سیستم است، به دیتابیس دست‌یابی دارند.

سومین دسته از استفاده‌کننده‌ها مدیر (Administrator) دیتابیس (DBA) است که مدیریت دیتابیس را به عهده دارد. شکل ۱-۴ نشان‌دهنده دیتابیس سیستم با مولفه‌های آن می‌باشد.



شکل ۱-۴: محیط دیتابیس سیستم

۱.۶ سیستم مدیریت دیتابیس(Database Management System)

سیستم مدیریت دیتابیس یا DBMS (Database Management System) یک نرم‌افزار کمپیوتری است که با دیتابیس، استفاده‌کننده و سایر برنامه‌ها تعامل و ارتباط برقرار می‌کند و امکان تعریف، ایجاد، به‌روزرسانی، اعمال پرس‌وجو (Query) و به طور کلی مدیریت دیتابیس را فراهم می‌آورد.

اکثر سیستم‌های دیتابیس از مودل رابطه‌یی (Relational) پشتیبانی می‌کنند که به نام RDBMS (Relational Database Management System)) یاد می‌شوند و از معروف‌ترین آن‌ها می‌توان به موارد زیر اشاره نمود:

۱. MySQL
۲. Microsoft SQL Server
۳. Oracle
۴. PostgreSQL
۵. Microsoft Access

در بین دیتابیس و استفاده‌کنندگان سیستم به صورت فزیکی یک طبقه از سافت‌ویر موجود است که سیستم اداره یا مدیریت دیتابیس گفته می‌شود. به عبارت دیگر، DBMS مجموعه‌یی از برنامه‌های کمپیوتری بوده که توسط آن یک دیتابیس ایجاد، اجراء و اداره می‌گردد. DBMS دستورها را توسط لسان SQL دریافت نموده و آن‌ها را بالای دیتابیس اجراء می‌نماید.

۱.۶.۱ وظایف DBMS

از وظایف برجستهٔ یک DBMS در یک سیستم دیتابیس می‌توان به موارد ذیل اشاره نمود:

۱. ادارهٔ دکشنری اطلاعات (Meta Data)؛
۲. ادارهٔ ذخیرهٔ اطلاعات؛
۳. ایجاد دیتابیس؛
۴. ایجاد جدول‌ها؛
۵. ایجاد ساختمان‌های کمکی؛
۶. خواندن اطلاعات از یک دیتابیس؛
۷. تغییرآوردن در اطلاعات یک دیتابیس؛
۸. حفظ و نگهداری ساختمان‌های داخلی دیتابیس؛
۹. اجرای اوامر؛
۱۰. کنترول Concurrency؛
۱۱. امنیت یا Security؛
۱۲. Backup گیری از دیتابیس.

DBMS مسؤل جابه‌جاسازی اطلاعات در مورد عناصر و ساختمان دیتابیس یعنی (Meta Data) می‌باشد که به نام دکشنری اطلاعات نیز یاد می‌شود. DBMS، دکشنری اطلاعات را جهت پیدانمودن اطلاعات مورد نظر و Relationshipها در داخل دیتابیس مورد استفاده قرار می‌دهد؛ یعنی برای پیدانمودن یک Relationship ضرورت به داشتن کد نبوده، بلکه از طریق دکشنری اطلاعات پیدا می‌شود.

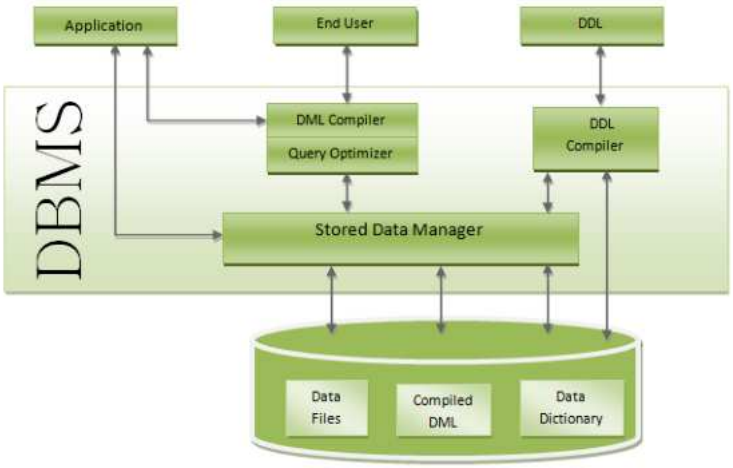
توسط DBMS، یک دیتابیس ایجاد شده و جدول‌ها به آن علاوه شده می‌تواند و هم‌چنان امکان علاوه‌نمودن ساختارهای کمکی چون Indexها را نیز فراهم می‌سازد؛ به طور مثال، اگر در یک دیتابیس یک جدول 1000 سطر موجود باشد و یکی از ستون‌های جدول عبارت از Department-Name باشد و ستون متذکره دیپارتمنت اعضای لست‌شده در جدول را مشخص نماید، دفعتاً ضرورت می‌شود تا اطلاعات مشخص‌شدهٔ این جدول نظر به Department-Name پیدا شود. از این‌که دیتابیس مورد نظر بسیار بزرگ می‌باشد، پیدانمودن اطلاعات مورد نظر، فرضاً از دیپارتمنت Accounting، وقت زیاد را در بر خواهد گرفت. بناءً ضرورت می‌افتد تا یک Index برای Department-Name ایجاد شود و نشان دهد که کدام شخص عضو کدام دیپارتمنت است. ایجاد چنین Indexها مثال خوبی برای ساختمان‌های کمکی می‌باشد که توسط DBMS ایجاد و اجراء می‌گردد.

از جمله وظایف دیگر DBMS، خواندن و تغییرآوردن در اطلاعات یک دیتابیس است. وظیفهٔ دیگری DBMS عبارت از حفظ و نگهداری ساختمان‌های داخلی دیتابیس می‌باشد؛ به طور مثال، ضرورت می‌افتد تا شکل یا فارمت جدول‌ها و یا دیگر ساختمان‌های کمکی در داخل یک دیتابیس Update شود. جهت اجرای این کار DBMS به صورت اتوماتیک اجراآت نموده و پروسهٔ Updateشدن را تکمیل می‌نماید. بسیاری از DBMSها

امکان اجرای بعضی اوامر را به استفاده‌کننده می‌دهد که این اوامر در استفاده از دیتابیس یک امر مهم به شمار می‌رود.

سه وظیفهٔ لست‌شدهٔ اخیر DBMSها بیش‌تر به اداره یا Administration دیتابیس ارتباط می‌گیرد. DBMS تصادف یا Concurrency را کنترول می‌نماید؛ به این معنا که کار یک استفاده‌کننده در عین دیتابیس با کار استفاده‌کنندهٔ دیگر، تصادف یا تداخل ننماید. هم‌چنان DBMS وظیفهٔ امنیت (Security) دیتابیس را به عهده دارد؛ به طور مثال، کارکنان هر بخش تنها اجازهٔ کار در یک قسمت معین دیتابیس را دارا می‌باشد و یا استفاده‌کنندگان تنها بخش‌هایی از دیتابیس را می‌بینند که برای کارهای مربوطهٔ آنان تعیین شده است، در حالی‌که مدیران یا طراحان دیتابیس قادر به دیدن قسمت‌های دیگر و تغییرآوردن در آن قسمت‌ها می‌باشند.

در اخیر باید افزود که دیتابیس یک سرمایهٔ مهم و پر ارزش خصوصاً برای کمپنی‌های بزرگ می‌باشد. پس برای طراحی و کار با دیتابیس باید از دقت زیاد کار گرفته شود، که تا حد ممکن اطلاعات به صورت مکمل در دیتابیس جابه‌جا گردد. این کار مستقیماً ارتباط می‌گیرد به رفع مشکلات Hardware و Software؛ سیستمی که DBMSها تا حد زیاد مانع به‌وجودآمدن چنین مشکلات می‌گردد. با مهیانمودن امکانات وسیع و گرفتن Backup اطلاعات، دقت و صحت کار را تضمین می‌نماید. در شکل ۱-۵ ساختار سیستم مدیریت دیتابیس و طرز کار آن نمایش داده شده است.

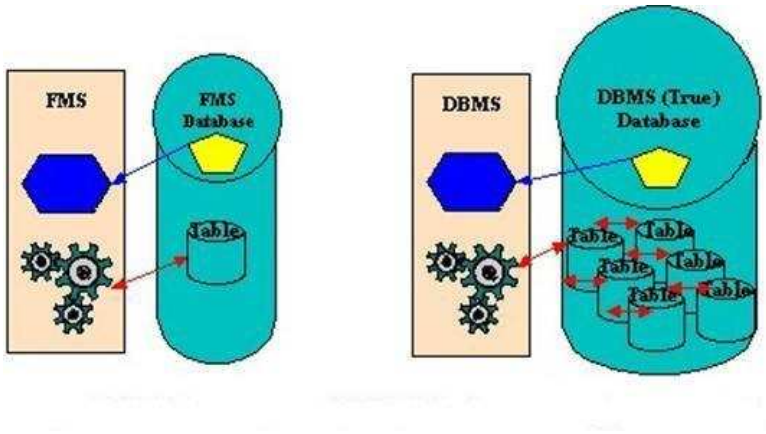


شکل ۱-۵: ساختار DBMS

۱.۷ مقایسهٔ سیستم مدیریت فایل با سیستم مدیریت دیتابیس

یک سیستم مدیریت دیتابیس یا DBMS (Database Management System) ترکیبی از نرم‌افزار کامپیوتر، سخت‌افزار و اطلاعاتی است که برای نگهداری الکترونیکی اطلاعات توسط کامپیوتر طراحی شده است. دو حالت سیستم مدیریت دیتابیس، یعنی DBMSها و FMSها هستند. به زبان ساده، یک سیستم

مدیریت فایل (File Management System) FMS می‌تواند یک سیستم مدیریت دیتابیس هم باشد که فقط اجازهٔ دست‌رسی به یک فایل یا جدول منفرد را در یک لحظه به ما می‌دهد. در شکل ۱-۶ طرز ذخیرهٔ اطلاعات در سیستم مدیریت فایل و سیستم مدیریت دیتابیس نمایش داده شده است.



شکل ۱-۶: طرز ذخیرهٔ اطلاعات

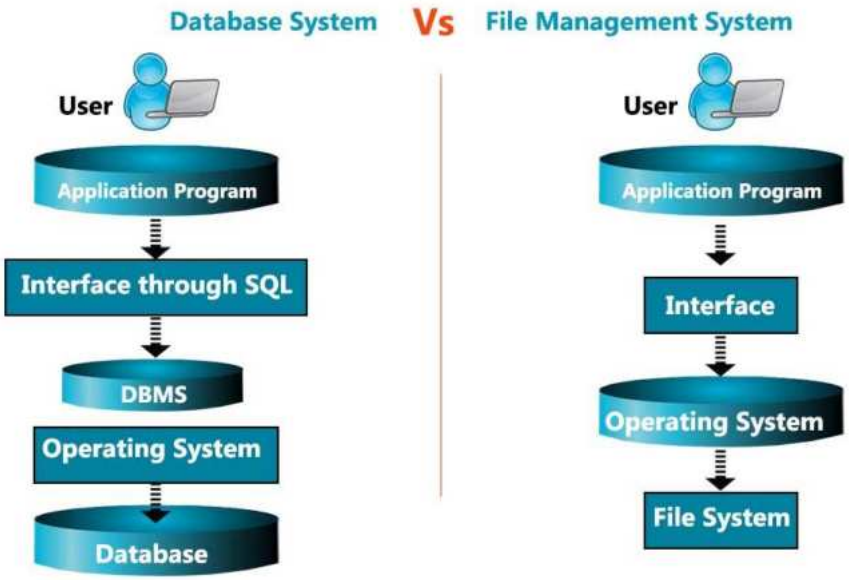
در سیستم‌های مدیریت فایل، فایل‌ها هیچ رابطه‌یی با یک‌دیگر ندارند و در واقع اجداد همین سیستم‌های دیتابیسی هستند که دست‌رسی هم‌زمان به چند فایل یا جدول را ممکن می‌نمایند.

جدول ۱-۱: فواید و نواقص سیستم مدیریت فایل

فواید	نواقص
آسان‌بودن استفاده	به طور معمول امکان استفادهٔ هم‌زمان چند استفاده‌کننده را ندارد.
قیمت ارزان	برای دیتابیس‌های کوچک‌تر مناسب است.
نیازهای اغلب استفاده‌کننده‌های خانگی یا تجاری کوچک را برآورده می‌کند.	امکانات محدود، مثلاً امکان معاملات پیچیده یا بازیابی و ... را ندارد.
FMSها معمولاً همراه سیستم‌عامل کامپیوترهای شخصی ارایه می‌شوند.	اطلاعات غیر متمرکز
مناسب برای نرم‌افزارهای دیتابیسی دستگاه‌های کوچک	مشکلات تکرار یا عدم جامعیت (Integrity) اطلاعات

جدول ۲-۱: فواید و نواقص سیستم مدیریت دیتاب

فواید	نواقص
انعطاف بیش‌تر	سختی در یادگیری
مناسب برای دیتابیس‌های بزرگ‌تر	پکیج مجزا از سیستم‌عامل
قدرت پروسس بالاتر	سرعت پروسس پایین‌تر
نیازهای اغلب سازمان‌های متوسط یا کوچک را پوشش می‌دهد.	به مدیر سیستم‌های با تجربه نیاز دارد.
امکان ذخیره‌سازی همهٔ اطلاعات مربوط	قیمت بالا
امکان مشاهدهٔ کارهای انجام‌شدهٔ سیستم توسط استفاده‌کننده	
اطمینان از صحت و جامعیت اطلاعات از طریق مدیریت معاملات ACID test = Atomicity, Consistency, Isolation,) (Durability	
امکان دسترسی هم‌زمان توسط چند استفاده‌کننده	
الزام به رعایت دقیق شرایط و ضوابط طراحی برای فارمت اطلاعات و ساختار آن‌ها	
امکانات Backup گیری و بازسازی و ترمیم مستقل از سیستم‌عامل	
امنیت پیشرفته‌تر	



شکل ۱-۷: مقایسهٔ سیستم مدیریت فایل و سیستم مدیریت دیتابیس

خلاصهٔ فصل اول

دیتابیس در مفهوم عام آن، به مجموعه‌یی از اطلاعات با ساختار منظم و سازمان‌یافته گفته می‌شود. در این مفهوم، ذخیره‌سازی سادهٔ اطلاعات در یک فایل را نیز می‌توان نوعی از دیتابیس دانست، اما در مفهوم خاص، منظور از دیتابیس مجموعه‌یی از این اطلاعات است که به شکلی ذخیره شده که توسط ابزارهای الکترونیکی قابل خواندن و دسترسی است.

سیستم دیتابیس یک سیستم کامپیوتری برای نگهداری و ثبت اطلاعات است که متشکل از اطلاعات، سخت‌افزار، نرم‌افزار و استفاده‌کننده‌های دیتابیس می‌باشد.

سیستم مدیریت دیتابیس یا به طور خلاصه DBMS (Database Management System) مهم‌ترین نرم‌افزار در دیتابیس سیستم است که به عنوان رابط بین دیتابیس، استفاده‌کننده و برنامه‌های کاربردی عمل می‌نماید. تمام فایل‌های دیتابیس فقط در اختیار این نرم‌افزار قرار گرفته و دست‌یابی به آن‌ها تنها از طریق DBMS امکان‌پذیر است.

سیستم مدیریت دیتابیس توسعه‌یافتهٔ سیستم مدیریت فایل است. بخشی از این تکامل همان نیاز به دیتابیس پیچیده‌تری است که سیستم مدیریت فایل امکان آن را ندارد (مانند ارتباطات داخلی بین جدول‌ها). با این وجود هنوز برای دیتابیس یک فایل (یک جدول) نیاز به سیستم مدیریت فایل به عنوان یک ابزار کاربردی داریم. انتخاب یک DBMS برای توسعهٔ دیتابیس‌های رابطه‌یی می‌تواند هزینه بر باشد.

سوالات و فعالیت های فصل اول

۱. دیتابیس چیست؟

۲. موارد استفادهٔ دیتابیس را بیان دارید.

۳. دیتابیس سیستم چیست؟

۴. مولفه‌های شامل دیتابیس سیستم را بیان دارید.

۵. سیستم مدیریت فایل و سیستم مدیریت دیتابیس را از هم تفکیک نمایید.

۶. فواید و نواقص سیستم مدیریت فایل را بیان دارید.

۷. فواید و نواقص سیستم مدیریت دیتابیس را بیان دارید.

۸. سیستم مدیریت دیتابیس را تعریف نمایید.

۹. وظایف عمدهٔ سیستم مدیریت دیتابیس را بیان دارید.

۱۰. استفادهٔ دیتابیس از کدام مشکلات ذخیره‌سازی اطلاعات جلوگیری می‌کند؟

فعالیت ها

۱. مورد استفادهٔ دیتابیس را در یک مثال واضح سازید. (انفرادی)

۲. در مورد نحوهٔ کارکرد سیستم مدیریت دیتابیس معلومات ارائه نمایید. (گروپی)

فصل دوم

معرفی مودل‌های دیتابیس
(Database Modeling Concepts)

هدف کلی: آشنایی محصلان با مودل‌های دیتابیس.

اهداف آموزشی: در پایان این فصل، محصلان قادر خواهند بود تا:

۱. مودل‌های دیتابیس را توضیح دهند.
۲. انواع مودل‌های دیتابیس را نام ببرند.
۳. تفاوت مودل‌های دیتابیس را بفهمند.

در این فصل مودل‌های دیتابیس را معرفی می‌نماییم و هم‌چنان انواع مختلفی چون مودل سلسله‌مراتبی، مودل رابطه‌یی، مودل شبکه‌یی، مودل شئ-رابطه و مودل شئ‌گرا را شرح می‌دهیم تا تفاوت بین انواع مختلف مودل‌های دیتابیس واضح گردد. از این‌که مودل رابطه‌یی یکی از پرکاربردترین مودل‌ها است، بناءً در این فصل به تشریح بیش‌تر این مودل می‌پردازیم.

۲.۱ مودل‌های دیتابیس (Database Modeling)

مودل اطلاعات زیربنای ساختاری دیتابیس را مشخص می‌کند و شامل مجموعه‌یی از اشیای مرتبط به هم و عمل‌گرها است که توسط سیستم مدیریت دیتابیس یا DBMS مورد پشتیبانی واقع می‌شود و هدف از وجود آن درک اطلاعات در دیتابیس می‌باشد.

مودل‌های دیتابیس نشان‌دهندهٔ ساختار منطقی دیتابیس می‌باشد که شامل ارتباطات، محدودیت‌ها و طرز ذخیره و بازیابی اطلاعات است.

مودل‌های دیتابیس به سه گروه اصلی تقسیم می‌شوند:

مودل‌های منطقی مبتنی بر اشیاء (Object Based Data Model)
مودل‌های منطقی مبتنی بر سطرها (Record Based Data Model)
مودل‌های فزیکی (Physical Data Model)

انواع مودل‌های دیتابیس (Types of Database Model)

مودل‌های دیتابیس انواع زیاد دارد که مشهورترین آن‌ها قرار ذیل می‌باشند:

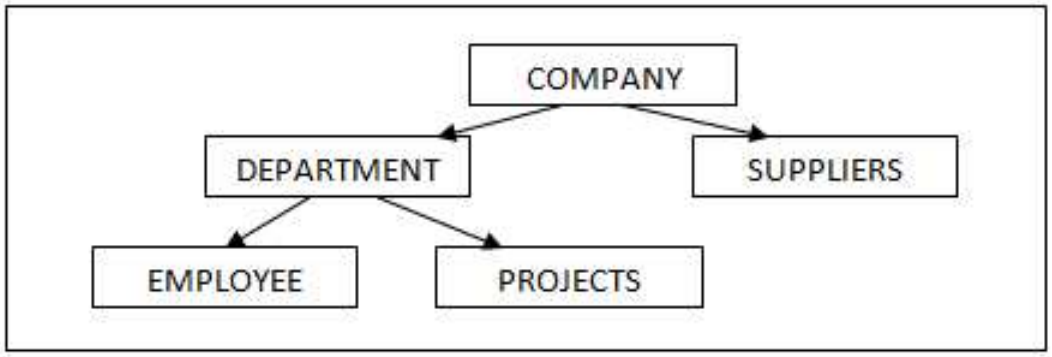
- مودل سلسه‌مراتبی (Hierarchical Database Model)
- مودل رابطه‌یی (Relational Model)
- مودل شبکه‌یی (Network Model)
- مودل شئ-رابطه (Object-Relational Model)
- مودل شئ‌گرا (Object-Oriented Database Model)

مودل سلسه‌مراتبی، مودل رابطه‌یی و مودل شبکه‌یی شامل گروهی مودل‌های منطقی مبتنی بر رکوردها (Record Based Data Model) بوده و مودل شئ‌گرا و مودل شئ-رابطه شامل گروهی مودل‌های منطقی مبتنی بر اشیاء (Object Based Data Model)می‌باشد. مودل‌های فزیکی که برای توصیف اطلاعات در پایین‌ترین سطح به کار می‌رود، برای حفظ اطلاعات در کامپیوتر است و دارای اطلاعاتی در مورد ساختار رکوردها، ترتیب رکوردها و دست‌یابی رکوردها می‌باشد.

۲.۱.۱ مودل سلسه‌مراتبی (Hierarchical Database Model)

این مودل قدیمی‌ترین مودل اطلاعات برای طراحی دیتابیس در سطح انتزاع (Conceptual Level) است. این مودل از این جهت که اطلاعات و ارتباط میان آن‌ها به وسیلهٔ رکوردها و پیوندها نشان داده می‌شود، همانند مودل شبکه‌یی است؛ با این تفاوت که در مودل سلسله‌مراتبی اطلاعات و ارتباط بین آن‌ها به کمک یک درخت قابل نمایش است. این درخت، مودلی است که یک ریشه دارد و دوران در آن وجود ندارد. بین هر دو گره (Node) یک رابطه وجود دارد و هر گره پدر (Parent) می‌تواند دارای یک یا چند فرزند (Child) باشد ولی هر فرزند تنها دارای یک پدر است. این مودل عمل‌گرهای ذخیره و بازیابی ساده دارد و دارای محیط انتزاعی مسطحی نیست. شکل ۲-۱ یک نمونه از مودل سلسله‌مراتبی می‌باشد.

این مودل ابتداء در دههٔ 60 و 70 توسط شرکت IBM استفاده می‌شد ولی امروزه کم‌تر مورد استفاده قرار می‌گیرد.



شکل ۲-۱: مودل سلسه‌مراتبی

فواید

۱. سرعت دست‌یابی بالا به حجم عظیم اطلاعات؛
۲. ساده‌بودن ساختار.

نواقص

۱. عدم امکان جست‌وجو در فیلدهای غیر توصیفی که در صورت نیاز باید دیتابیس بازسازی شود و نیز دامنهٔ تمام پرسش‌ها از قبل باید معلوم باشد؛
۲. مشکل‌بودن اصلاح ارتباط اطلاعات؛
۳. کارایی کم بازیابی اطلاعات به علت نیاز به پیمودن سلسله‌مراتب برای دست‌یابی به اطلاعات واقع در سطوح پایین درخت؛
۴. غیر قابل انعطاف‌بودن؛

۵. عدم امکان وجود چند گره پدر؛
۶. عدم توافق در کاربردهای GIS ؛
۷. افزونی اطلاعات.

۲.۱.۲ مودل رابطه‌یی (Relational Model)

در این مودل، دیتابیس از چندین جدول تشکیل شده است. جدول ساختاری است که از چند ستون و تعدادی سطر تشکیل یافته است. این در حالی است که ستون‌ها بیان‌گر صفات خاصه از یک نوع جدول یا شئ و سطرها نیز نمایان‌گر نمونه‌یی از آن شئ است. برخی خصوصیات مودل رابطه‌یی عبارت‌اند از عمل‌گرهای ذخیره و بازیابی ساده و رویهٔ پاسخ‌گویی واحد برای پرس‌وجوها. در ضمن، اطلاعات و ارتباطات بین آن‌ها در این مودل، با مکانیزم خاص به نام سطر نشان داده شده اند.

مودل رابطه‌یی دیتابیس‌ها برای بار اول توسط محقق آمریکایی (E. F. Codd) در شرکت IBM در سال 1985 پیشنهاد گردید. بعد از گذشت مدتی در سال 1985 همین شخص قانون 13 فقره‌یی (1+12) مودل رابطه‌یی را پیش‌کش نمود که این قانون تا به امروز عملی بوده و استفاده می‌شود.

در پایین، لیست مکمل قانون (E. F. Codd) همراه با نام‌های اصلی (Original) هر بند و توضیح آن‌ها ذکر شده است.

۱. اساس (Foundation): در مودل رابطه‌یی یک دیتابیس باید به صورت مکمل به اساس ساختمان رابطه‌یی آن مدیریت شده و قابل دسترسی باشد.
۲. معلومات (Information): تمام معلومات باید در حجره‌های جدول به شکل دو بُعدی و به صورت محتوای جداگانه در دیتابیس ثبت شده بتواند.
۳. دست‌یابی تضمین‌شده (Guaranteed Access): تمام اطلاعات ذخیره‌شده در دیتابیس بدون کدام ابهام باید قابل دسترسی باشد. این دسترسی می‌تواند با استفاده از نام جدول، کلید اولیهٔ جدول و نام ستون جدول صورت گیرد.
۴. قبول‌کردن محتوای ناشناخته (Systematic Treatment Of Null Values): محتواهای ناشناخته (Null Values) عبارت از معلوماتی است که نوعیت و محتوای آن هیچ‌کدام قیمت نباشد. البته محتوای ناشناخته به ستون‌های اولیه (Primary-Key)، ستون‌های فهرست (Index Fields) و ستون‌هایی که (Not Null) تعریف شده باشند، وارد شده نمی‌توانند، اما در ستون‌های عادی جدول‌ها در مودل رابطه‌یی باید وارد شده بتوانند. قابل یادآوری است که (Null) معادل صفر یا خانه‌خالی نمی‌باشد بلکه به معنای نبود معلومات و یا معلومات غیر قابل اجراء می‌باشد.
۵. داینامیک آنلاین کاتلوگ مبنی بر مودل رابطه‌یی (Dynamic Online Catalog Based On Relational Model): توضیحات دیتابیس در سطح منطقی مانند اطلاعات عادی موجود می‌باشد. به این خاطر استفاده‌کننده‌هایی دارای مجوز با استفاده ار عین لسان رابطه‌یی درخواست (Query) می‌کنند.

۶. استفاده از زبان فرعی (Comprehensive Data Sublanguage): دیتابیس رابطه‌یی می‌تواند از زبان‌های متعدد پشتیبانی نماید، اما باید حد اقل یک زبان جامع را حمایت نماید که در آن زبان دستورهای تعریف اطلاعات، ادارهٔ اطلاعات، دقت اطلاعات و غیره شامل باشند. تمام دیتابیس‌های رابطه‌یی تجارتی زبان (SQL) را که دساتیر یادشده در آن استفاده شده می تواند، حمایه نموده و استفاده می نمایند.

۷. تازه‌کردن نما (View Updating): اطلاعات شامل یک دیتابیس با استفاده از نما (View) که به شکل منطقی ترتیب شده باشند، قابل نمایش است. هر نما در دیتابیس رابطه‌یی باید قابلیت تازه‌شدن (Updating) را داشته باشند.

۸. واردکردن، تازه‌کردن و حذف اطلاعات به سطح بالا (High-Level Insert, Update and Delete): واردکردن، تازه‌کردن و حذف اطلاعات دیتابیس رابطه‌یی به چندین ستون و به چندین جدول به طور هم‌زمان و به شکل گروپی باید ممکن باشد.

۹. استقلال فزیکی اطلاعات (Physical data independence): در دیتابیس‌های رابطه‌یی، استفاده‌کنندگان باید از شکل فزیکی اطلاعات و شکل ذخیره‌سازی اطلاعات به کلی مجزاء باشند. هر زمانی‌که در شکل فزیکی ثبت اطلاعات، موقعیت اطلاعات ذخیره شده و یا هم وسایل ذخیرهٔ اطلاعات کدام تغییری وارد شود، در روش استفاده از آن باید کدام تغییری وارد نگردد.

۱۰. استقلال منطقی اطلاعات (Logical Data Independence): در دیتابیس‌های رابطه‌یی طریقه‌های نمایش اطلاعات با به‌میان‌آمدن تغییرات در شکل منطقی اطلاعات باید تغییر ننماید. تغییرات در ساختمان جدول‌ها و نوع اطلاعات ستون‌ها باعث ایجاد تغییرات در نمایش اطلاعات نشود.

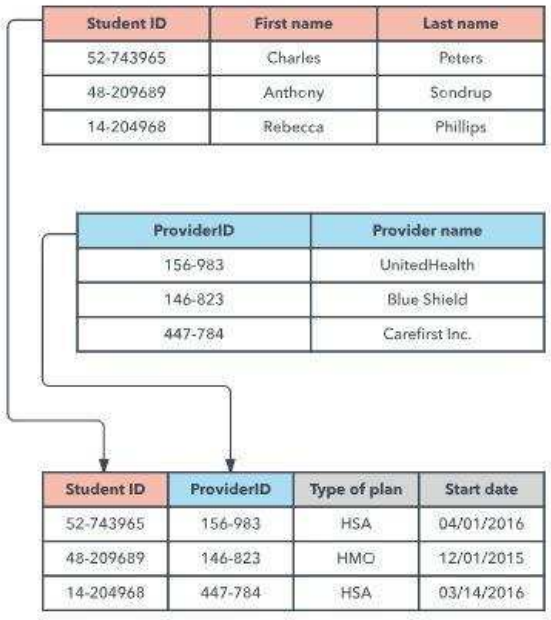
۱۱. استقلال درستی اطلاعات (Integrity Independence): زبان دیتابیس مانند (SQL) باید درست‌بودن دستورهای واردشده به دیتابیس را حمایه نماید و در زمان اجرای دساتیر، قوانین وضع‌شده توسط استفاده‌کننده را به صورت درست تطبیق نماید. این قانون به صورت کامل در دیتابیس‌ها تطبیق نشده است، ولی حداقل مسایل ذیل بر مبنای این قانون در دیتابیس‌ها تطبیق شده اند:

- کلیدهای اولیه (Primary-Keys) جدول‌های محتویات ناشناخته (Null Values) را قبول نمی‌نمایند.
- محتوای کلید خارجی (Foreign-Key) در جدول فرعی (Child) باید ست فرعی کلید اولیهٔ جدول اصلی (Parent) باشد.

۱۲. استقلال توزیع (Distribution Independence): پخش و توزیع یک دیتابیس در اضافه از یک موقعیت باید مشکلی ایجاد ننماید. در صورت نصب قسمت‌های دیتابیس در چندین کمپیوتر با موقعیت‌های مختلف، برای استفاده‌کننده باید قسمی معلوم شود که تمام اطلاعات در یک مکان موقعیت دارد و از توزیع اطلاعات در موقعیت‌های مختلف اطلاع نداشته باشد.

۱۳. عدم تخریب (Non-subversion): اگر یک سیستم رابطه‌یی از زبان سطح پائین استفاده کند، دستورهای زبان سطح پایین باید در ساختمان دیتابیس کدام تغییرات وارد ننمایند؛ تغییراتی از قبیل تطبیق‌نکردن قوانین درست (Integrity Rules) و محدودیت‌ها (Constraints) که توسط

زبان‌های رابطه‌یی سطح بالا مشخص شده است. نمونه‌یی از مودل رابطه‌یی در شکل ۲-۲ نمایش داده شده است.



شکل ۲-۲: مودل رابطه‌یی

فواید قانون کد

۱. انعطاف‌پذیری بالا و عدم محدودیت در پرسش‌دهی، یا به عبارت دیگر، امکان انجام جست‌وجو در هر جدول و بر اساس هر ستون توصیف؛
۲. عدم نیاز به ذخیره‌سازی جدول الحاقی (Join) و جلوگیری از تولید افزونی اطلاعات؛
۳. عادی‌بودن جدول‌هایی که باعث ایجاد انسجام (Integrity) و حد اقل افزونی می‌شود؛
۴. افزونی کم‌تر نسبت به مودل‌های سلسله‌مراتبی و شبکه‌یی؛
۵. دارای پایهٔ تئوری دقیق و ریاضی؛
۶. سادگی ساختار؛
۷. سادگی ایجاد تغییرات؛
۸. وجود زبان استندرد SQL (Structured Query Language) که در RDBMS های مختلف کاربرد دارد؛
۹. قابلیت استفاده SQL در مودل‌های رابطه‌یی مختلف؛
۱۰. رایج‌ترین مودل در نرم‌افزارهای تجاری GIS و کاربردهای GIS[2] به علت انعطاف‌پذیری بالا.

---

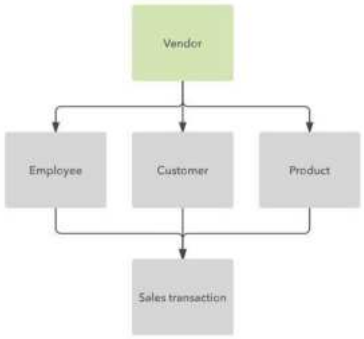
2 GIS (Geographical Information System)

نواقص

۱. سخت‌بودن پیاده‌سازی و اجراء؛
۲. کارآیی کم‌تر و کندبودن سیستم به خاطر عدم وجود ارتباطات فزیکی یا اشاره‌گرها (Pointers)؛
۳. مناسب‌نبودن برای کار با اشیای پیچیده؛
۴. میزان بالای بروز اشتباهات منطقی به علت انعطاف‌پذیری بالای این مودل.

۲.۱.۳ مودل شبکه‌یی (Network Model)

مودل شبکه‌یی همراه با مجموعه‌یی از رکوردها مشخص می‌شوند که ارتباط بین اطلاعات با پیوندهای همانند اشاره‌گر نمایش داده می‌شود. این مودل همانند یک گراف است و هر گره (Node) فرزند می‌تواند یک یا چند گره پدر داشته باشد. برخی خصوصیات مودل شبکه‌یی عبارت اند از: فاقد محیط انتزاعی مسطح، عمل‌گرهای ذخیره و بازیابی پیچیده، کارکرد سخت و نیز برای محیط‌های دارای ارتباط یک به چند (دو طرفه) مودل مناسبی است و ارتباط یک به چند (یک طرفه) حالتی خاص از این مودل است. در شکل ۲-۳ نمونه‌ای از مودل شبکه‌یی نمایش داده شده است.



شکل ۲-۳: مودل شبکه‌یی

فواید

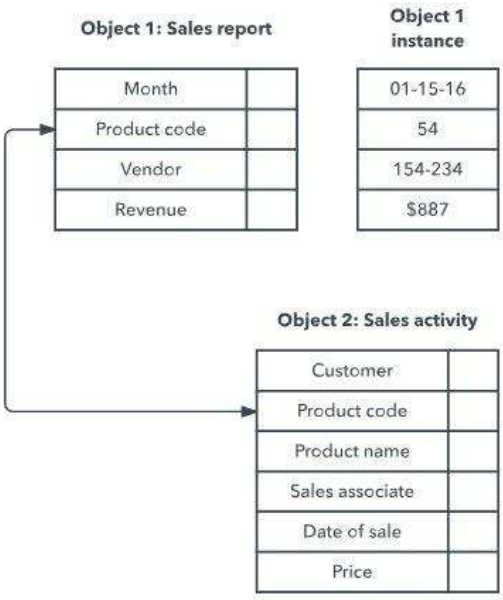
۱. دست‌رسی به رکوردها بدون نیاز به پیمایش تمامی سلسله‌مراتب بالای سطر؛
۲. عدم نیاز به ریشه (Root)؛
۳. امکان وجود چند پدر برای یک فرزند؛
۴. امکان ارتباط چند به چند (Many-To-Many) ؛
۵. انعطاف‌پذیری بیش‌تر نسبت به مودل سلسله‌مراتبی؛
۶. افزونی کم‌تر اطلاعات نسبت به مودل سلسله‌مراتبی؛
۷. سازگاری بیش‌تر با پیچیدگی‌های جهان واقعی؛
۸. سرعت بالای بازیابی به علت کدگذاری قبل از قبل گره (Node) اطلاعات (اما نسبت به سلسله‌مراتبی سرعت بازیابی کم‌تر است).

نواقص

۱. پیچیده‌بودن مودل؛
۲. بزرگ‌بودن حجم فایل به علت نیاز به ذخیره‌سازی اطلاعات بیش‌تر مربوط به ارتباطات؛
۳. نیاز به زمان بیش‌تر برای بازسازی اطلاعات در صورت ایجاد تغییر؛
۴. عدم توافق در کاربردهای GIS به علت انعطاف‌پذیری بالاتر مودل رابطه‌یی برای کاربردهای GIS.

۲.۱.۴ مودل شئ–رابطه‌یی (Object–Relational Model)

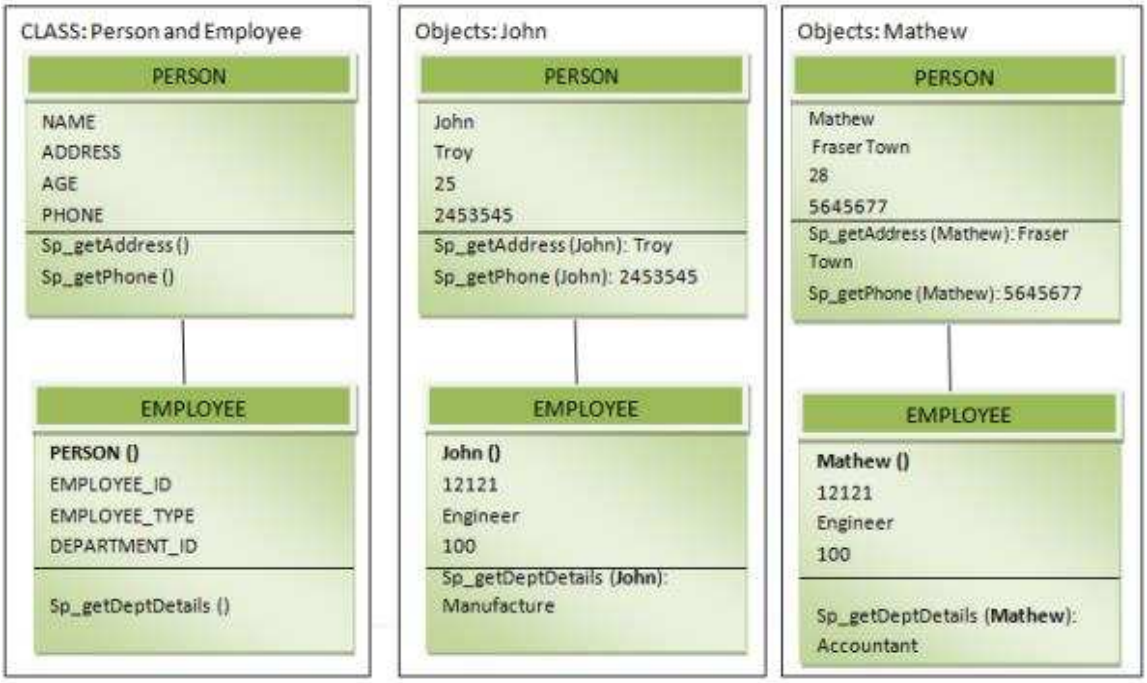
این مودل، شامل دو مجموعه از اشیاء است که بر مبنای ادراک و منطقی از یک دنیای حقیقی به دست می‌آیند. این دو مجموعه، شئ و رابطه بین اشیاء می‌باشند. در یک دیتابیس همه اشیای هم‌نوع در یک مجموعه شئ واقع می‌شوند و هم‌چنین تمامی روابط هم‌نوع در یک مجموعه رابطه قرار می‌گیرند. شکل ۲-۴ نشان‌دهندهٔ مودل شئ-رابطه می‌باشد.



شکل ۲-۴: مودل شئ- رابطه

۲.۱.۵ مودل شئ‌گرا (Object–Oriented Database Model)

مودل شئ‌گرا نیز مانند مودل شئ – رابطه بر مبنای اشیاء پایه‌گذاری شده است؛ با این تفاوت که هر شئ علاوه بر ویژگی‌هایی که دارد و ارتباطی که می‌تواند با اشیای دیگر داشته باشد، دارای زیرطریقه‌هایی است که روی آن شئ اعمال خاصی را انجام می‌دهند. به هر یک از این طریقه‌ها یک میتود (Method) می‌گویند و همه اشیایی که دارای مجموعه‌یی از ویژگی‌های یک‌سان و میتودهای مشابهی باشند، در قالب یک کلاس واقع می‌شوند. در شکل ۲-۵ مثال از مودل شئ‌گرا نمایش داده شده است.



شکل ۲-۵: مودل شیءگرا

خلاصهٔ فصل دوم

مودل‌های دیتابیس نشان‌دهندهٔ ساختار منطقی دیتابیس می‌باشد که شامل ارتباطات، محدودیت‌ها و طرز ذخیره و بازیابی اطلاعات است. مودل‌های دیتابیس انواع متعدد دارد که مشهورترین و پرکابردترین‌های آن از قبیل مودل سلسله‌مراتبی، مودل رابطه‌یی، مودل شبکه‌یی، مودل شئ‌گرا و غیره می‌باشد.

مودل سلسه‌مراتبی، مودل رابطه‌یی و مودل شبکه‌یی شامل گروه مودل‌های منطقی مبتنی بر رکوردها (Record Based Data Model) بوده و مودل شئ‌گرا و مودل شئ-رابطه شامل گروه مودل‌های منطقی مبتنی بر اشیاء (Object Based Data Model)می‌باشد. مودل‌های فزیکی که برای توصیف اطلاعلات در پایین‌ترین سطح به کار می‌رود، برای حفظ اطلاعات در کامپیوتر است و دارای اطلاعاتی در مورد ساختار رکوردها، ترتیب رکوردها و دست‌یابی رکوردها می‌باشد.

از این میان مودل رابطه‌یی امروزه استفادهٔ بیش‌تر دارد که در این فصل به تفصیل ذکر شده است. از جمله فواید و نواقص آن و هم‌چنان قانون (1+12) محقق آمریکایی (E. F. Codd) که هر کدام آن مفصل شرح داده شده است.

سوالات و فعالیت های فصل دوم

۱. مودل‌های دیتابیس چیست؟

۲. مودل‌های دیتابیس به چند گروه تقسیم شده است؟ شرح دهید.

۳. مودل سلسله‌مراتبی را با فواید و نواقص آن تشریح نمایید.

۴. مودل شبکه‌یی را با فواید و نواقص آن تشریح نمایید.

۵. مودل رابطه‌یی را با فواید و نواقص آن تشریح نمایید.

۶. فوانین (1+12) در مورد مودل رابطه‌یی را به طور خلاصه توضیح دهید.

۷. مودل شئ-رابطه را با مودل شئ‌گرا مقایسه نمایید.

فعالیت ها

۱. قوانین (۱+۱۲) در مورد مودل رابطه‌یی را مورد بحث قرار دهید. (انفرادی)

۲. چند مودل دیتابیس (به جز از مودل‌هایی که در این فصل به آن اشاره شده است) را بعد از تحقیق در مورد مودل‌های دیتابیس دریافت نمایید. (انفرادی)

فصل سوم

معرفی جدول‌های دیتابیس (Database Tables)

هدف کلی: آشنایی محصلان با جدول‌های دیتابیس.

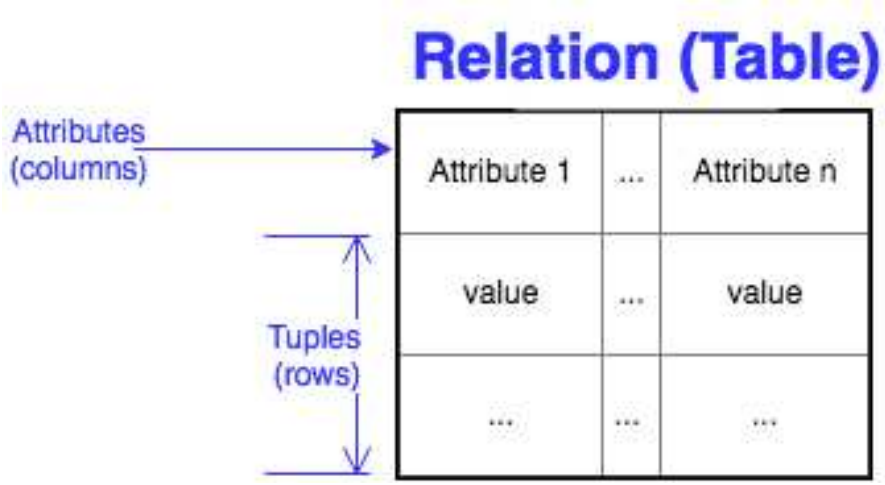
اهداف آموزشی: در پایان این فصل، محصلان قادر خواهند بود تا:

۱. ساختار جدول‌های دیتابیس را بدانند.
۲. نوعیت اطلاعات (Data Type) را در جدول‌ها توضیح دهند.
۳. ستون‌های کلیدی جدول‌ها را تشخیص دهند.
۴. وضع محدودیت‌ها (Constraints) را بالای ستون‌های جدول‌ها یاد گیرند.

در این فصل جدول را به معرفی گرفته، اجزایی که یک جدول از آن تشکیل می‌گردد (ستون‌ها و سطرها)، مورد بحث قرار می‌دهیم. بر علاوه نوعیت اطلاعات برای ستون‌ها، انتخاب ستون کلیدی و تطبیق محدویت‌ها بالای یک جدول را خواهیم آموخت.

۳.۱ جدول (Table)

یک جدول مجموعه‌یی از عناصر اطلاعات (مقادیر) می‌باشد که با استفاده از یک مودل ستون عمودی (که با نام خودشان شناخته می‌شوند) و سطر افقی سازماندهی شده‌است. یک جدول تعدادی ستون‌های مشخص، اما هر تعداد سطری می‌تواند داشته باشد. هر سطر توسط مقادیر موجود در زیر مجموعهٔ یک ستون خاص شناخته می‌شود. برای وضاحت بیشتر شکل ۳-۱ مشاهده کنید.

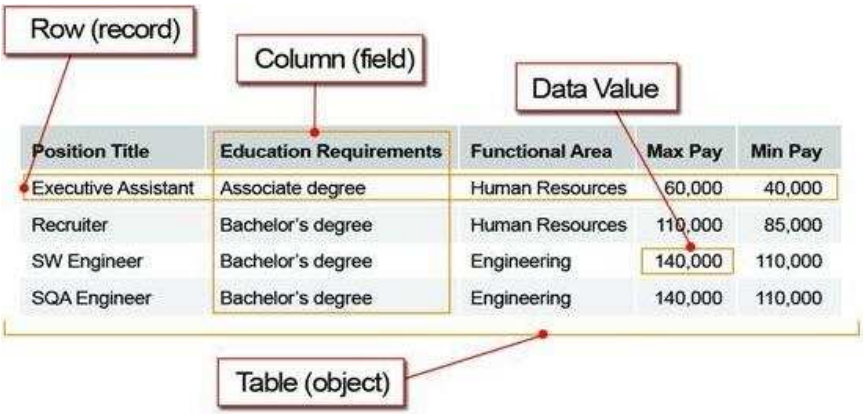


شکل ۳-۱: ساختار جدول

اطلاعات در دیتابیس داخل جدول یا (Table)ها ذخیره می‌شوند. هر دیتابیس می‌تواند شامل چندین جدول باشد. هر جدول از تعدادی سطر و ستون تشکیل شده‌است.

برای تمام ستون‌ها در دیتابیس نظر به نوع کارکرد آن می‌توانید نوعیت (Data Type) مورد نظر خود را تعریف کنید. مقدارهایی که در هر ستون (Column) ذخیره می‌شود باید با نوع انتخاب ستون مطابقت داشته باشد.

جدول مجموعه‌یی است از اطلاعات ثبت‌شدهٔ مرتبط و وابسته به هم که از ستون‌ها (Columns) و سطرها (Rows) تشکیل شده است. جدول‌ها مهم‌ترین عناصر سیستم‌های دیتابیس هستند که برای ذخیره و نگهداری سازمان‌یافتهٔ اطلاعات مورد استفاده قرار می‌گیرند. جدول بخشی از دیتابیس است و یک دیتابیس از جدول‌های مختلف تشکیل شده است.



شکل ۳-۲: نمایش سطر و ستون در یک جدول

مثال: برای ذخیره‌سازی انواع مختلف اطلاعات شما نیازمند ایجاد جدول‌های جداگانه هستید؛ برای مثال، اگر شما یک نرم‌افزار مدیریت مکتب دارید، ممکن است نیاز به ایجاد جدول‌های زیر باشد:

شاگردان: برای ذخیرهٔ لستی از تمام اعضای شاگردان؛
استادان: برای ذخیرهٔ لستی از تمام استادان؛
حاضری: برای دخیرهٔ حاضری تمام شاگردان؛
مضامین: برای ذخیرهٔ معلومات راجع به تمام مضامین؛
نمرات: برای ذخیرهٔ معلومات راجع به نتایج امتحانات شاگردان.

۳.۱.۱ سطر (Row)

یک سطر نشان‌دهندهٔ یک ورودی در جدول است. یک جدول می‌تواند هر تعداد سطر داشته باشد. اگر شما جدول "Students" برای ذخیرهٔ اطلاعات شاگردان داشته باشید، در این جدول یک سطر نشان‌دهندهٔ معلومات راجع به یک شاگرد خواهد بود. برای اضافه‌کردن شاگرد به دیتابیس، باید یک سطر به جدول "Students" اضافه کنید. برای پاک‌کردن یا ویرایش اطلاعات شاگرد هم باید شما یک سطر را از این جدول حذف کنید.

۳.۱.۲ ستون (Column)

به هر کدام از خانه‌های عمودی یک جدول ستون (Column) می‌گویند. هر ستون خصوصیت و مقدار آن را مشخص می‌کند. به عبارتی دیگر، هر ستون در برگیرندهٔ یک صفت و ویژگی برای جدول می‌باشد.

یک سطر مجموعه‌یی از ستون‌ها است. تمام سطرها در یک جدول همان ستون‌ها را خواهند داشت. اگر شما یک جدول با نام "Students" داشته باشید، ستون‌های زیر مورد نیاز هست:

Name: برای ذخیرهٔ نام و نام خانوادگی شاگردان؛
Address: برای ذخیرهٔ آدرس شاگردان؛
DateofBirth: برای ذخیرهٔ تاریخ تولد شاگردان؛

RegistrationDate: برای ذخیرهٔ تاریخ ثبت نام شاگردان؛ و...

اگر شما یک ستون را به جدول اضافه کنید، این ستون به تمام سطرهای موجود آن جدول اضافه خواهد شد. در مثال فوق، تمامی سطرهای جدول "Students" همان ۴ ستون را خواهند داشت. خلاصه این‌که مجموعه‌یی از جدول‌ها دیتابیس را تشکیل می‌دهند و مجموعه‌یی از سطرها جدول را تشکیل می‌دهند و تمام سطرها در یک جدول عین ستون‌ها را دارند.

شکل ۳-۳ نشان‌دهندهٔ واژه‌های مشابه در جدول می‌باشد.

Table	Row	Column
File	Record	Field
Relation	Tuple	Attribute

شکل ۳-۳: واژه‌های مترادف در دیتابیس

در شکل فوق واژه‌هایی که مترادف همدیگر هستند، ذکر گردیده است.

۳.۲ نوعیت اطلاعات (Data Type)

در دیتابیس تعریف‌کردن صحیح ستون‌ها در یک جدول برای مطلوب‌سازی کل آن دیتابیس بسیار حائز اهمیت است. بدین ترتیب باید تنها انواع و اندازه‌هایی را که مورد نیاز است، تعریف کنید. برای نمونه اگر می‌دانید که در یک ستون تنها به 2 حرف نیاز خواهید داشت، نباید یک ستون 10 حرفی تعریف کنید.

هر ستون در یک جدول دیتابیس باید یک نام و یک نوعیت (Data Type) داشته باشد. توسعه‌دهندگان دیتابیس در هنگام ساختن جدول باید تصمیم بگیرند که چه نوعی از اطلاعات در هر ستون جدول ذخیره شود. نوعیت اطلاعات یک راهنما برای دیتابیس محسوب می‌شود تا بفهمد که چه نوعی از اطلاعات در هر ستون می‌تواند جای بگیرد و نیز تشخیص می‌دهد که چه‌گونه دیتابیس با اطلاعات ذخیره‌شده تعامل خواهد داشت.

Field Name	Data Type	
CUSTOMER_ID	AutoNumber	← Autonumber
FORENAME	Text	
SURNAME	Text	← Text
ADDRESS	Text	
TELEPHONE_NUMBER	Text	
DATE_OF_BIRTH	Date/Time	← Date/Time
NUMBER_OF_CHILDREN	Number	← Number
RECEIVE_MAIL	Yes/No	← Logical/ Boolean/ Yes/No

شکل ۳-۴: تعین نوعیت برای جدول

نوعیت اطلاعات در دیتابیس به چند دسته تقسیم می‌شوند که از هر دسته پرکاربردترین آن‌ها را مورد بحث قرار می‌دهیم.

نوع رشته‌یی (Character String)

از این نوع جهت نگهداری عبارات و یا حروف ASCII مورد استفاده قرار می‌گیرد. به عبارت دیگر، برای نگهداری هر حرف، یک بایت از حافظه اشغال می‌شود، لذا نیاز به Collation برای تعین زبان اطلاعات می‌باشد.

CHAR(M): یک رشته با طول ثابت بین ۱ تا ۲۵۵ کاراکتر تعریف می‌کند؛ برای نمونه Char(۵) وقتی کاراکتر کم‌تر از مقدار ذکرشده در این نوع ذخیره شود، مقادیر خالی سمت راست با کاراکتر فاصله یا Space پر می‌شوند. تعریف‌کردن طول ضروری نیست و مقدار پیش‌فرض آن برابر با ۱ است.

VARCHAR(M): یک رشته با طول متغیر بین ۱ و ۲۵۵ کاراکتراست؛ برای مثال، VARCHAR(۲۵) هنگام تعریف‌کردن یک ستون از این نوع باید طول آن قید شود.

BLOB و TEXT: ستون با طول ۶۵۵۳۵ کاراکتر است. منظور از BLOB اشیای باینری بزرگ (Binary Large Objects) است و از آن برای ذخیره‌سازی مقادیر بزرگی از اطلاعات باینری مانند تصاویر یا دیگر انواع فایل استفاده می‌شود. ستون‌هایی که به صورت TEXT تعریف می‌شوند، نیز مقادیر بالای از متن را در خود جای می‌دهند. تفاوت بین این دو آن است که ذخیره‌سازی و مقایسهٔ اطلاعات ذخیره‌سازی‌شده در نوع BLOB به حروف کوچک و بزرگ حساس هستند، در حالی‌که در مورد ستون‌های TEXT چنین مسأله‌یی وجود ندارد؛ بنابراین لازم نیست طولی برای این نوع اطلاعات ذکر شود.

ENUM: نام این نوع اختصار برای عبارت (Enumeration) است که در واقع اصطلاحی برای لیست محسوب می‌شود. زمانی‌که یک نوع از ENUM تعریف می‌کنیم، در واقع لیستی از اقلام ایجاد می‌کنیم که مقادیر آن‌ها باید ذکر شوند یا این که می‌توانند NULL باشند.

۳.۲.۱ نوع عددی (Number Type)

این نوع ستون‌ها برای نگهداری اعداد استفاده می‌گردد و دارای انواع مختلف به شرح زیر است:

TINYINT: یک نوع عدد صحیح بسیار کوچک است که می‌تواند با علامت یا بی علامت باشد. اگر با علامت باشد، محدودهٔ اعداد ۱۲۸- تا ۱۲۷ است و در صورت بی علامت‌بودن اعدادی را از ۰ تا ۲۵۵ در خود جای می‌دهد.

SMALLINT: یک نوع عدد صحیح کوچک که می‌تواند بی علامت یا با علامت باشد. اگر با علامت باشد امکان درج اعدادی در محدودهٔ ۳۲۷۶۸- تا ۳۲۷۶۷ و در صورت بی علامت‌بودن اعدادی از ۰ تا ۶۵۵۳۵ در آن جای می‌گیرند.

INT: یک عدد صحیح با اندازهٔ عادی است که می‌تواند با علامت یا بی علامت باشد. اگر با علامت باشد امکان درج اعداد در محدودهٔ ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷ وجود دارد و اگر بی علامت باشد می‌توان اعدادی را از ۰ تا ۴۲۹۴۹۶۷۲۹۵ در آن جای داد.

BIGINT: این نوع از اعداد صحیح می‌تواند بی علامت یا با علامت باشد. در صورتی‌که با علامت باشد، محدودهٔ مجاز برای آن از ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۸- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷ است و در صورتی‌که بی علامت باشد، می‌توان اعدادی از ۰ تا ۱۸۴۴۶۷۴۴۰۷۳۷۰۹۵۵۱۶۱۵ را در آن تعریف کرد.

DECIMAL(P,S): این نوع ستون برای نگهداری اعداد اعشاری با تعداد اعشار مشخص استفاده می‌گردد. این نوع ستون‌ها بسیار کند بوده و استفاده از آن‌ها توصیه نمی‌گردد. در این نوع Precision به معنای تعداد کل رقم‌های عدد و Scale تعداد ارقام اعشار را مشخص می‌کند. اگر ستون به صورت Deciaml(۶,۲) تعریف شود، حداکثر آن برابر ۹۹۹۹/۹۹ می‌باشد.

FLOAT(M, D): این نوع، یک عدد اعشاری تعریف می‌کند که نمی‌تواند بی علامت باشد. شما می‌توانید طول نمایشی M و تعداد ارقام D را در این نوع تعریف کنید. این پارامترها الزامی نیستند و مقادیر پیش‌فرض آن‌ها به ترتیب ۱۰ و ۲ است که ۲ تعداد ارقام اعشار و ۱۰ تعداد ارقام نمایش‌یافته از عدد (شامل مقدار اعشاری) است.

DOUBLE(M,D): یک عدد اعشاری با دقت مضاعف است که نمی‌تواند بی علامت باشد. در این نوع می‌توان طول M و هم‌چنین تعداد ارقام اعشاری D را تعریف کرد. این پارامترها الزامی نیستند و در صورتی‌که تعین نشوند، مقادیر پیش‌فرض به ترتیب ۱۲۶ و ۴ هستند که ۴ تعداد ارقام اعشار است. دقت اعشاری در این نوع می‌تواند تا ۵۳ رقم باشد. نام دیگر این نوع REAL است.

DECIMAL(M,D): یک عدد اعشاری غیر فشرده است که نمی‌تواند بی علامت باشد. در اعداد اعشاری غیر فشرده، هر رقم اعشار معادل یک بایت است. تعریف‌کردن طول نمایشی M و تعداد اعشار D الزامی است. NUMERIC یک مترادف برای نوع عددی DECIMAL محسوب می‌شود.

۳.۲.۲ نوع تاریخ و زمان (Date and Time)

DATE: این نوع شامل تاریخی با شکل YYYY-MM-DD است که بین ۱۰۰۰-۰۱-۰۱ تا ۹۹۹۹-۱۲-۳۱ قرار دارد؛ برای نمونه، تاریخ سی‌ام دسامبر ۱۹۷۳ را می‌توان به صورت ۱۹۷۳-۱۲-۳۰ ذخیره کرد.

DATETIME: این نوع شامل ترکیبی از تاریخ و زمان را به شکل YYYY-MM-DD HH:MM:SS ذخیره می‌کند. تاریخ‌ها باید بین ۱۰۰۰-۰۱-۰۱ ۰۰:۰۰:۰۰ تا ۹۹۹۹-۱۲-۳۱ ۲۳:۵۹:۵۹ باشند؛ برای مثال، ساعت ۳:۳۰ بعد از ظهر تاریخ سی‌ام دسامبر ۱۹۷۳ به صورت ۱۹۷۳-۱۲-۳۰ ۱۵:۳۰:۰۰ خواهد بود.

TIME: زمان را در قالب HH:MM:SS ذخیره می‌کند.

YEAR(M): یک سال را در قالب ۲ رقمی یا ۴ رقمی ذخیره می‌کند. اگر طول به صورت ۲ تعین شده باشد، در این صورت می‌توان سال‌های بین ۱۹۷۰ تا ۲۰۶۹ (۷۰ تا ۶۹) را ذخیره کرد و اگر طول سال به صورت ۴ رقمی تعین شده باشد در این صورت می‌تواند سال‌های بین ۱۹۰۱ تا ۲۱۵۵ را در خود جای دهد.

۳.۳ کلید و انواع آن (Keys)

کلید (Key) عبارت از یک یا اضافه از یک ستون در یک جدول است که معمولاً برای شناسایی یک یا چندین سطر استفاده می‌شود. این نوع ستون‌ها می‌تواند اطلاعات تکراری و غیر تکراری داشته باشد.

در دیتابیس بعضی صفات با ویژگی‌های خاصی شناخته می‌شوند و هر کدام با توجه به خاصیتی که دارند، کلید خاصی نام می‌گیرند که انواع کلید شامل ابر کلید، کلید کاندید، کلید اصلی، کلید فرعی و کلید خارجی و غیره می‌باشد و بارزترین آن‌ها را مورد بحث قرار می‌دهیم.

۳.۳.۱ ابر کلید (Super–Key S.K)

ابر کلید ترکیبی از صفت‌هایی است که خاصیت کلید را دارند و سایر خاصیت‌ها یا ستون‌های جدول به آن وابستگی تابعی داشته باشد. این نوع کلید می‌تواند دارای زیرمجموعه‌های مختلفی از کلیدها باشد. این ترکیب از صفات باید باهم یک سطر از مجموعه‌یی سطرها را به صورت یکتا مشخص کند و یا می‌توان گفت ابر کلید عبارت است از هر ترکیبی از اسامی صفات جدولی که در هیچ دو سطر مقدار یک‌سان نداشته باشد؛

برای مثال شمارۀ تذکره به تنهایی برای مشخص‌کردن یکتابودن یک فرد کافی است. بنابراین، شمارۀ تذکره می‌تواند به عنوان یک ابر کلید مطرح باشد. حال هر ویژگی و صفت دیگری همراه با شمارۀ تذکره نیز یک ابر کلید است؛ برای مثال، نام فرد و شمارۀ تذکره با همدیگر یک ابر کلید را تشکیل می‌دهند. در حالت کلی اگر K یک ابر کلید باشد، آن‌گاه هر مجموعه‌یی که K زیر مجموعۀ آن باشد نیز یک ابر کلید است.

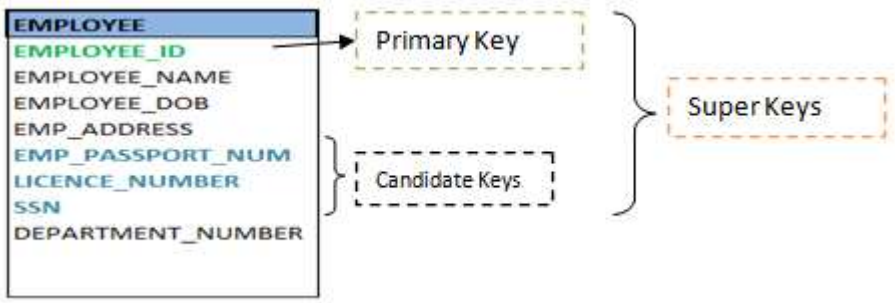
۳.۳.۲ کلید کاندید (Candidate–Key C.K)

کلید کاندید یک شناسۀ منحصربه‌فرد برای سطری از یک رابطه می‌باشد. کلید کاندید ممکن است ساده (یک ستون) یا مرکب (دو یا چند ستون) باشد. هر رابطه باید حداقل یک ستون کاندید داشته باشد، اما بعضی رابطه‌ها می‌توانند بیش از یک کلید کاندید نیز داشته باشند. چنان‌چه تنها یک کلید موجود باشد، آن‌گاه به طور خودکار کلید اصلی نیز می‌شود. اگر چند کلید کاندید وجود داشته باشد، طراح باید یک یا ترکیبی از آن‌ها را به عنوان کلید اصلی معرفی کند. کلید کاندید مجموعه‌یی از یک یا چند خاصیت است که سایر خاصیت‌های جدول به آن وابستگی تابعی کامل دارند.

۳.۳.۳ کلید اصلی (Primary–Key P.K)

هر جدول باید دارای یک کلید اصلی باشد. کلید اصلی یک مقدار خاص و غیر خالی (not-null) می‌باشد که یک سطر را در دیتابیس شناسایی می‌کند و هم‌چنان برای ارتباط بین جدول‌ها استفاد می‌شود.

هنگامی‌که کلیدهای کاندید یک مجموعه رابطه مشخص شدند، آن‌گاه طراح دیتابیس به سلیقهٔ خود، یکی از آن‌ها را به عنوان کلید اصلی بر می‌گزیند. البته در انتخاب این کلید دو نکته بسیار مهم است: اول هرچه کوتاه‌تربودن طول کلید اصلی از نظر طول رشتهٔ بایتی و دوم اهمیت کلید اصلی نسبت به سایر کلیدهای کاندید در پاسخ‌گویی به پرس‌وجوهای استفاده‌کننده؛ برای مثال، در چند مثال قبل بهتر است که شمارهٔ تذکره به عنوان کلید اصلی انتخاب شود تا مجموعه‌یی (آدرس و نام)؛ چون اولاً طول یک ویژگی کم‌تر از طول دو ویژگی است و دوم این‌که پرس‌وجوهای استفاده‌کننده‌ها از طریق شمارهٔ تذکره انجام می‌شود، نه از طریق آدرس و نام.



شکل ۳-۵: نمایش کلید کاندید، کلید اصلی و ابر کلید

۳.۳.۴ کلید فرعی (Alternate–Key A.K)

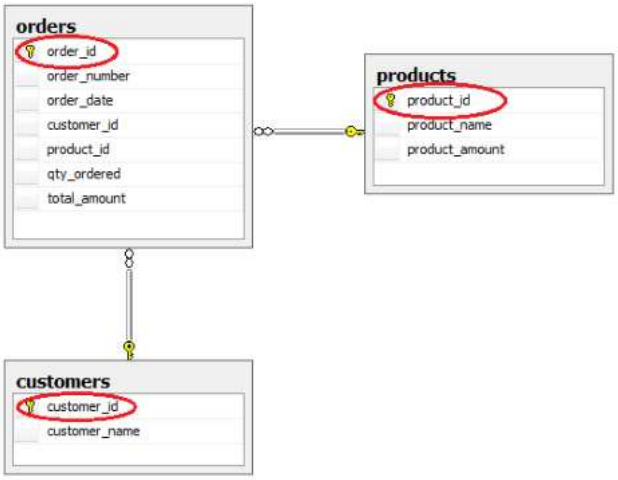
هر کلید کاندیدی که به عنوان کلید اصلی انتخاب نشده است، یک کلید فرعی است و می‌تواند در مواقع لزوم که به ندرت اتفاق می‌افتد، به جای کلید اصلی مورد استفاده قرار گیرد.

Roll_No	Name	Branch	City
01	Deepak	Computers	Bhiwani
02	Mukesh	Electronics	Rohtak
03	Teena	Mechanical	Bhiwani
04	Deepti	Chemical	Rohtak
05	Monika	Civil	Delhi

شکل ۳-۶: نمایش کلید فرعی در جدول

۳.۳.۵ کلید جانشین (Surrogate Key S.K)

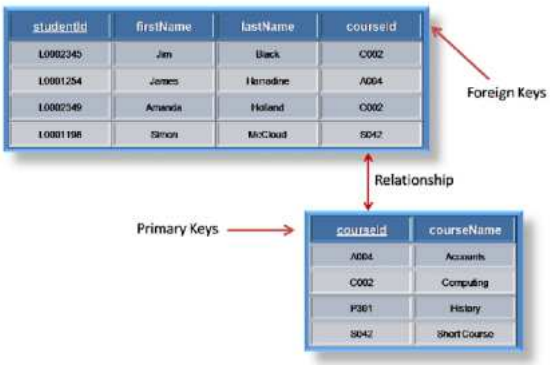
کلید جانشین یا (Surrogate Key) یک ستون است که مقادیرش به صورت خودکار توسط موتور دیتابیس (DB Engine) تولید و ارائه می‌شود. کلید جانشین زمانی کاربرد دارد که هیچ راه موجه و مشخص برای استفاده از مقادیر یک ستون (ایجادشده در جدول) به عنوان کلید اصلی وجود نداشته باشد. مقادیر ستون کلید جانشین معمولاً اعداد صحیح (از نوع integer) هستند که معنا و مفهوم خاصی برای استفاده‌کننده ندارند و صرف به خاطر استفاده در دیتابیس معرفی می‌گردد.



شکل ۳-۷: نمایش کلید جانشین

۳.۳.۶ کلید خارجی (Foreign–Key F.K)

اگر صفت خاصهٔ A از رابطهٔ 1Rیک کلید اصلی باشد و همین صفت A در رابطهٔ 2R نیز وجود داشته باشد، صفت A در رابطهٔ 2R یک کلید خارجی است که می‌تواند باعث ارجاع دو رابطهٔ 1R و 2R نسبت به هم شود و در واقع این دو رابطه را به هم پیوند دهد. کلید خارجی تنها کلیدی است که می‌تواند مقدار Null را اختیار کند.



شکل ۳-۸: نمایش کلید فرعی

توجه کنید که تنها راه ارتباط بین دو رابطه کلید خارجی نیست، بلکه وجود هر صفت با ویژگی (عین Datatype) مشترک بین دو رابطه می‌تواند باعث ایجاد ارتباط بین آن روابط گردد. در روابط یک به چند، در یک طرف چند کلید خارجی وجود دارد و در طرف دیگر یک کلید اصلی و یا فرعی وجود دارد.

هم‌چنین تنها راه مشخص‌کردن سطر در جدول وجود کلید اصلی در آن رابطه است. البته می‌توان به کمک هر صفت خاص اعم از کلید و یا غیر کلید به یک سطر دست‌رسی داشت، ولی نتیجهٔ دست‌یابی به آن حد اکثر بیش از یک پرس‌وجو خواهد بود. این در حالی است که با کلید اصلی آن تعداد پرس‌وجو به حد اقل خود، یعنی یک پرس‌وجو می‌رسد.

در مثال ذیل استفاده از کلید اصلی و کلید خارجی واضح می‌گردد.

جدول ۳-۱: جدول "Persons"

PersonID	FirstName	LastName	Age
1	Ola	Hansen	**30**
2	Tove	Svendson	**23**
3	Kari	Pettersen	**20**

جدول۳-۲: جدول "Orders"

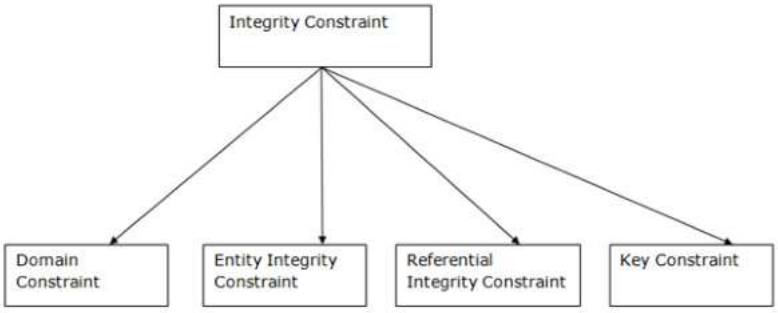
OrderID	OrderNumber	PersonID
1	77895	**3**
2	44678	**3**
3	22456	**2**
4	24562	**1**

دو جدول فوق از طریق ستون "PersonID" که در هردو جدول وجود دارد، باهم ارتباط دارند که این ستون در جدول "Persons" به عنوان کلید اصلی (Primary Key) این جدول و در جدول "Orders" به عنوان کلید خارجی (Foreign Key) معرفی شده است.

کلید خارجی در واقع ارتباط بین جدول‌ها را حفظ می‌کند و از اقداماتی که ارتباط بین جدول‌ها را از بین می‌برد، جلوگیری می‌کند. قابل یادآوری است که اطلاعات در ستون کلید خارجی باید ابتداء در ستون کلید اصلی جدول اصلی یا "Persons" وجود داشته باشد.

۳.۴ محدودیت‌ها (Constraints)

از محدودیت یا (Constraints) برای تعیین قوانین بالای اطلاعات در یک جدول استفاده می‌شود. اگر تناقضی بین محدودیت‌های تعیین‌شده و عملی که قرار است انجام شود وجود داشته باشد، محدودیت‌ها جلو انجام عمل فوق را گرفته و اجراء نمی‌گردد. برای درک بیشتر انواع محدودیت ها شکل ۳-۹ را مشاهده کنید.



شکل ۳-۹: انواع محدودیت‌ها

محدودیت‌ها می‌توانند هنگام ایجاد یک جدول و یا بعد از این‌که جدول ایجاد شد، مشخص شوند. محدودیت‌ها برای موتور دیتابیس داخلی هستند که گزینهٔ موثرتری برای چک‌کردن درستی اطلاعات می‌باشند. محدودیت‌های (Primary-Key، DEFAULT، UNIQUE، CHECK و Foreign-Key) باید برای محافظت از اطلاعات در مقابل ورودی‌های نامعتبر تعریف شود.

محدودیت‌ها شامل انواع زیر می‌باشند:

۱. Entity Integrity Constraints که شامل کلید اصلی (Primary-Key) و کلید یکتا (Unique key) می‌باشد.
۲. Referntial Integrity Constraints که شامل کلید خارجی (Foreign-Key) می‌باشد.
۳. Domain Integrity Constraints که شامل (Not-Null، Default و Check) می‌باشد.
۴. Key Constraint شامل کلیدهایی است که در فوق توضیح داده شد.

۳.۴.۱ Entity Integrity Constraints

کلیدهای Unique و Primary Key شباهت‌های بسیار دارند. برای بسیاری افراد این سوال به وجود می‌آید که چه تفاوت‌های بین این دو وجود دارد که در ادامه به آن‌ها اشاره خواهد شد.

ویژگی‌ها و شباهت‌ها:

هردوی این کلیدها برای به اجراءدرآوردن جامعیت اطلاعات (Enforcing Data Integrity) به کار گرفته می‌شوند. در حقیقت هر دو، نوعی محدودیت (Constraint) می‌باشند. این محدودیت مکانیزم استندردی را برای دیتابیس آماده می‌کنند تا جامعیت اطلاعات توسط آن‌ها اعمال شود.

هردو برای جلوگیری از واردشدن اطلاعات تکراری (Duplicate) مورد استفاده قرار می‌گیرند، یعنی هردو برای اعمال (Uniqueness) ستون یا ستون‌های (در صورت ترکیبی‌بودن) به کار گرفته می‌شوند.

زمانی‌که یک قید کلید اصلی و یا یکتا ایجاد می‌کنیم، یک [۳]ایندکس منحصربه‌فرد (Unique Index) توسط Database Engine ساخته می‌شود. از این ایندکس برای سرعت‌بخشیدن به Queryهایی که بر اساس این ستون‌ها هستند، مورد استفاده قرار می‌گیرد.

اگر مقادیر تکراری در جدول درج شده باشند (در ستون یا ترکیب از ستون‌های مورد نظر) نمی‌توانیم این محدودیت‌ها را ایجاد کنیم. هردوی این کلیدها می‌توانند از ترکیب چند ستون ایجاد شوند.

تفاوت‌ها:

کلید اصلی اجازه نمی‌دهد که مقدار یا قیمت NULL جزئی از قیمت‌های کلید باشد؛ یعنی ما اجازه نداریم که ستون‌هایی را که در PK شرکت دارند، Null در نظر بگیریم، ولی برعکس آن کلید یکتا به ما این امکان را می‌دهد که مقادیر یا قیمت‌های NULL را برای ستون یا ستون‌های (در صورتی‌که محدودیت Composite/Compound باشد) که در UK شرکت دارند، درج کنیم.

در یک جدول می‌توانیم چندین محدودیت Unique ایجاد کنیم (که آن وابسته به محدودیت تعداد شاخصی است که در یک جدول می‌توانیم ایجاد کنیم)، ولی تنها یک کلید Primary-Key جدول می‌تواند داشته باشد.

۳.۴.۲ Referential Integrity Constraints

این دسته شامل کلید خارجی می‌گردد که در بخش کلیدها و انواع آن مفصل مورد بحث قرار گرفت. مقادیر کلید اصلی و کلید یکتا از جدولی Parent که به جدول Child ذکر می‌گردد، به نام Parent-Key یا Referenced-Key یاد می‌شود که کلید خارجی ست فرعی کلید اصلی می‌باشد؛ بدین ملحوط درج دیتا باید اول در جدول parent صورت گیرد

۳.۴.۳ Domain Integrity Constraints

NOT-NULL: امکان Null بودن این ستون وجود ندارد و حتماً باید مقدار برای آن تعریف گردد.

DEFAULT: این محدودیت برای مشخص‌کردن مقدار پیش‌فرض به یک ستون استفاده می‌شود.

CHECK: این محدودیت برای مشخص‌کردن محدوده‌یی از مقدارها به یک ستون استفاده می‌شود.

---

3 Index عبارت از ساختمانی است که به یک یا چندین ستون جدول انتخاب گردیده و سبب جست‌وجوی سریع در جدول دیتابیس می گردد.

خلاصهٔ فصل سوم

جدول مخزن برای ذخیرهٔ اطلاعات می‌باشد که یک دیتابیس می‌تواند متشکل از یک یا چندین جدول باشد. یک جدول می‌تواند متشکل از ستون‌های مشخص و هر اندازه از سطرها باشد.

برای تمام ستون‌ها در دیتابیس نظر به نوع کارکرد آن می‌توانید نوعیت (Data Type) مورد نظر خود را تعریف کنید. مقدارهایی که در هر ستون (Column) ذخیره می‌شود، باید با نوع انتخابی ستون مطابقت داشته باشد، که مهم‌ترین و پر استفاده‌ترین آن‌ها مفصل مورد بحث قرار گرفت.

برای سرعت‌بخشیدن جست‌وجو در جدول و مطلوب‌ساختن آن باید ستون‌های کلیدی را مشخص بسازیم که در فوق انواع کلید با تفاوت و کاربرد آن ذکر گردیده است.

از محدودیت یا (Constraints) برای تعیین قوانین بالای اطلاعات در یک جدول استفاده می‌شود. اگر تناقضی بین محدودیت‌های تعیین‌شده و عملی که قرار است انجام شود وجود داشته باشد، محدودیت‌ها جلو انجام عمل فوق را گرفته و اجراء نمی‌گردد.

سوالات و فعالیت های فصل سوم

۱. جدول چیست؟ واضح سازید.

۲. جدول متشکل از کدام بخش‌ها می‌باشد؟ بیان دارید.

۳. نوعیت اطلاعات در دیتابیس را واضح ساخته و چند مورد آن را بیان دارید.

۴. کلید چیست و انواع آن را نام ببرید.

۵. چگونه یک کلید از بین ستون‌های موجود انتخاب نماییم؟

۶. چرا محدودیت‌ها بالای ستون‌ها وضع می‌گردد؟

۷. انواع محدویت‌ها را تشریح نمایید.

۸. در مورد قاعدهٔ RIC (Referential Integrity Constraints) معلومات ارائه نمایید. (جست‌وجو در انترنت)

فعالیت‌ها

۱. جدول مورد نظر خود را انتخاب نموده، برای ستون‌های آن نوعیت (Data Type) تعیین نمایند. (گروپی)

۲. یکی از ستون‌های موجود در جدول مورد نظر خود را کلید اصلی (Primary-Key) و کلید یکتا (Unique-Key) انتخاب نموده، دلیل انتخاب آن را بیان دارید. (انفرادی)

فصل چهارم

معرفی ارتباطات (Relationship Concepts)

هدف کلی: آشنایی محصلان با ارتباطات بین جدول‌ها.

اهداف آموزشی: در پایان این فصل، محصلان قادر خواهند بود تا:

۱. در مورد Relationships معلومات حاصل نمایند.
۲. انواع Relationship را نام ببرند.
۳. در بین جدول‌ها Relationship ایجاد نمایند.

در این فصل ارتباطات بین جدول‌ها را به معرفی گرفته، انواع ارتباطات را از لحاظ اشتراک سطرها در یک رابطه و تعداد جدول‌های موجود در یک رابطه را مورد بحث قرار می‌دهیم و نیز این‌که یک جدول تا کدام حد می‌تواند در یک ارتباط سهم داشته باشد را بررسی خواهیم کرد.

۴.۱ ارتباطات (Relationships)

جدول‌های هر محیط عملیاتی باهم ارتباطاتی دارند که همین ارتباطات، وابستگی بین چند جدول را نشان می‌دهد. یک ارتباط (Relationship) یک وابستگی معنادار بین دو یا چند نوع جدول مختلف است.

تعداد جدول‌ها در ارتباط:

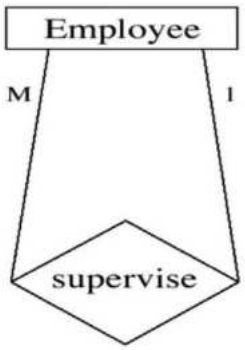
تعداد جدول‌هایی که در یک ارتباط شرکت می‌کنند، درجهٔ (Degree) ارتباط نامیده می‌شود.

یک ارتباط بازگشتی (Recursive Relationship) وقتی اتفاق می‌افتد که یک جدول به خودش مربوط می‌شود. اگر دو جدول با هم مربوط شوند، ارتباط شان درجه دو است و ارتباط درجه دو نامیده می‌شود. ارتباطات درجه دو معمول‌ترین نوع در دنیای واقعی هستند. اگر سه نوع جدول درگیر باشند، ارتباط از درجه سه است و ارتباط سه‌تایی (Ternary Relationship) نامیده می‌شود. ارتباطات سه‌تایی اکثراً به دو یا چند ارتباط دوتایی تجزیه می‌شود.

۴.۱.۱ ارتباط بازگشتی (Recursive Relationship)

یک جدول می‌تواند رابطه را بین عناصر خود جدول به شکل داخلی ایجاد نماید که چنین رابطه به نام ارتباط بازگشتی (Recursive Relationship) یاد می‌شود.

این نوع رابطه‌ها مانند ارتباط درجه دو می‌توانند یک به یک (1:1)، یک به چند (M:1) و چند به چند (N:M) باشد. شکل ۴-۱ نشان‌دهندهٔ یک ارتباط بازگشتی می‌باشد.



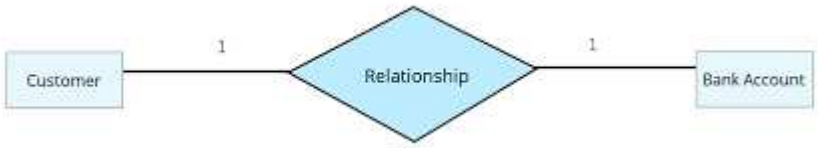
شکل ۴-۱: نمایش ار تباط بازگشتی

نظر به شکل انتخاب‌شده یک Employee می‌تواند Employeeهای دیگر را supervise کند؛ یعنی هم employee است و هم‌زمان دیگران را supervise می‌کند

۴.۱.۲ ارتباط درجه دو (Binary Relationship)

رابطه‌های درجه دو (Binary Relationship) به رابطه‌هایی گفته می‌شود که تعداد جدول‌های اشتراک‌کننده در این نوع رابطه دو باشد. این نوع رابطه مانند ارتباط بازگشتی به سه نوع است که عبارت از یک به یک (1:1)، یک به چند (M:1) و چند به چند (N:M) می‌باشد.

این نوع ارتباطات (ارتباط درجه دو) از معمول‌ترین نوع آن در دنیای واقعی است و در اکثر موارد کاربرد دارد. شکل ۴-۲نشان‌دهندهٔ یک ارتباط درجه دو از نوع یک به یک می‌باشد.

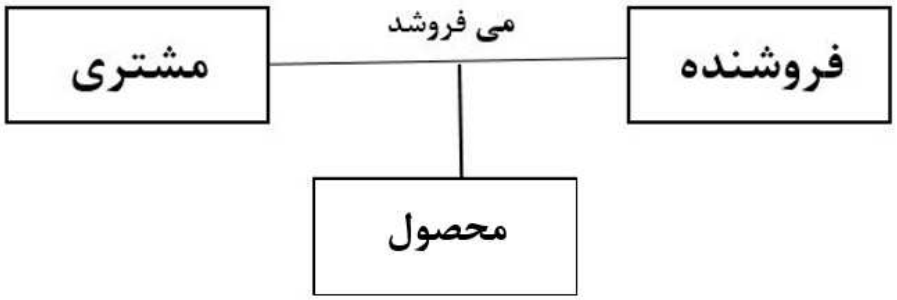


شکل ۴-۲: نمایش ارتباط درجه دو

۴.۱.۳ ارتباط درجه سه (Ternary Relationship)

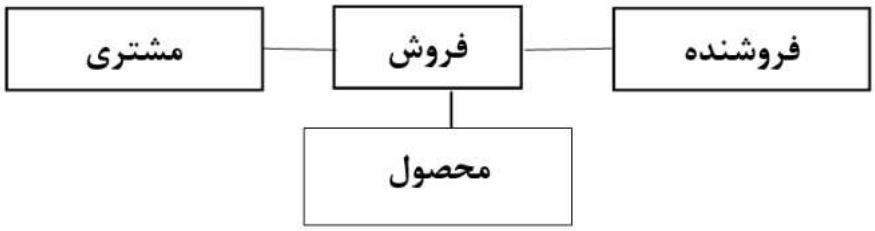
اگر سه جدول با هم در ارتباط باشند، ارتباط درجه سه نامیده می‌شود. ارتباطات درجه سه (Ternary) اکثراً به دو یا چند ارتباط درجه دو تجزیه می‌شود.

مثال: یک فروشنده محصولی را به یک مشتری می‌فروشد؛ این یک ارتباط از درجه سه می‌باشد. . شکل ۴-۳ نشان‌دهندهٔ یک ارتباط درجه سه است.



شکل ۴-۳: نمایش ارتباط درجه سه

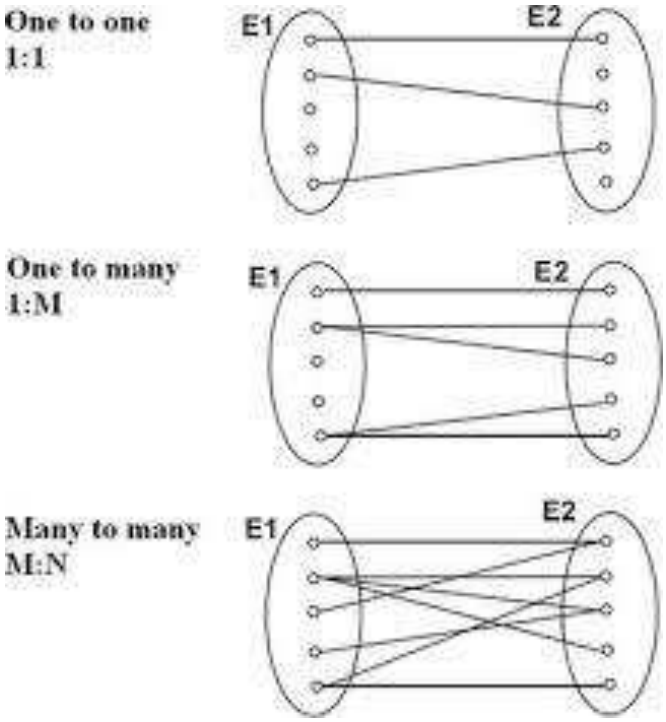
این ارتباط درجه سه را می‌توان به چند ارتباط درجه دو تجزیه کرد. جدول جدیدی با نام فروش جای‌گزین ارتباط "می‌فروشد" می‌شود. حالا فروشنده می‌تواند به مشتری پیوند بخورد.



شکل ۴-۴: تبدیل ارتباط درجه سه به دو ارتباط درجه دو

۴.۲ حد (Cardinality)

حد (Cardinality) در یک ارتباط تعداد حد اقل و حد اکثر سطرهای یک جدول را که در یک ارتباط مشارکت می‌کنند، مشخص می‌کند. به بیان دیگر، تناظر بین عناصر مجموعهٔ سطرهای یک جدول با عناصر مجموعهٔ سطرهای جدول دیگر در یک ارتباط را بیان می‌کند.



شکل ۴-۵: حد اشتراک

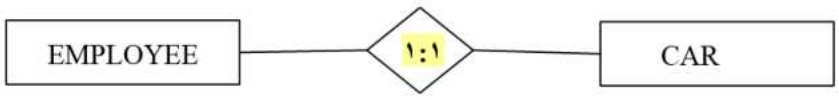
کاردینالیتی یا ارتباط از نظر نوع اتصال حالت‌های زیر را ممکن است داشته باشد:

ارتباط یک به یک (۱:۱)؛
ارتباط یک به چند (۱:M)؛
ارتباط چند به چند (M:N).

۴.۲.۱ ارتباط یک به یک (۱:۱)

ساده‌ترین شکل یک ارتباط درجه دو (Binary Relationship) عبارت از (1:1) است که در آن Entity-Instance از یک نوع به حد اعظمی "یک" با Entity-Instance۴ از نوع دیگر ارتباط برقرار نموده می‌تواند.

مثال آن در ذیل نشان داده شده است. به اساس این دیاگرام، یک Employee می‌تواند یک موتر یا (CAR) داشته باشد و یک موتر تنها به یک Employee ارتباط داشته می‌تواند.



شکل ۴-۶: نمایش ارتباط یک به یک

وقتی خواسته شود تا یک Relationship یک به یک در مودل رابطه‌یی نشان داده شود، جدول‌ها به صورت ساده تبدیل شده و کاپی کلید اصلی یک جدول در جدول دیگر به شکل کلید خارجی (Foreign-Key) تنظیم می‌شود.

در مثال فوق کلید جدول CAR در جدول EMPLOYEE به شکل (Foreign-Key) اضافه می‌شود و یا کلید جدول EMPLOYEE در جدول CAR به شکل (Foreign-Key) تنظیم و ارتباط بر قرار می‌شود. هردو حالت ممکن بوده و درست اند. به یک نکته باید دقت شود که همیشه کاپی (Primary-Key) جدول والد یا Parent در جدول Child به شکل (Foreign-Key) جابه‌جا شده و مطابق قاعدهٔ RIC دیتابیس عناصر جدول Child ست فرعی جدول Parent می‌باشند.

۴.۲.۲ ارتباط یک به چند (M:۱)

نوع دوم ارتباط درجه دو (Binary Relationship) عبارت از (M:1) یا یک به چندین می‌باشد. بدین مفهوم که یک Entity-Instance از یک نوع به چندین Entity-Instance از نوع دیگر ارتباط داشته می‌تواند. یا یک عنصر از جدول اول به چندین عنصر از جدول دوم مربوط شده می‌تواند. شکل ذیل یک ارتباط از نوع یک به چندین (M:1) را نشان می‌دهد. در این شکل اطلاعات در مورد پروفیسور و دانشجو نشان داده شده است.

شکل ۴-۷: نمایش ارتباط یک به چند

---

4 Entity instance عبارت از مشخصه های یک Entity میباشد

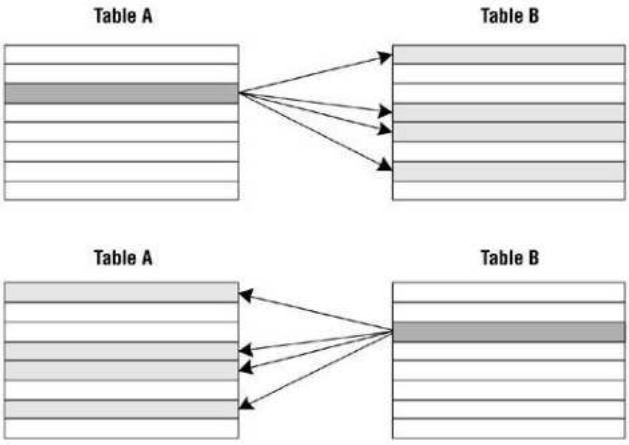
طوری‌که دیده می‌شود، یک پروفیسور امکان دارد هیچ شاگرد نداشته باشد و یا هم می‌تواند چندین شاگرد داشته باشد. برعکس یک شاگرد تنها و تنها می‌تواند یک پروفیسور را به حیث راهنما داشته باشد. شرایط ذکرشده توسط سمبول‌های "بیضوی" و "خط عمودی"، که بالای خط ارتباط ترسیم شده اند، نشان داده می‌شود.

نکتهٔ اساسی که در رابطه‌های (N:1) و (1:1) باید مراعات شود این است که در هر صورت باید کاپی کلید جدول Parent در جدول Child به حیث (Foreign-Key) اضافه شود. در رابطه‌های (N:1) مسأله زیاد پیچیده نیست، فقط کاپی کلید سمت یک به سمت چندین اضافه شده و پروسس می شود. اما در رابطه‌های (1:1) تعیین جدول‌های Parent و Child یک اندازه مشکل است و به دقت و معلومات بیش‌تر ضرورت دارد. زمانی‌که رابطهٔ یک به یک در دیتابیس طرح می‌شود، در این قسمت باید معلومات اضافه و کامل از استفاده‌کنندگان دیتابیس دریافت شده و بعد تصمیم گرفته شود.

هر جدولی که در رابطهٔ یک به یک قوی‌تر است و اول اطلاعات به آن داخل می‌گردد، همان جدول باید Parent باشد و کاپی کلید آن به جدول مقابل در رابطه اضافه شده و به جدول (Parent) مرجع (Reference) داده شود.

۴.۲.۳ ارتباط چند به چند (M:N)

نوع دیگر ارتباط درجه دو (Binary Relationship) عبارت از (N:M) است و آن طوری است که یک Entity-Instance از نوع اول به چندین Entity-Instance از نوع دوم ارتباط گرفته می‌تواند و به همین ترتیب یک Entity-Instance از نوع دوم به چندین Entity-Instance از نوع اول ارتباط گرفته می‌تواند.



شکل ۴-۸: ارتباط نمونه‌های دو جدول

در شکل ذیل یک رابطهٔ چندین به چندین را بین STUDENT و CLASS نشان می‌دهد. یک شاگرد می‌تواند چندین صنف بگیرد و یک صنف می‌تواند چندین شاگرد داشته باشد. اشتراک هر کدام از Entity-Instanceها در رابطه اختیاری است؛ یعنی کاردینالتی اصغری صفر است. یک صنف می‌تواند هیچ شاگرد

نداشته باشد و یا یک صنف می‌تواند چندین شاگرد داشته باشد و یک شاگرد می‌تواند هیچ صنف نگیرد و یا یک شاگرد می‌تواند چندین صنف بگیرد.

شکل ۴-۹: نمایش ارتباط چند به چند

در رابطه‌های چندین به چندین، مانند رابطه‌های یک به یک و یک به چندین، به صورت مستقیم نمایش داده نمی‌شوند بلکه باید یک جدول یا Relation سومی ایجاد شود که نمایان‌گر همین Relationship باشد. جدول تقاطع یا Intersection دارای یک سطر برای هر خط بین جدول‌ها STUDENT و CLASS می‌باشد. در حقیقت یک رابطهٔ (N:M) به دو رابطهٔ (N:1) شکستانده شده و کلیدهای اصلی این جدول‌ها به حیث کلید خارجی به جدول سومی (تقاطع) درج می‌گردد.

۴.۲.۴ مشارکت اجباری و اختیاری

یک جدول در یک ارتباط می‌تواند به صورت اجباری (Mandatory) یا اختیاری (Optional) شرکت کند. اگر یک سطر از یک جدول همیشه در یک رابطه مشارکت کند، مشارکت اجباری است و اگر وجود یک سطر جدول در ارتباط الزامی نباشد، مشارکت اختیاری است؛ یعنی در مشارکت اجباری برای هر عنصر از Entity A در Entity B عنصر موجود باشد و هم‌چنان برای هر عنصر از Entity B در Entity A عنصر موجود باشد؛ مثلاً رابطه بین مادر و طفل از جمله مشارکت اجباری است، یعنی موجودبودن طفل به موجودیت مادر مربوط می‌شود و هم‌چنان بودن مادر در موجودیت طفل واقع می‌شود.

اما در مشارکت اختیاری برای هر عنصر از Entity A ضرورت نیست که حتماً یک عنصر در Entity B موجود باشد؛ یعنی حد مشارکت می‌تواند که صفر (0) باشد.

خلاصهٔ فصل چهارم

جدول‌های هر محیط عملیاتی باهم ارتباطاتی دارند. ارتباط وابستگی بین چند جدول را نشان می‌دهد. یک ارتباط (Relationship) یک وابستگی معنادار بین دو یا چند نوع جدول مختلف است. ارتباطات را از دو لحاظ می‌توانیم طبقه‌بندی کنیم: اول از لحاظ تعداد جدول‌های شریک در ارتباط و دوم از لحاظ کاردینالیتی.

تعداد جدول‌هایی که در یک ارتباط شرکت می‌کنند، درجهٔ (Degree) ارتباط نامیده می‌شود. یک ارتباط بازگشتی (Recursive Relationship) وقتی اتفاق می‌افتد که یک جدول به خودش مربوط می‌شود. اگر دو نوع جدول به هم مربوط شوند، ارتباط درجه دو است و اگر سه نوع جدول درگیر باشند، ارتباط درجه سه است. کاردینالیتی (Cardinality) در یک ارتباط تعداد حد اقل و حد اکثر سطرهای یک جدول را که در یک ارتباط مشارکت می‌کنند، مشخص می‌کند که عبارت از (1:1)، (N:1) و (M:N) است.

یک جدول در یک ارتباط می‌تواند به صورت اجباری (Mandatory) یا اختیاری (Optional) شرکت کند.

سوالات و فعالیت های فصل چهارم

۱. ارتباط چیست؟ واضح سازید.
۲. انواع رابطه را از لحاظ درجهٔ رابطه توضیح دهید.
۳. انواع رابطه را از لحاظ حد توضیح دهید.
۴. تفاوت رابطهٔ چند به چند (N:M) را با یک به یک (1:M) و یک به چند (1:1) توضیح دهید.
۵. مشارکت اختیاری و اجباری چیست؟

فعالیت‌ها

۱. یک دیتابیس دل‌خواه‌تان را مد نظر گرفته، بین جدول‌ها ارتباطات را تشخیص دهید. (گروپی)
۲. یک مثال برای ارتباط بازگشتی در دنیای واقعی را مشخص کرده، توضیح دهید. (انفرادی)

فصل پنجم

نارمل‌سازی (Normalization)

هدف کلی: آشنایی محصلان با وابستگی‌ها و نارمل‌سازی.

اهداف آموزشی: در پایان این فصل، محصلان قادر خواهند بود تا:

۱. Functional Dependency را تعریف کنند.
۲. Transitive Dependency را تشریح کنند.
۳. انواع Normalization را توضیح دهند.
۴. Normalization را بالای جدول‌ها تطبیق نمایند.

در این فصل انواع وابستگی‌ها را مورد بحث قرار داده و سپس نارمل‌سازی و اشکال آن را به معرفی می‌گیریم. دانستن وابستگی‌ها بین ستون‌های یک جدول یک امر ضروری قبل از نارمل‌سازی می‌باشد. وابستگی‌ها دارای انواع مختلف اند که در این فصل فقط دو نوع مهم آن را که عبارت از وابستگی تابعی و وابستگی با واسطه می‌باشند، توضیح داده شده تا خواننده به طور درست یک جدول را نارمل‌سازی نماید.

۵.۱ وابستگی تابعی (Functional Dependency)

در مودل رابطه‌یی دیتابیس، وابستگی تابعی (Functional Dependency) رابطه بین دو مجموعه از ستون‌ها را در یک جدول مشخص می‌کند. به عبارت دیگر، وابستگی تابعی عبارت از رابطه‌یی است که یک ستون توسط ستون دیگر به صورت یکتا مشخص شود.

برای دانستن مفهوم وابستگی تابعی (Functional Dependency)، مثال‌های سادۀ الجبری زیر مرور شوند:

فرضاً در یک فروشگاه، قطی‌های کلچه فی دانه مبلغ 10 افغانی قیمت دارند، قیمت اضافه از یک قطی به طور ساده چنین معلوم می‌شود:

CookieCost = NumberOfBoxes * 10

اگر به سطر بالا توجه شود، رابطه بین ستون Cookie Cost و ستون Number Of Boxes عبارت از مربوط‌بودن Cookie Cost به Number Of Boxes می‌باشد؛ یعنی عنصر اول تابع عنصر دوم است. در اساسات دیتابیس این جمله به شکل ذیل ارائه می‌شود:

Number of Boxes ⟶ Cookie Cost

جملۀ بالا به این‌گونه نیز گفته شده می‌تواند که عنصر Number Of Boxes مشخص‌کنندۀ (Determinant) عنصر Cookie Cost است. متحولی که در طرف چپ واقع شده است، یعنی عنصر Number Of Boxes به نام Determinant عنصر طرف راست (CookieCost) یاد می‌شود.

در مثال دیگر، فکر شود که یک خریطه با اشیای مختلف با رنگ‌های سرخ، آبی و زرد با شخص اول موجود است. هم‌چنان فرض شود که اشیای با رنگ سرخ دارای وزن 5 کیلوگرام، اشیای با رنگ آبی دارای وزن 3 کیلوگرام و اشیای با رنگ زرد دارای وزن 7 کیلوگرام اند. شخص دوم یکی از این اشیاء را بدون این‌که شخص اول ببیند، برداشته و صرف رنگ آن را می‌گوید. با دانستن نوعیت رنگ شخص اول به صورت فوری وزن شئ متذکره را تشخیص می‌دهد. بناءً رنگ شئ مشخص‌کنندۀ (Determinant) وزن شئ است. به عبارت دیگر، وزن شئ مربوط به رنگ شئ می‌باشد. همین جمله طور ذیل واضح شده می‌تواند:

Object Color ⟶ Weight

به همین ترتیب، اگر گفته شود که اشیای سرخ عبارت از توپ‌ها و اشیای آبی و زرد عبارت از شش ضلعی‌ها اند، جملهٔ ذیل نیز درست می‌باشد:

Object Color ⟶ Shape

چون رنگ شئ مشخص‌کنندهٔ (Determinant) وزن و شکل هر دو می‌باشد، پس

Object Color ⟶ (Weight, Shape)

همین رابطه طور ذیل نیز نشان داده شده می‌تواند:

جدول ۵-۱: رابطهٔ به‌وجودآمده از وابستگی تابعی

ObjectColor	Weight	Shape
Red	5	Ball
Blue	3	Cube
Yellow	7	Cube

در جدول بالا تمام شرایط ذکرشده، موجودیت یک جدول را نشان می‌دهد. ستون ObjectColor عبارت از (Primary-Key) است. پس چنین نوشته شده می‌تواند:

OBJECT (Object Color, Weight, Shape)

پس گفته می‌توانیم که کلید اصلی (Primary-Key) و کلید کاندید (Candidate-Key) هر کدام مشخص‌کنندهٔ (Determinant) جدول مربوطه می‌باشند و ستون‌های چون وزن و شکل وابسته (Dependent) به کلید اصلی هستند.

۵.۲ وابستگی با واسطه (Transitive Dependency)

وابستگی با واسطه (Transitive Dependency) عبارت از یک وابستگی تابعی ا ست که دارای خاصیت انتقالی باشد. این نوع وابستگی تنها در جدول‌هایی که تعداد ستون‌های آن سه و یا بیش‌تر از آن باشد، می‌تواند وجود داشته باشد.

وابستگی با واسطه (Transitive Dependency) وقتی به میان می‌آید که یک یا چندین ستون در یک جدول توسط یکی از ستون‌های غیر کلیدی در همان جدول مشخص (Determine) شده باشند.

به مثال ذیل توجه شود:

STUDENT(Stu_ID, Stu_Name, Dorm, Dorm_Location)

در مثال بالا ستون Stu_ID عبارت از کلید اصلی (Primary-Key) است و مشخص‌کنندهٔ (Determinant) متباقی ستون‌ها نیز می‌باشد. ستون Dorm که یک ستون غیر کلیدی است و توسط ستون Stu_ID مشخص شده است، خود ستون Dorm_Location را مشخص می‌کند؛ پس چنین نوشته شده می‌تواند:

Stu_ID ⟶ Dorm ⟶ Dorm_Location

جمله بالا یک وابستگی با واسطه (Transitive Dependency) را در جدول STUDENT نشان می‌دهد.

مثال دیگری در نظر گرفته شود:

VEHICLE(VIN, Make, Model, Year, NID, Owner)

در مثال بالا، ستون VIN عبارت از (Primary-Key) بوده و متباقی ستون‌های جدول را نیز مشخص می‌نماید. ستون NID که یک ستون غیر کلیدی است و توسط VIN مشخص شده است، خود مشخص‌کنندهٔ ستون Owner می‌باشد؛ پس چنین نوشته شده می‌تواند:

VIN ⟶ NID ⟶ Owner

این جمله نیز یک وابستگی با واسطه (Transitive Dependency) را در جدول VEHICLE نشان می‌دهد.

در یک جدول احتمال دارد چندین وابستگی با واسطه (Transitive Dependency) وجود داشته باشد و اضافه از دو مشخص‌کننده (Determinant) نیز وجود داشته می‌تواند.

۵.۳ نارمل‌سازی (Normalization)

یکی از اصول علم دیتابیس از بین‌بردن افزونی است. افزونی به این معناست که اطلاعات خاص در چند محل مختلف دیتابیس ذخیره شود. این امر موجب می‌شود که این خطر به وجود آید که اطلاعات هر لحظه با هم در تضاد قرار گیرند و استخراج معلومات از آن‌ها غیر ممکن شود. به بیان دیگر، پروسه‌یی است که بر اساس آن اطلاعات در واحدهای منطقی به نام جدول به شکلی توزیع می‌شود که علاوه بر حفظ جدول، اطلاعات از ایجاد افزونی جلوگیری به عمل می‌آورد؛ به این منظور شکل‌های نارمل متعددی تعریف و مورد استفاده قرار می‌گیرد. برای نارمل‌سازی یک جدول لازم است ابتداء در شکل اول نارمل شده و سپس در شکل‌های بعدی بررسی گردد. در هنگام نارمل‌سازی جدول‌ها، تعداد جدول‌ها در دیتابیس افزایش می‌یابد.

نارمل‌سازی مجموعه قوانینی است که رعایت آن‌ها حذف افزونی (یا در مواردی نادر به حد اقل رساندن آن) را تضمین می‌کند.

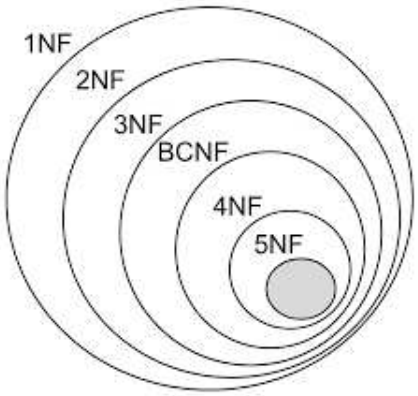
نارمل‌سازی (Normalization) یا به تعبیر دیگر معیاری‌سازی پروسه‌یی است در رابطه با دیتابیس، که با دو هدف عمدهٔ زیر انجام می‌شود:

کاهش افزونی اطلاعات: به این معنا که اطلاعات فقط در یک مکان (جدول) ذخیره و در تمام دیتابیس با استفاده از ارتباط منطقی تعریف‌شده (Relationship) قابل دست‌رسی باشد.
حفظ یک‌پارچگی اطلاعات: به این معنا که اعمال تغییرات بر روی اطلاعات (مانند ایجاد، به‌هنگام‌سازی و حذف) در یک مکان انجام و به دنبال آن آثار تغییرات در تمام دیتابیس مشاهده گردد.

برای روشن‌شدن مفهوم یک‌پارچگی بد نیست به مثال ذیل توجه نمایید:

فرض کنید در یک دیتابیس دارای دو جدول کتاب و نویسنده باشیم، هر یک از جدول‌های فوق دارای ستون‌های (Column) مختص به خود می‌باشند؛ به عنوان نمونه، جدول "کتاب" دارای ستون‌های نام نویسنده و جدول "نویسنده " دارای ستون‌های متعددی مانند نام نویسنده، آدرس نویسنده و ... باشد. در صورتی‌که در جدول "کتاب" یک سطر (Row) ایجاد نماییم، بدون این‌که نام نویسندهٔ آن را در جدول "نویسنده" ایجاد کرده باشیم، دچار یک ناهمگونی اطلاعات خواهیم شد.

با توجه به اهداف فوق می‌توان گفت که پروسهٔ نارمل‌سازی از مشکلات و خرابی‌های به‌وجودآمده به دلیل بروز تغییرات در دیتابیس جلوگیری خواهد نمود. با انجام‌دادن پروسهٔ نارمل‌سازی، یک دیتابیس کارآ و مطمئن را خواهیم داشت. در شکل ۵-۱ نشان‌دهندهٔ اشکال مختلف نارمل‌سازی را با ترتیب آن نشان داده شده است.



شکل ۵-۱: نمایش اشکال مختلف نارمل‌سازی

۵.۳.۱ شکل‌های نارمل

پروسهٔ نارمل‌سازی شکل‌های متفاوتی دارد که معمول‌ترین انواع آن به شرح ذیل است:

شکل اول نارمل‌سازی (۱NF)

شکل دوم نارمل (۲NF)
شکل سوم نارمل (۳NF)
شکل بویس کد نارمل (BCNF)
شکل چهارم نارمل (۴NF)

۵.۳.۲ شکل اول نارمل (۱NF(۱ Normal Form

جدولی در شکل اول نارمل است که تمامی ستون (Column)های آن یکتا و یا به اصطلاح Atomic باشند یا جدولی در شکل اول نارمل ا ست که شرایط زیر را داشته باشد:

۱. نوع اطلاعات در هر ستون یک‌سان باشد.
۲. محتویات هر ستون یک مقدار مشخص (Atomic) باشد.
۳. هر سطر از جدول منحصر به فرد باشد؛ به این منظور می‌توان از کلید اصلی یا (Primary-Key) استفاده نمود.
۴. هر ستون دارای نام مشخص می‌باشد.
۵. ستون‌های دیتا در مورد مشخصه‌های جدول را دارا می‌باشد.
۶. به ترتیب قرارگرفتن ستون ها در جدول اختیاری می‌باشد.
۷. به ترتیب قرارگرفتن سطرها در جدول اختیاری می‌باشد.

مثلاً جدول Table_Product _Price را در نظر می‌گیریم.

شکل ۲-۵: شکل اول نارمل

Table_Product_Price			
product id	Product Price	Model	Colore
1	2000	2000	red
2	1500	2010	black
3	2500	2020	yallow
4	2900	2000	blue
5	2400	2020	black
6	1600	2000	white
7	1500	2000	blue
8	2300	2000	white
9	3000	2000	red

برای روشن‌شدن این موضوع، فرض کنید جدولی با نام "رسید فروش" داشته باشیم که دارای ستون‌های ذیل باشد:

شمارهٔ فروش (کلید اصلی)؛
تاریخ فروش؛
کد مشتری؛
نام مشتری؛
جنس ۱؛
تعداد جنس ۱؛
قیمت واحد جنس ۱؛
...
جنس n؛
تعداد جنس n؛
قیمت جنس n.

با مشاهدهٔ جدول فوق متوجه این موضوع خواهیم شد که ستون‌های جنس، تعداد جنس و قیمت جنس بیش از یک مرتبه در جدول وجود داشته، به اصطلاح یک گروه تکرار را تشکیل می‌دهند. برای اجرای مودل فزیکی این جدول ناچار خواهیم بود در طراحی جدول لیست به طول ثابت (به عنوان نمونه با ده عضو) تعریف و در آن به ترتیب اجناس 1 تا 10 را تعریف نماییم.

مشکل

طراحی فوق ما را با دو مشکل عمده روبه‌رو خواهد ساخت:

اول این‌که کارایی دیتابیس پایین خواهد آمد. اگر در آینده تعداد اجناس رسید فروش بیش از ۱۰ جنس باشد، آن‌گاه مجبور خواهیم بود طراحی جدول مربوطه و متعاقب آن نرم‌افزارهایی را که از آن استفاده می‌کنند، تغییر دهیم.
دوم این‌که بسیاری از لست رسید فروش حتماً دارای ۱۰ جنس نیستند؛ بنابراین محتوای بسیاری از ستون‌ها در جدول فوق خالی (Null) خواهد ماند و حجم زیادی از حافظه ضایع می‌شود.

راه‌حل

برای حل این مشکل کافی است تمامی گروه‌های تکرار و یا لیست‌ها را از جدول خارج کرده، به جدول دیگری منتقل نماییم. در چنین مواردی، کلید اصلی جدول اول را به عنوان بخشی از کلید اصلی جدول جدید قرار داده و با پیوند یکی دیگر از اقلام جدول جدید که تضمین‌کنندهٔ یکتابودن سطرهای آن جدول است، کلید اصلی جدول ایجاد می‌گردد. بدین ترتیب، یک ارتباط بین جدول اصلی و فرعی بر اساس کلید اصلی جدول اصلی برقرار خواهد شد.

دوباره به جدول "رسید فروش" مثال قبل، پس از تبدیل به شکل اول نارمل توجه نمایید:

ارتباط بین جدول اصلی و فرعی بر اساس کلید اصلی جدول اصلی

جدول ۵-۲: شکل اول نارمل

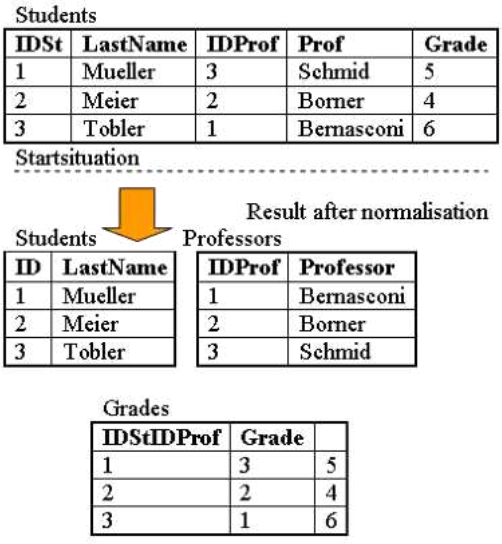
سطرهای رسید فروش	ارتباط بین دو جدول بر اساس کلید اصلی شمارهٔ فروش از جدول رسید فروش	رسید فروش
شمارهٔ فروش (کلید اصلی قسمت اول)		شمارهٔ فروش (کلید اصلی)
جنس (کلید اصلی قسمت دوم)		تاریخ فروش
تعداد		کد مشتری
قیمت فی واحد		نام مشتری

به طور خلاصه می‌توان گفت که هدف از شکل اول نارمل‌سازی حذف گروه‌های تکرار و لیست‌ها از جدول است. روش فوق باید بر روی تمامی جدول‌های دیتابیس انجام گردد تا بتوان گفت دیتابیس در شکل اول نارمل ا ست.

۵.۳.۳ شکل دوم نارمل ۲(NF (۲ Normal Form

جدولی در شکل دوم نارمل است که در شکل اول نارمل باشد و هم‌چنان تمامی ستون‌های (Column) غیر کلیدی آن وابستگی تابعی به تمام کلید اصلی جدول داشته باشند نه به بخشی از آن. یا جدولی در شکل دوم نارمل ا ست که شرایط زیر را داشته باشد:

۱. در شکل اول نارمل باشد.
۲. تمام ستون‌های غیر کلیدی وابستگی تابعی کامل به تمام ستون‌های کلیدی داشته باشد.



شکل ۵-۳: تبدیل شکل اول به شکل دوم نارمل

همان‌گونه که از تعریف فوق استنباط می‌گردد، شکل دوم نارمل‌سازی در خصوص جدول‌های بررسی و تطبیق می‌شود که دارای کلید اصلی مرکب هستند (بیش از یک جز). بنابراین، در شکل (5-2) جدول "رسید فروش" خود در شکل دوم نارمل است، ولی جدول "سطرهای رسید فروش" که دارای کلید اصلی مرکب است، نیاز به بررسی دارد.

مشکل

در صورتی‌که جدول در شکل دوم نارمل نباشد، آن‌گاه با تغییر اطلاعات قسمت‌های غیر وابسته به تمام کلید، این تغییرات در یک سطر اعمال می‌شود، ولی تاثیری بر روی سایر سطرها و یا جدول‌ها نخواهد داشت. در مثال فوق با تغییر محتوای "قیمت فی واحد" در جدول "سطر های رسید فروش"، قیمت فی واحد جنس در یک رسید فروش اصلاح می‌گردد اما در سایر رسیدها اعمال نخواهد شد.

راه‌حل

برای حل این مشکل کافی است جدول جدیدی ایجاد نماییم و کلید اصلی آن را برابر با آن بخش از کلید اصلی جدولی که دارای ستون‌های وابسته به آن است، مورد بررسی قرار دهیم. سپس تمام ستون‌های وابستهٔ تابعی به این کلید را از جدول مورد بررسی خارج کرده، به جدول جدید منتقل نماییم. در این حالت بین جدول جدید ایجادشده و جدول نارمل‌شده، بر اساس کلید اصلی جدول جدید ایجادشده یک ارتباط تعریف خواهد شد. دقت کنید که بر عکس شکل نارمل‌سازی، در این جا جدول مورد بررسی فرعی بوده و جدول جدید اصلی خواهد بود.

به مثال فوق برمی‌گردیم و شکل دوم نارمل‌سازی را بر روی آن تطبیق می‌نماییم. جدول "رسید فروش" دارای کلید مرکب نیست، پس در شکل دوم نارمل بوده و نیاز به بررسی ندارد، اما جدول "سطرهای رسید فروش" نیاز به بررسی دارد. در این جدول ستون "قیمت فی واحد" وابستگی تابعی به ستون جنس دارد که بخشی از کلید است نه کل کلید؛ پس لازم است تا این جدول را تبدیل به شکل دوم نارمل نماییم. بدین منظور جدول جدیدی به نام "جنس" ایجاد کرده، کلید اصلی آن را برابر "جنس" قرار داده و ستون قیمت واحد را از جدول قبلی خارج نموده، به این جدول منتقل می‌نماییم.

مثال فوق پس از تبدیل به شکل دوم نارمل به ترتیب ذیل خواهد بود:

جدول ۵-۳: شکل دوم نارمل

سطرهای رسید فروش		رسید فروش
شمارهٔ فروش (کلید اصلی قسمت اول)	**ارتباط بین دو جدول بر اساس کلید اصلی "شمارهٔ فروش" از جدول رسید فروش**	شمارهٔ فروش (کلید اصلی)
جنس (کلید اصلی قسمت دوم)		تاریخ فروش
تعداد		کد مشتری
ارتباط بین جدول سطرهای رسید فروش و جنس بر اساس کلید اصلی "جنس" از جدول جنس		نام مشتری
جنس		
جنس (کلید اصلی)		
قیمت فی واحد		

۵.۳.۴ شکل سوم نارمل ۳(NF (۳ Normal Form

جدولی در شکل سوم نارمل است که در شکل دوم نارمل بوده و نیز تمام ستون‌های غیر کلید آن وابستگی تابعی به کلید اصلی داشته باشند، نه به یک ستون غیر کلید. یا جدولی در شکل سوم نارمل است که شرایط زیر را داشته باشد:

۱. در شکل اول و دوم نرمال باشد.
۲. فاقد وابستگی‌های با واسطه باشد.

2NFa

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

3NFa

Lecturer	Dept
L1	D1
L2	D1
L3	D2
L4	D2

3NFb

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

شکل ۵-۴: تبدیل شکل دوم به شکل سوم نارمل

مشکل

در صورتی‌که جدول در شکل سوم نارمل نباشد، آن‌گاه با تغییر ستون و یا ستون‌های غیر وابسته به کلید اصلی در یک سطر، تغییرات در سایر سطرها تطبیق نخواهد شد و دچار دوگانگی اطلاعات خواهیم شد (مثال یک مشتری با دو نام متفاوت).

راه‌حل

کافی است ستون‌های غیر کلیدی به هم وابسته را به جدول جدیدی منتقل و کلید اصلی جدول جدید را تعیین نماییم. آن‌گاه کلید اصلی جدول جدید را در جدول نارمل‌شده باید به عنوان یک کلید خارجی (Foreign-Key) در نظر گرفت. در جدول "رسید فروش" مثال فوق، ستون نام مشتری وابستگی تابعی به ستون کد مشتری دارد که خود یک ستون غیر کلید است؛ بنابراین، باید نارمل‌سازی شکل سوم در خصوص آن تطبیق شود. شکل ذیل نحوهٔ انجام این کار را نشان می‌دهد:

جدول ۵-۴: شکل سوم نارمل

سطرهای رسید فروش		رسید فروش
شمارهٔ فروش (کلید اصلی قسمت اول)	**ارتباط بین دو جدول بر اساس کلید اصلی**	شمارهٔ فروش (کلید اصلی)
جنس (کلید اصلی قسمت دوم)		تاریخ فروش
تعداد		کد مشتری (کلید خارجی)

ارتباط بین جدول سطرهای رسید فروش و جنس بر اساس کلید اصلی "جنس" از جدول جنس	"شمارهٔ فروش" از جدول رسید فروش	ارتباط بین جدول رسید فروش و مشتری بر اساس کلید خارجی (کد مشتری)
جنس		مشتری
جنس (کلید اصلی)		کد مشتری
قیمت فی واحد		نام مشتری

۵.۳.۵ شکل بویس کد نارمل BCNF (Boyce–Code Normal Form)

شکل بویس کد (Boyce-Code Normal Form) دارای مفهوم جامع‌تری نسبت به شکل دوم و سوم نارمل است. در شکل دوم و سوم نارمل بحث بر سر وابستگی تابعی ستون‌های غیر کلیدی به کلید اصلی است، اما در شکل بویس کد، جدولی در شکل بویس کد نارمل است که در شکل اول نارمل بوده و نیز تمام ستون‌های غیر کلیدی آن به طور کامل وابستگی تابعی به یک کلید باشند و نه چیز دیگر. نکتهٔ حائز اهمیت در این شکل این است که بحث بر سر وابستگی تابعی با یک کلید است نه فقط کلید اصلی. مفهوم فوق در خصوص جدول‌هایی که دارای چندین کلید (Alternate-Key) هستند، مطرح می‌شود.

۵.۳.۶ شکل چهارم نارمل ۴ NF(۴ Normal Form)

این شکل در خصوص جدول‌هایی است که ارتباط بین ستون‌های آن یک ارتباط چند مقدار و یا چند به چند باشد؛ به عنوان مثال، جدول صنف می‌تواند شامل چندین شاگرد و چندین استاد باشد. در چنین مواردی ارتباط بین استاد و شاگرد یک ارتباط چند به چند می‌باشد. در این حالت با ایجاد یک جدول رابط مابین جدول‌های مذکور، مشکل ارتباط چند به چند حل خواهد شد (بسیاری از سیستم‌های مدیریت دیتابیس رابطه‌یی از رابطهٔ چند به چند پشتیبانی نمی‌نمایند؛ یعنی نمی‌توان بین دو جدول یک رابطهٔ چند به چند ایجاد نمود). به طور عموم تمام ستون‌های جدول رابط ایجادشده بخشی از کلید اصلی است.

خلاصهٔ فصل پنجم

یکی از اصول علم دیتابیس از بین‌بردن افزونی است. افزونی به این معناست که اطلاعات خاص در چند محل مختلف دیتابیس ذخیره شود. این امر موجب می‌شود که این خطر به وجود آید که اطلاعات هر لحظه باهم در تضاد قرار گیرند و استخراج معلومات از آن‌ها غیر ممکن شود. نارمل‌سازی مجموعه قوانینی است که رعایت آن‌ها حذف افزونی (یا در مواردی نادر به حد اقل رساندن آن) را تضمین می‌کند.

نارمل‌سازی شکل‌های دیگری نیز دارد که به دلیل نادربودن و خاص‌بودن آن‌ها در این فصل به آن‌ها اشاره نشده است. آن‌چه در خصوص نارمل‌سازی عمومیت دارد تا شکل سوم آن است؛ یعنی در هنگام طراحی دیتابیس‌ها باید پروسهٔ نارمل‌سازی تا شکل سوم را انجام داد. پروسهٔ نارمل‌سازی یک پروسهٔ تکراری (Recursive) است؛ یعنی پس از هر مرحلهٔ نارمل‌سازی که منجر به ایجاد جدول‌های جدید می‌گردد، پروسه را باید از ابتداء تا انتها بر روی جدول‌های تازه ایجادشده نیز اجراء نمود.

سوالات و فعالیت های فصل پنجم

۱. وابستگی تابعی را تشریح نمایید.

۲. وابستگی با واسطه چیست؟ معلومات ارائه نمایید.

۳. در مورد اشکال موجود در نارمل‌سازی معلومات ارائه نمایید. (جست‌وجو در انترنت)

۴. در مورد شکل DKNF نارمل‌سازی توضیح دهید. (جست‌وجو در انترنت)

۵. در مورد شکل پنجم نارمل‌سازی معلومات دریابید. (جست‌وجو در انترنت)

۶. نارمل‌سازی چیست؟ توضیح دهید.

۷. شکل اول نارمل‌سازی را با یک مثال واضح سازید.

۸. شکل دوم نارمل‌سازی را با یک مثال واضح سازید.

۹. شکل سوم نارمل‌سازی را با یک مثال واضح سازید.

۱۰. نکات مهم در مورد شکل بویس کد و شکل چهارم نارمل‌سازی را بیان دارید.

فعالیت‌ها

۱. جدول مورد نظرتان را انتخاب نموده، وابستگی تابعی بین ستون‌های آن را تشخیص دهید. (انفرادی)

۲. جدول مورد نظرتان را انتخاب نموده، وابستگی‌های با واسطه را بین ستون‌های آن تشخیص دهید. (انفرادی)

۳. جدول مورد نظرتان را بعد از تشخیص وابستگی‌ها، تا شکل سوم نارمل‌سازی نمایید. (گروپی)

فصل ششم

ER Diagram

هدف کلی: آشنایی محصلان با ER Diagram.

اهداف آموزشی: در پایان این فصل، محصلان قادر خواهند بود تا:

۱. ER Diagram را شرح دهند.
۲. اشکال مختلف در ER Diagram را تشخیص دهند.
۳. برای دیتابیس مورد نظر ER Diagram ترسیم نمایند.

در این فصل به توضیح مودل ER پرداخته و موارد استفاده و طرز ایجاد و یا ترسیم ER Diagram را به تفصیل مورد بحث قرار می‌دهیم. بعد از تشریح ER دیاگرام یک پروژهٔ مکمل برای یک دانشگاه ایجاد می‌نماییم. هم‌چنان مراحل و قواعد برای تبدیل ER دیاگرام را به یک دیتابیس به معرفی می‌گیریم.

۶.۱ مودل (Entity Relationship)ER

اطلاعاتی که قرار است در دیتابیس ذخیره شوند، ابتدا باید با یک دید سطح بالا از لحاظ معنایی و مفهومی مودل‌سازی شوند. مودل‌ها و روش‌های مختلفی برای مودل‌سازی معنایی اطلاعات وجود دارند که از معروف‌ترین آن‌ها می‌توان از مودل ER، .مودل Niam، مودل زبان یکپارچهٔ مودل‌سازی UML و غیره نام برد. مودل ER در اواسط دههٔ ۸۰ میلادی در دانشگاه MIT توسط فردی به نام چن پیشنهاد شد. این مودل به مرور زمان پیشرفت کرد و کم کم به EER(Enhanced Entity-Relationship) معروف شد، اما هنوز هم در خیلی جاها با همان نام ER از آن نام برده می‌شود. در این بخش به شرح کامل این مودل می‌پردازیم.

در مودل ER سه مفهوم اصلی وجود دارد که عبارت‌اند از: نهاد (Entity)، صفت(Attribute) و ارتباط (Relationship). ER هم یک دیاگرام است برای مودل‌سازی معنایی اطلاعات بر پایهٔ مودل ER و در واقع سه مفهوم اساسی مودل ER، یعنی نوع جدول، صفت و ارتباط را به صورت شکلی نشان می‌دهد. قبل از هر چیز، بهتر است با این سه مفهوم آشنا شویم:

۶.۱.۱ جدول در ER Diagram

مفهوم کلی شئ، پدیده و به طور کلی هر آن چیزی است که می‌خواهیم در موردش اطلاعاتی در دیتابیس داشته باشیم و با استفاده از آن، اطلاعات و شناخت خود را در موردش افزایش دهیم؛ به عنوان مثال، برای یک سیستم آموزشی که قرار است در یک دانشگاه استفاده شود، جدول‌های دانشجو، استاد، دانشکده و درس را می‌توان نام برد.

همان‌طوری‌که می‌بینیم جدول یک مفهوم کلی و یا یک نوع اطلاعات واحد است، اما هر جدول می‌تواند نمونه‌های داشته باشد. نمونه‌های یک جدول، مثال‌های آن جدول هستند و در واقع نمونه‌هایی هستند که وجود خارجی دارند؛ به عنوان مثال، درس دیتابیس‌ها با یک سلسله ویژگی‌های خاص خودش یک نمونه از جدول درس است.

اولین گام در مودل‌سازی معنایی دیتابیس، شناسایی جدول‌های موضوع است، اما سوالی که ممکن است پیش بیاید این است که مثلاً در مثال فوق، چرا خود دانشگاه را به عنوان یکی از جدول‌ها در نظر نگرفتیم؟ یا به طور کلی بر چه اساسی اقدام به شناسایی جدول‌ها می‌کنیم؟

یک قانون ساده برای تشخیص این‌که یک مفهوم خاص را آیا جدول به حساب آوریم یا خیر، این است که آن مفهوم بتواند شامل چند نمونهٔ متمایز در رابطه با موضوع ما باشد. به بیان ساده، چیزی جدول تلقی می‌شود

که بخواهیم مجموعه‌یی از آن را در سیستم نگهداریم. در مثال فوق، دانشگاه را جدول به حساب نمی‌آوریم؛ چون سیستم قرار است اطلاعات محصلان یک دانشگاه را مدیریت کند. در این‌جا دانشگاه خود حوزهٔ اصلی موضوع است که شامل جدول‌های نام‌برده می‌باشد، اما اگر صورت موضوع را این‌طور عوض کنیم که می‌خواهیم اطلاعات آموزشی محصلان سراسر کشور را از طریق این سیستم ذخیره و بازیابی کنیم، آن‌گاه دانشگاه هم یک جدول خواهد بود.

نکتهٔ دیگر در مورد جدول‌ها این است که یک جدول به طور معمول بیش از یک ویژگی دارد که باید ذخیره شود، اما همان‌طوری‌که خواهیم دید، این قانون همیشه صادق نیست.

در ER دیاگرام هر جدول به شکل یک مستطیل ساده نمایش داده می‌شود.

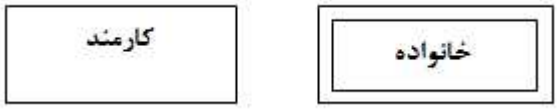
شکل ۶-۱: نمایش جدول در **ER** دیاگرام

۶.۱.۲ جدول مستقل و وابسته(Depended and Independed Table)

جدول مستقل (قوی)، عبارت از جدولی است که مستقل از هر جدول دیگر و به خودی خود در یک محیط مشخص مطرح باشد. جدول وابسته (ضعیف)، عبارت از جدولی است که وجودش وابسته به یک نوع جدول دیگر است. در واقع هر نمونهٔ موجود از یک جدول ضعیف به یکی از نمونه‌های یک جدول قوی وابسته است. به این نوع ارتباط بین یک جدول ضعیف با یک جدول قوی، وابستگی وجودی گفته می‌شود.

به عنوان مثال، فرض کنید کارمند و معاش دو جدول در یک سیستم کارمندان باشند؛ پس اطلاعات تعدادی کارمند و نیز مقدار معاش (به عنوان نمونه‌های دو جدول) در سیستم ذخیره شده اند. اگر یک کارمند را از سیستم حذف کنیم، وجود معاش‌های مربوط به او دیگر مفهومی ندارد و باید حذف شوند. پس در این مثال، کارمند یک جدول مستقل و معاش یک جدول وابسته است. اگر اطلاعات افراد خانوادهٔ کارمندان (اقارب کارمند) نیز به عنوان یک جدول دیگر در این سیستم ذخیره شده باشد، آن نیز یک جدول وابسته به جدول کارمند می‌باشد.

برای دانستن تفاوت جدول‌های وابسته از جدول‌های مستقل، در ER دیاگرام، جدول‌های وابسته با دو خط نشان داده می‌شوند.



شکل ۶-۲: نمایش جدول مستقل و وابسته

۶.۲ صفت (Attribute)

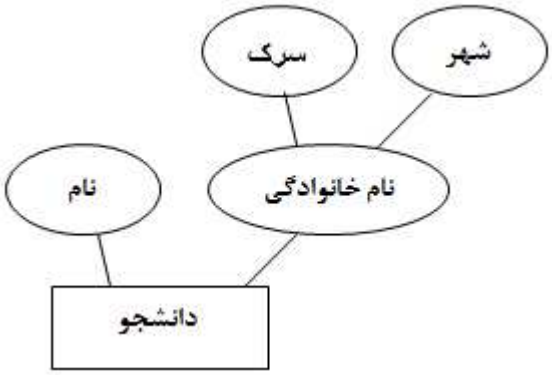
صفت (Attribute) عبارت از خاصیت یا ویژگی یک نوع جدول است. هر نوع جدول مجموعه‌یی از صفات دارد. هر صفت یک نام، یک نوع و یک معنای مشخص دارد. در ER دیاگرام هر یک از صفات یک جدول با شکل بیضوی نشان داده شده و به جدول متصل می‌شوند.

شکل ۶-۳: نمایش صفت در **ER**

۶.۳ انواع صفت (Types of Attributes)

۶.۳.۱ صفت ساده و مرکب (Simple and Composite Attributes)

صفت ساده صفتی است که مقدار آن از لحاظ معنایی غیر قابل تجزیه می‌باشد. صفت مرکب صفتی است که از چند صفت ساده تشکیل شده است و به هر جزء آن بتوان به طور مستقیم دست‌رسی داشت؛ برای مثال، صفت آدرس که خود شامل صفات جزئی‌تر شهر، سرک و ... است، می‌تواند یک صفت مرکب باشد؛ در صورتی‌که ساختار آن به گونه‌یی باشد که بتوان به این اجزاء به طور مجزاء و مستقیماً دست‌رسی داشت، اما اگر ویژگی آدرس برای یک جدول به صورت یک رشتهٔ معمولی در نظر گرفته شود، دیگر یک صفت مرکب محسوب نمی‌شود؛ چون دست‌رسی مستقیم به هر جزء آن نداریم.



شکل ۶-۴: نمایش صفت ساده و مرکب

در دیتابیس رابطه‌یی، امکان تعریف صفات مرکب وجود ندارد و همهٔ صفت‌ها باید ساده باشند. پس اگر بخواهیم به اجزای یک صفت مانند آدرس دست‌رسی داشته باشیم، چه باید بکنیم؟ راه حل کلی برای این

موضوع این است که در صورتی‌که چنین ضرورتی وجود دارد، به جای تعریف یک صفت خاص به نام آدرس برای جدول، اجزای آن را به طور جداگانه (چند صفت ساده به جای یک صفت) در نظر بگیریم.

۶.۳.۲ صفت یک‌مقداری و چندمقداری(Single value and Multi value)

صفت یک‌مقداری، صفتی است که برای یک نمونه از یک نوع جدول حد اکثر یک مقدار از حوزهٔ مقادیر را می‌گیرد. صفتی که برای بعضی از نمونه‌های جدول ممکن است بیش از یک مقدار داشته باشد، یک صفت چندمقداری است؛ به عنوان مثال، صفت نمبر اساس، نام خانوادگی و غیره برای جدول دانشجو صفات یک مقداری محسوب می‌شوند؛ چون هر دانشجو تنها یک نام خانوادگی و یک نمبر اساس دارد، اما صفتی مانند شمارهٔ موبایل برای جدول دانشجو یک صفت چندمقداری است؛ زیرا بعضی از محصلان بیش از یک شمارهٔ موبایل دارند.

نمایش صفات خاص چندمقداری در ER دیاگرام به صورت زیر است:

شکل ۶-۵: صفت یک‌مقداری و چندمقداری

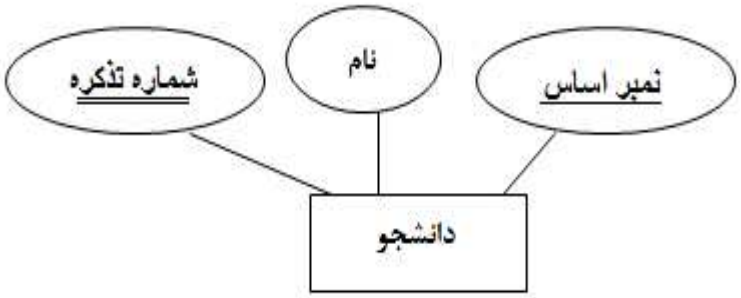
۶.۳.۳ صفت کلید و غیر کلید(Identifire and Non Identifire Attributes)

صفت کلید یا شناسه (هویت) جدول صفتی است که مقدار یکتا دارد. به عبارت دیگر، مقدار این صفت بین نمونه‌های مختلف تکرار نمی‌شود؛ به عنوان مثال، نمبر اساس برای جدول دانشجو یک صفت کلید است.

در مودل‌سازی معنایی، برای هر جدول باید یک شناسه مشخص کرد، اما یک جدول ممکن است بیش از یک شناسه داشته باشد؛ مثلاً شمارهٔ تذکره نیز یک شناسهٔ دیگر برای جدول دانشجو است. در چنین حالاتی، یکی از شناسه‌ها را باید به عنوان شناسهٔ اول (اصلی) مشخص کرد و صفت دیگر نیز می‌تواند به عنوان شناسهٔ دوم (فرعی) معرفی شود.

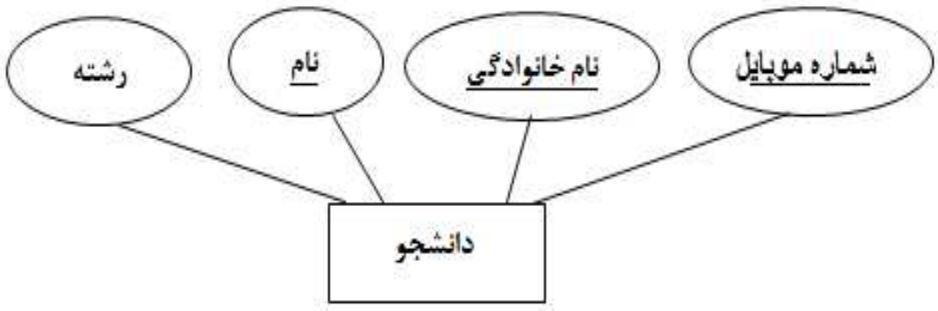
صفتی که قرار است به عنوان شناسهٔ اصلی یک جدول استفاده شود، بهتر است تا حد امکان طول مقادیرش کوتاه باشد.

در ER دیاگرام شناسهٔ اول یک جدول با کشیدن یک خط و شناسهٔ دوم با کشیدن دو خط در زیر عنوان آن مشخص می‌شود.



شکل ۶-۶: نمایش صفت کلید و غیرکلید

گاهی اوقات، یک جدول صفت تکرارناپذیری ندارد تا به عنوان کلید آن در نظر گرفته شود. یک راه برای حل این مشکل استفاده از کلیدهای ترکیبی است، یعنی باید ترکیبی از صفات را بیابیم که آن ترکیب بین نمونه‌های آن جدول تکرار نشود؛ مثلاً فرض کنیم جدول دانشجو در اصل صفاتی به نام نمبر اساس و شمارهٔ تذکره ندارد. در این حالت با این فرض که ترکیب سه صفت نام، نام خانوادگی و شمارهٔ کارت هویت بین محصلان امکان تکرار ندارد، می‌توان ترکیب این سه صفت را به عنوان کلید جدول دانشجو معرفی کرد. به چنین کلیدهای کلید ترکیبی گفته می‌شود. روش نمایش کلیدهای ترکیبی در ER دیاگرام به صورت زیر است:



شکل ۶-۷: نمایش کلیدهای ترکیبی

۶.۳.۴ صفت هیچ‌مقدارپذیر یا هیچ‌مقدارناپذیر (Null and Not Null Attributes)

هیچ‌مقدار یعنی مقدار ناشناخته، مقدار غیر قابل تطبیق یا مقدار تعریف‌نشده که به طور معمول با واژهٔ Null نشان داده می‌شود. اگر مقدار یک صفت در یک یا بیش از یک نمونه از یک جدول، بتواند برابر با هیچ‌مقدار باشد، آن صفت هیچ‌مقدارپذیر است.

هیچ‌مقدارپذیربودن یا نبودن هر صفت توسط طراح دیتابیس و با توجه به اهمیت آن صفت تعیین می‌شود؛ به طور مثال، نام خانوادگی دانشجو به دلیل این‌که مهم است باید برای هر دانشجو مقدارش مشخص باشد و بهتر است هیچ‌مقدارپذیر نباشد، اما صفتی مثل شمارهٔ موبایل به دلیل اهمیت کم‌تر و نیز به این علت که ممکن است دانشجو شمارهٔ موبایل نداشته باشد، ممکن است هیچ‌مقدارپذیر تعریف شده باشد.

۶.۳.۵ صفت ذخیره‌شده و مشتق

صفت ذخیره‌شده صفتی از یک جدول است که مقادیرش برای نمونه‌های آن جدول در دیتابیس ذخیره شود. صفت مشتق صفتی است که مقادیرش در دیتابیس ذخیره نشده باشد، بلکه از روی اطلاعات ذخیره‌شده قابل محاسبه باشد؛ به عنوان مثال، اگر تاریخ تولد هر دانشجو (به عنوان یکی از صفات) در سیستم ذخیره شود، دیگر نیازی به ذخیرهٔ صفت سن دانشجو نیست و این ویژگی با توجه به تاریخ تولد قابل محاسبه است. پس در این‌جا تاریخ تولد یک صفت ذخیره‌شده و سن یک صفت مشتق است.

برای نمایش صفات مشتق در ER دیاگرام، می‌توان بیضوی مربوط به آن صفت و یا خط اتصال آن صفت به جدول را به صورت نقطه چنین رسم کرد:

شکل ۶-۸: نمایش صفت ذخیره‌شده و مشتق

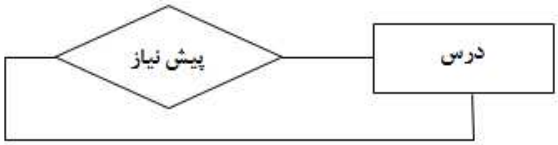
۶.۴ ارتباط(Relation)

ارتباط عبارت از تعامل و وابستگی بین دو یا بیش از دو نوع جدول است. ارتباط بین دو جدول در حقیقت ارتباط بین نمونه‌های آن دو جدول است. هر ارتباطی دارای یک عنوان و یک مفهوم مشخص است؛ به عنوان مثال، بین دو جدول دانشجو و درس ارتباط "انتخاب" وجود دارد، یعنی نمونه‌های جدول دانشجو، نمونه‌های از جدول درس را انتخاب می‌کنند.

در ER دیاگرام ارتباط با شکل لوزی نشان داده می‌شود.

شکل ۶-۹: نمایش ارتباط درجه دو در **ER**

اگر نمونه‌هایی از یک جدول بتوانند با نمونه‌هایی از همان جدول ارتباط داشته باشند، باید یک ارتباط از جدول به خودش رسم شود؛ مثلاً، بین نمونه‌های جدول درس، ارتباط پیش‌نیازی برقرار است.



شکل ۶-۱۰: نمایش ارتباط بازگشتی در **ER**

۶.۴.۱ ماهیت ارتباط(Relationship Nature)

ماهیت ارتباط یا به عبارت دیگر نوع تناظر بین نمونه‌های دو جدول حاضر در ارتباط را نشان می‌دهد. ماهیت ارتباط می‌تواند یک به یک، یک به چند و چند به چند باشد.

ارتباط (یک به یک): هر نمونه از جدول اول می‌تواند فقط با یک نمونه از جدول دوم ارتباط داشته باشد و هر نمونه از جدول دوم نیز فقط می‌تواند با یک نمونه از جدول اول ارتباط داشته باشد. در واقع همان مفهوم تناظر یک به یک در نظریهٔ مجموعه‌هاست.

به عنوان مثال، اگر درجهٔ ارتباط بین جدول‌های دانشجو و درس (یک به یک) باشد (مطابق شکل زیر)، مفهوم ارتباط چنین است:

شکل ۶-۱۱: نمایش ارتباط یک به یک

هر دانشجو می‌تواند فقط یک درس را بگیرد و هر درس هم می‌تواند فقط توسط یک دانشجو گرفته شود.

ارتباط (یک به چند): هر نمونه از جدول اول می‌تواند با چند نمونه از جدول دوم ارتباط داشته باشد، اما هر نمونه از جدول دوم فقط می‌تواند با یک نمونه از جدول اول ارتباط داشته باشد.

اگر درجه ارتباط بین جدول‌های دانشجو و درس (یک به چند) باشد (مطابق شکل زیر)، مفهوم ارتباط چنین است:

شکل ۶-۱۲: نمایش ارتباط یک به چند

هر دانشجو می‌تواند چند درس بگیرد اما هر درس می‌تواند تنها توسط یک دانشجو گرفته شود.

ارتباط(چند به چند): هر نمونه از جدول اول می‌تواند با چند نمونه از جدول دوم ارتباط داشته باشد و هر نمونه از جدول دوم نیز می‌تواند با چند نمونه از جدول اول ارتباط داشته باشد.

حال اگر درجهٔ ارتباط بین جدول‌های دانشجو و درس (چند به چند) باشد (مطابق شکل زیر)، مفهوم ارتباط چنین است:

شکل ۶-۱۳: نمایش ارتباط چند به چند

هر دانشجو می‌تواند چند درس را بگیرد و هر درس هم می‌تواند توسط چند دانشجو گرفته شود.

۶.۴.۲ حد ارتباط(Cardinality)

برای هر یک از جدول‌های شرکت‌کننده در یک ارتباط می‌توان یک حد مشخص کرد. این حد نشان‌دهندهٔ حد اقل و حد اکثر تعداد نمونه‌های آن جدول است که می‌توانند در ارتباط شرکت کنند.

به عنوان مثال، حدود نشان‌داده‌شده در شکل زیر برای دو جدول دانشجو و درس بیان‌گر مفاهیم زیر هستند:

هر دانشجو حد اقل ۱ و حد اکثر ۵ درس را می‌تواند انتخاب کند. هر درس می‌تواند توسط هیچ دانشجو انتخاب نشود و یا حد اکثر توسط ۲۰ دانشجو انتخاب شود.

شکل ۶-۱۴: نمایش حد ارتباط در **ER**

۶.۴.۳ شرکت اجباری و اختیاری در ارتباط

جدول‌هایی‌که در یک ارتباط شرکت دارند، نوع ارتباط شان می‌تواند اجباری یا اختیاری باشد. اگر مقدار حد اقل اشتراک یک جدول در یک ارتباط باشد، به این معنا است که نمونه‌یی از این جدول می‌تواند وجود داشته باشد که اصلاً در ارتباط شرکت نکند. در این حالت شرکت جدول را اختیاری می‌گوییم و در غیر این صورت، شرکت جدول در ارتباط اجباری است.

اگر شرکت جدول (دانشجو) در ارتباط (انتخاب درس) اجباری و شرکت (درس) در این رابطه اختیاری باشد، این دو موضوع به شکل زیر در ER دیاگرام نشان داده می‌شوند.

شکل ۶-۱۵: نمایش شرکت اجباری و اختیاری

روش دیگر نمایش ارتباط اجباری با استفاده از دو خط است:

شکل ۶-۱۶: نمایش شرکت اختیاری و اجباری طریقهٔ دوم

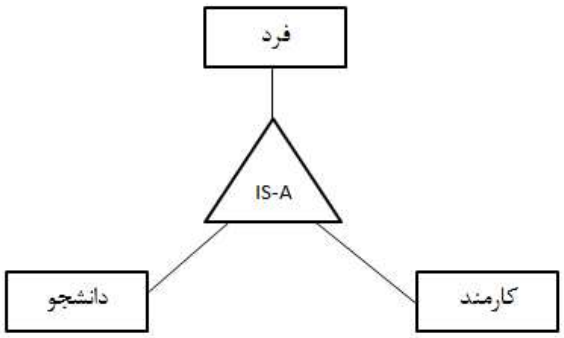
نکتهٔ مهم: در یک ارتباط (یک به چند) اگر نوع مشارکت جدول سمت چند اجباری باشد، این جدول یک جدول وابسته است که وابستگی آن به جدول سمت یک است.

۶.۴.۴ ارتباط IS–A (هست یک)

نوعی ارتباط بین دو جدول است و زمانی برقرار است که جدول دوم حالت خاص‌تری از جدول اول باشد. به عبارت دیگر، جدول دوم برعلاوهٔ یک یا چند صفت اضافه‌تر مخصوص خودش، همهٔ صفت‌های جدول اول را دارد. این ارتباط در واقع همان ارتباط وراثت بین دو جدول است.

به عنوان مثال، اگر دو جدول دانشجو و کارمند داشته باشیم، هر دوی این جدول‌ها در بعضی صفات عمومی از قبیل نام، نام خانوادگی و محل سکونت مشترک هستند. صفات مذکور صفاتی هستند که هر فرد انسانی دارد. پس می‌توان جدول دیگری به نام فرد در نظر گرفت که شامل این صفات عمومی است و در واقع گسترش‌دهندهٔ دو جدول دیگر است. در این حالت هر یک از دو جدول دیگر با این جدول ارتباط IS-A دارند (دانشجو هست یک فرد؛ کارمند هست یک فرد).

در ER دیاگرام، ارتباط IS-A به صورت زیر نشان داده می‌شود:



شکل ۶-۱۷: نمایش ارتباط **IS-A**

این ارتباط بیش‌تر در دیتابیس‌های شئ‌گرا مناسب و مفید است؛ چرا که با رعایت سلسله‌مراتب مفهومی بین جدول‌ها از تعریف صفات تکراری در آن‌ها خودداری می‌کنیم، اما در دیتابیس رابطه‌یی نیز امکان پیاده‌سازی این ارتباط وجود دارد. این کار به دو روش قابل انجام است:

روش اول: برای هر جدول سطح بالا و پایین یک جدول جداگانه در نظر گرفته شده که در جدول سطح پایین کلید اصلی جدول سطح بالاتر به عنوان کلید خارجی قرار می‌گیرد.

مشکل این روش این است که برای بازیابی اطلاعات یک دانشجو نیازمند مراجعه به دو جدول فرد و دانشجو هستیم.

روش دوم: همهٔ صفت‌های جدول سطح بالاتر که جدول سطح پایین‌تر نیز آن‌ها را داراست، در جدول سطح پایین‌تر قرار می‌گیرد. در این حالت، دیگر نیازی به وجود جدول سطح بالاتر نخواهد بود، مگر این‌که ارتباط موجود یک توسعهٔ کامل نباشد؛ یعنی جدول دوم فقط تعدادی از صفات جدول اول را داشته باشد. کاملاً روشن است که مشکل این روش، افزونی اطلاعات است.

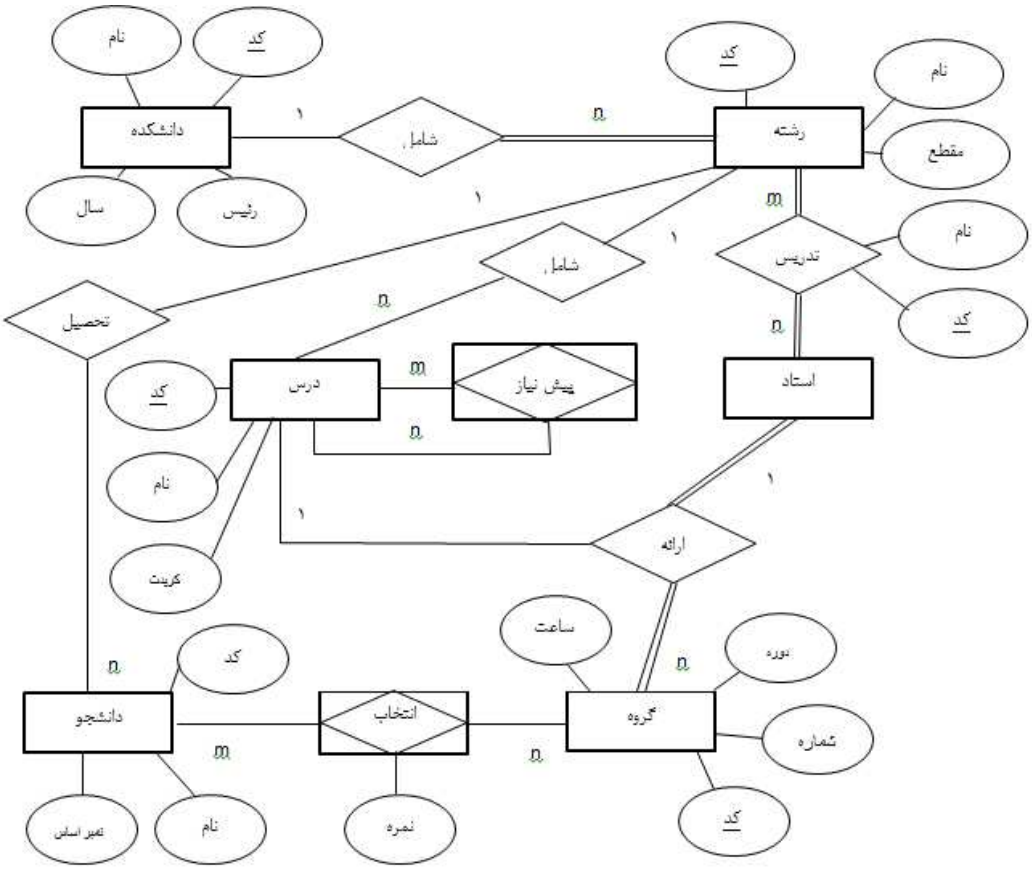
۶.۵ پروژهٔ رسم ER دیاگرام سیستم دانشگاه

برای یک سیستم دانشگاه با مشخصات زیر یک ER دیاگرام طراحی کنید و سپس آن را به دیتابیس رابطه‌یی تبدیل کنید.

دانشگاه شامل تعدادی دانشکده است که هر دانشکده یک کد، نام، رئیس و یک سال تاسیس دارد. در هر دانشکده یک تعداد رشته‌ها اند که هر رشته شامل یک کد، یک نام و یک مقطع است. در هر رشته تعدادی دانشجو تحصیل می‌کنند که هر دانشجو دارای نمبر اساس، نام و نام خانوادگی می‌باشد. هر رشته تعدادی استاد دارد که البته بعضی از اساتید بین چند رشته مشترک هستند. هر استاد دارای کد و نام است.

در سرفصل هر رشته تعدادی درس تعریف شده که هر درس دارای کد، نام درس و تعداد کریدت است. هر درس می‌تواند چندین پیش‌نیاز داشته باشد و یا پیش‌نیاز چندین درس دیگر باشد.

در هر "دوره" از هر "درس" چندین "گروه درسی" ارائه می‌شود که هر "گروه" یک کد، یک شماره و یک ساعت تشکیل دارد. از دروس ارائه‌شدهٔ هر دانشجو چندین مورد را انتخاب می‌کند و در نهایت برای هر کدام نمره‌یی کسب می‌کند.



شکل ۶-۱۸: پروژهٔ **ER** دیاگرام

جدول ۱-۶: رشته‌ها

جدول رشته			
کد رشته	نام	مقطع	کد پوهنځی

جدول ۶-۲: دانشکده‌ها

جدول دانشکده			
کد پوهنځی	نام	رئیس	سال تأسیس

جدول ۶-۳: درس‌ها

جدول درس			
کد درس	نام	کریدت	کد رشته

جدول ۶-۴: استادان

جدول استاد	
کد استاد	نام

جدول ۶-۵: ارتباط استاد با رشته

جدول رابط استاد و رشته	
کد رشته	کد استاد

جدول ۶-۶: انتخاب‌ها

جدول انتخاب		
نمبر اساس	کد گروه	نمره

جدول ۶-۷: محصلان

جدول محصل			
نمبر اساس	نام	نام خانوادگی	کد رشته

جدول ۶-۸: گروه‌ها

جدول گروه					
کد گروه	شماره	دوره	ساعت	کد درس	کد استاد

جدول ۶-۹: پیش نیازها

جدول پیش‌نیاز	
کد درس پیش	کد درس پس

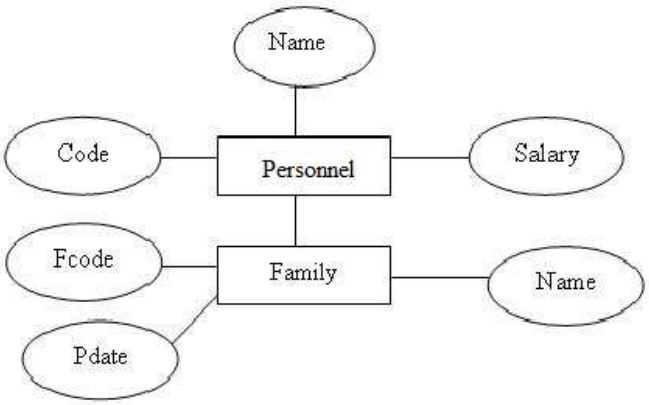
۶.۶ تبدیل ER دیاگرام به دیتابیس

در این قسمت چند قاعدهٔ ساده برای تبدیل ER دیاگرام به دیتابیس را ارائه می‌دهیم. با استفاده از این قواعد می‌توان یک دیتابیس نارمل ایجاد کرد. برای تبدیل ER دیاگرام به دیتابیس قواعد زیر را استفاده می‌کنیم:

قاعدهٔ اول: هر نهاد (Entity) قوی به یک جدول تبدیل می‌شود که دارای ستون‌هایی برای هر کدام از صفات بوده و هر بخش از صفات مرکب به صورت یک ستون جداگانه پیاده‌سازی می‌گردد.

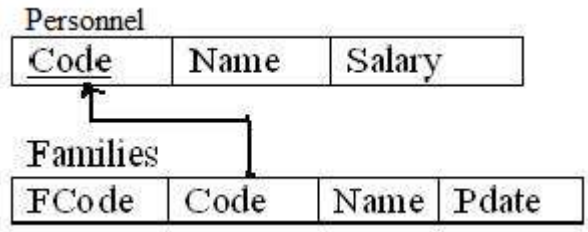
قاعدهٔ دوم: هر نهاد ضعیف (Weak Entity) تبدیل به یک جدول شده و این جدول دارای یک کلید خارجی متناظر با کلید اصلی در جدول مربوط به جدول قوی است.

مثال:



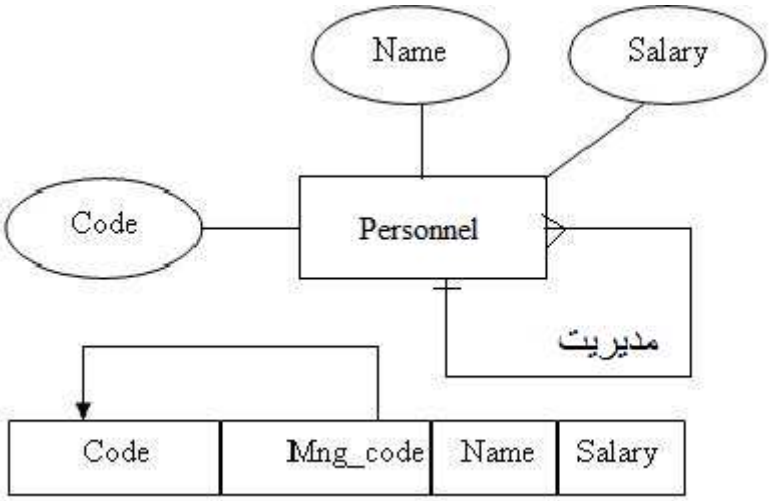
شکل ۶-۱۹: تطبیق قاعدهٔ دوم

Family یک نهاد ضعیف است چون وجودش وابسته به نهاد Personnel می‌باشد؛ یعنی اگر کارمند وجود نداشته باشد، اعضای خانوادهٔ او هم نمی‌توانند وجود داشته باشند. پیاده‌سازی آن به صورت زیر خواهد بود:



شکل ۶-۲۰: پیاده‌سازی قاعدهٔ دوم

قاعدهٔ سوم: برای یک رابطهٔ درجه یک (رابطه‌یی که بین یک جدول و خودش است)، یک ستون اضافی به عنوان کلید خارجی در آن ایجاد کرده که در ارتباط با کلید اصلی جدول می‌باشد.



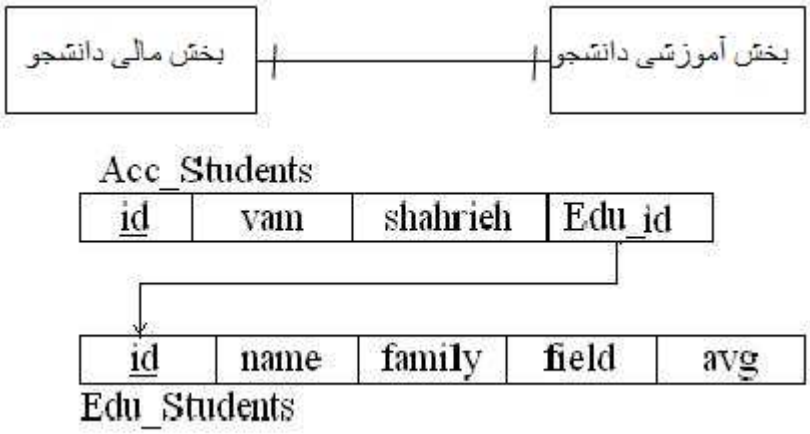
شکل ۶-۲۱: تطبیق قاعدهٔ سوم

مثال

جدول ۶-۱۰: کارمندان

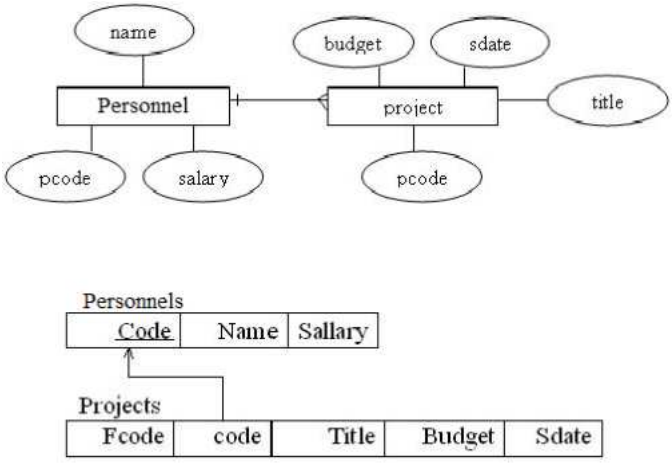
Code	Mng_Code	Name	Salary
1	0	Ali	200
2	1	Reza	150
3	1	Sara	120
4	3	Hadi	110
5	3	Zahra	150

قاعدهٔ چهارم: برای یک رابطهٔ درجه دو و با کاردینالیتی (یک به یک)، برای هر نهاد یک جدول ایجاد کرده و کلید اصلی آن دو جدول با هم متناظر خواهد بود.



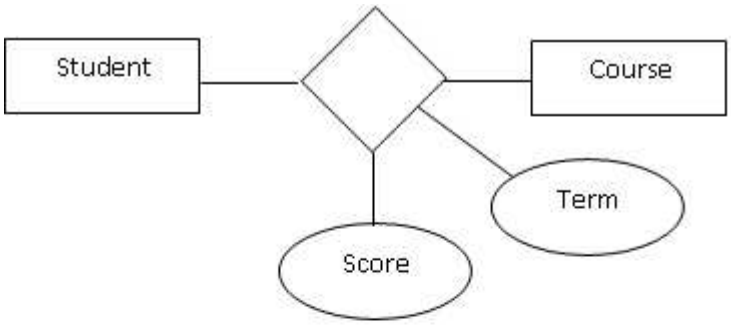
شکل ۶-۲۲: تطبیق و پیاده‌سازی قاعدهٔ چهارم

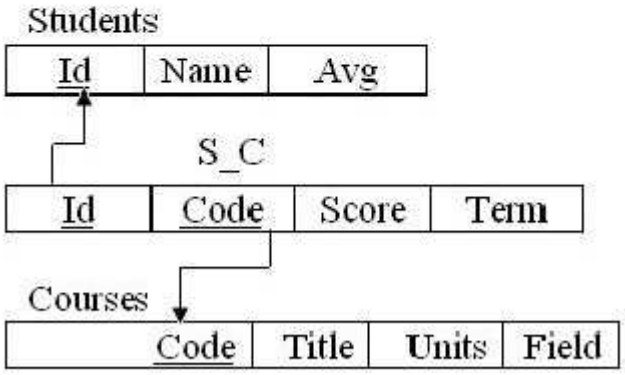
قاعدهٔ پنجم: برای یک رابطهٔ درجه دو با کاردینالیتی (یک به چند)، نهاد طرف یک تبدیل به یک جدول می‌شود و نهاد طرف چند تبدیل به یک جدول با کلید خارجی متناظر با کلید اصلی جدول دیگر می‌گردد؛ به عنوان مثال، برای نهادهای کارمند و پروژه، به شرط این‌که بر روی هر پروژه فقط یک کارمند کار کند و هر کارمند بتواند چند پروژه را انجام دهد، خواهیم داشت:



شکل ۶-۲۳: تطبیق و پیاده‌سازی قاعدهٔ پنجم

قاعدهٔ ششم: در رابطهٔ درجه دو با کاردینالیتی (چند به چند، هر کدام از نهادها به صورت یک جدول پیاده‌سازی شده و خود رابطه نیز به صورت یک جدول جداگانه ایجاد می‌شود که دو کلید خارجی متناظر با کلیدهای اصلی در دو جدول نهاد دارد. ضمن این‌که ترکیب کلیدهای خارجی در جدول رابطه می‌تواند کلید اصلی آن باشد.

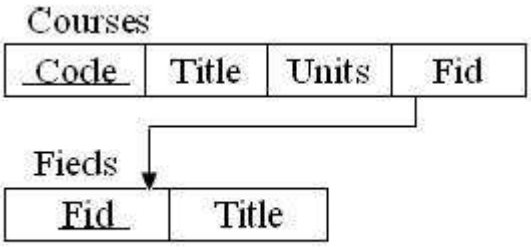




شکل ۶-۲۴: تطبیق و پیاده‌سازی قاعدهٔ ششم

مثال: رابطهٔ محصل با درس، هر محصل می‌تواند چند درس را انتخاب کرده و هر درس می‌تواند توسط چند محصل انتخاب شود.

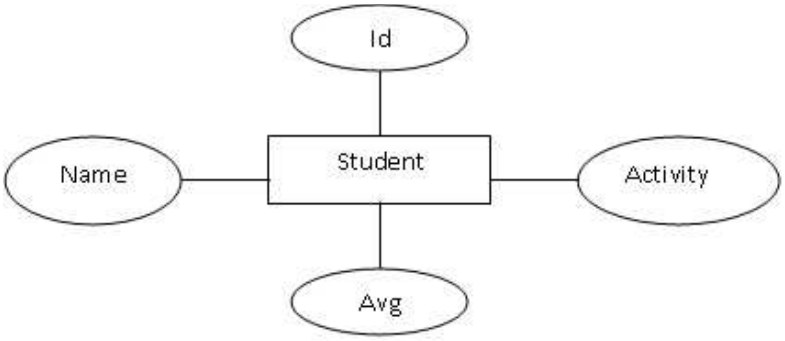
قاعدهٔ هفتم: برای صفاتی که می‌توانند مقدار آن‌ها از بین چند مقدار معین انتخاب شوند.

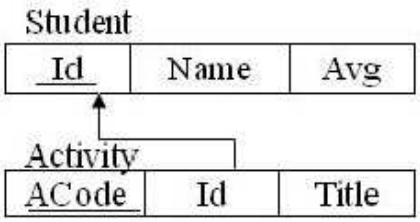


شکل ۶-۲۵: پیاده‌سازی قاعدهٔ هفتم

یک جدول برای نگهداری مقادیر قابل انتخاب ایجاد می‌کنیم؛ مثلاً جدول رشته‌ها و در جدولی که دارای آن صفت است، صفت به صورت کلید خارجی متناظر با کلید اصلی این جدول پیاده‌سازی می‌شود.

قاعدهٔ هشتم: برای صفات چندمقداری مثل صفت فعالیت علمی دانشجو یا مدارک استاد یک جدول جداگانه برای نگهداری فعالیت‌ها و مقادیر آن صفت ایجاد می‌کنیم که دارای یک کلید خارجی مرتبط با کلید اصلی در جدول است.





شکل ۶-۲۶: تطبیق و پیاده‌سازی قاعدهٔ هشتم

مثال: جدول محصلان و فعالیت‌های آن‌ها

جدول ۶-۱۱: دانشجویان

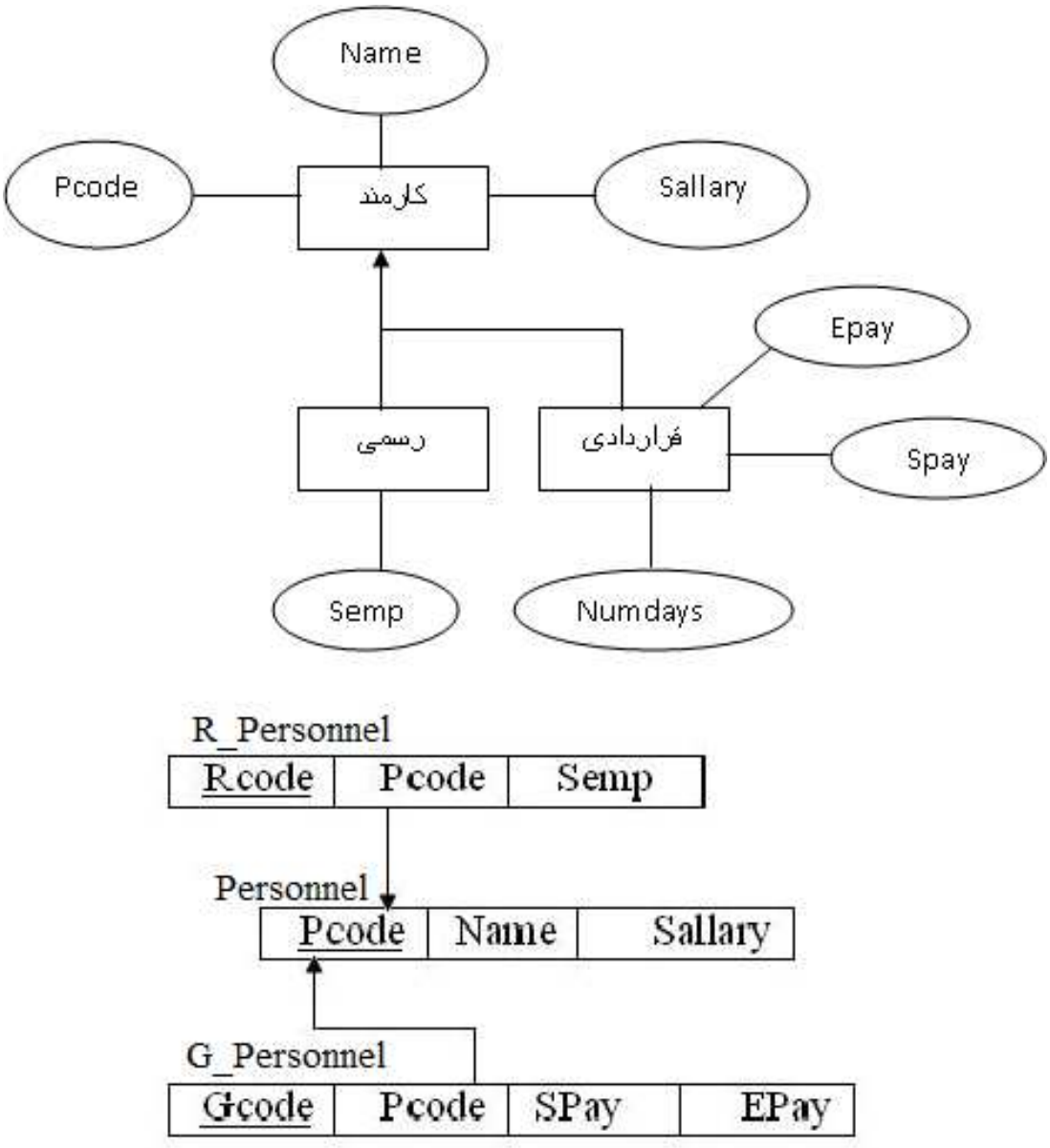
Id	Name	Avg
100	Ali	17
101	Sara	14

جدول ۶-۱۲: فعالیت‌ها

ACode	Id	Title
1	100	رتبهٔ اول مسابقات ورزشی
2	101	رتبهٔ اول مسابقات علمی
3	100	برگذاری سیمینار آموزشی

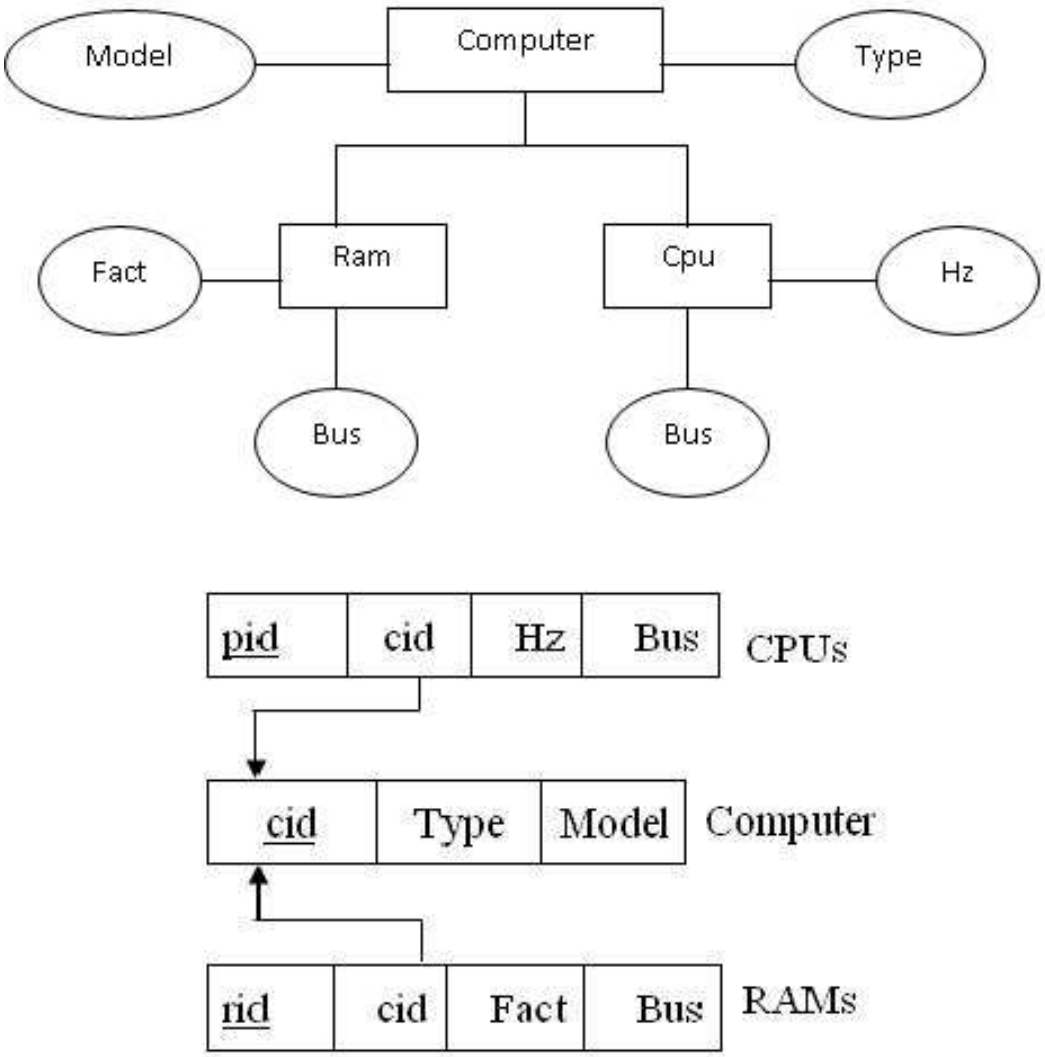
قاعدهٔ نهم: یک رابطهٔ تخصیص به صورت دو جدول است که جدول اصلی دارای اطلاعات نهاد اصلی و جدول مشتق دارای اطلاعات نهادهای خاص (مشتق) می‌باشد. در جدول مشتق یک ستون به عنوان کلید خارجی مرتبط با کلید اصلی در جدول اصلی خواهد بود و برای هر سطر در جدول مشتق باید یک سطر در جدول اصلی وجود داشته باشد.

Spay: شروع قرارداد
Epay: پایان قرارداد
Semp: شروع استخدام
Numday: تعداد روزهای کارکرد



شکل ۶-۲۷: تطبیق و پیاده‌سازی قاعدهٔ نهم

قاعدهٔ دهم: رابطهٔ تجمیع مثل رابطه بین کمپیوتر و قطعات آن است. یک جدول ایجاد می‌کنیم که دارای ستون‌هایی برای صفات نهاد اصلی است و یک جدول برای هر کدام از نهادهای جزئی می‌سازیم که دارای ستون‌هایی برای نگهداری مشخصات آن‌ها، بر علاوه یک ستون به عنوان کلید خارجی مرتبط با کلید اصلی در جدول اصلی هستند.



شکل ۶-۲۸: تطبیق و پیاده‌سازی قاعدهٔ دهم

خلاصهٔ فصل ششم

اطلاعاتی که قرار است در دیتابیس ذخیره شوند، ابتدا باید با یک دید سطح بالا از لحاظ معنایی و مفهومی مودل‌سازی گردند. در مودل ER سه مفهوم اصلی وجود دارد که عبارت‌اند از: نهاد (Entity)، صفت(Attribute) و ارتباط (Relationship). ER هم یک دیاگرام است برای مودل‌سازی معنایی اطلاعات بر پایهٔ مودل ER و در واقع سه مفهوم اساسی مودل ER، یعنی نوع جدول، صفت و ارتباط را به صورت تصویری نشان می‌دهد. قبل از هر چیز، بهتر است با این سه مفهوم آشنا شویم.

جدول مفهوم کلی شئ، پدیده و به طور کلی هر آن چیزی است که می‌خواهیم در موردش اطلاعاتی در دیتابیس داشته باشیم و با استفاده از آن، اطلاعات و شناخت خود را در موردش افزایش دهیم. صفت (Attribute) عبارت از خاصیت یا ویژگی یک نوع جدول است. هر نوع جدول مجموعه‌یی از صفات دارد. هر صفت یک نام، یک نوع و یک معنای مشخص دارد. ارتباط عبارت از تعامل و وابستگی بین دو یا بیش از دو نوع جدول است. ارتباط بین دو جدول در حقیقت ارتباط بین نمونه‌های آن دو جدول می‌باشد. هر ارتباطی دارای یک عنوان و یک مفهوم مشخص است.

بعد از ایجاد ER دیاگرام با استفاده از اصول و قواعد ذکرشده از حالت دیاگرام به حالت دیتابیس رابطه‌یی تبدیل گردد. ترتیب در تطبیق این قواعد باید مد نظر گرفته شود.

۱. هر نهاد (Entity) قوی به یک جدول تبدیل می‌شود که دارای ستون‌هایی برای هر کدام از صفات بوده و هر بخش از صفات مرکب به صورت یک ستون جداگانه پیاده‌سازی می‌گردد.

۲. هر نهاد ضعیف تبدیل به یک جدول شده و این جدول دارای یک کلید خارجی متناظر با کلید اصلی در جدول مربوط به موجودیت قوی است.

۳. برای یک رابطهٔ درجه یک (رابطه‌یی که بین یک جدول و خودش است)، یک ستون اضافی به عنوان کلید خارجی در آن ایجاد کرده که در ارتباط با کلید اصلی جدول می‌باشد.

۴. برای یک رابطهٔ درجه دو و با کاردینالیتی (یک به یک)، برای هر نهاد یک جدول ایجاد کرده و کلید اصلی آن دو جدول باهم متناظر خواهد بود.

۵. برای یک رابطهٔ درجه دو باکاردینالیتی (یک به چند)، نهاد طرف یک تبدیل به یک جدول می‌شود و نهاد طرف چند تبدیل به یک جدول با کلید خارجی متناظر با کلید اصلی جدول دیگر می‌گردد.

۶. در رابطهٔ درجه دو با کاردینالیتی (چند به چند)، هر کدام از نهادها به صورت یک جدول پیاده‌سازی شده و خود رابطه نیز به صورت یک جدول جداگانه ایجاد می‌شود که دو کلید خارجی متناظر با کلیدهای اصلی در دو جدول نهاد دارد.

۷. برای صفاتی که می‌توانند مقدار آن‌ها از بین چند مقدار معین انتخاب شوند.

۸. برای صفات چندمقداری مثل صفت فعالیت علمی دانشجو یا مدارک استاد یک جدول جداگانه برای نگهداری فعالیت‌ها و مقادیر آن صفت ایجاد می‌کنیم که دارای یک کلید خارجی مرتبط با کلید اصلی در جدول است.

۹. یک رابطهٔ تخصیص به صورت دو جدول، که جدول اصلی دارای اطلاعات نهاد اصلی و جدول مشتق دارای اطلاعات نهادهای خاص (مشتق) می‌باشد. در جدول مشتق یک ستون به عنوان کلید خارجی مرتبط با کلید اصلی در جدول اصلی خواهد بود و برای هر سطر در جدول مشتق باید یک سطر در جدول اصلی وجود داشته باشد.

۱۰. رابطهٔ تجمیع (Aggregation) مثل رابطه بین کمپیوتر و قطعات آن است. یک جدول ایجاد می‌کنیم که دارای ستون‌هایی برای صفات نهاد اصلی است و یک جدول برای هر کدام از نهادهای جزئی می‌سازیم که دارای ستون‌هایی برای نگهداری مشخصات آن‌ها برعلاوه یک ستون به عنوان کلید خارجی مرتبط با کلید اصلی در جدول اصلی هستند.

سوالات و فعالیت های فصل ششم

۱. مودل ER را با اجزای آن تشریح کنید.

۲. انواع صفت را بیان داشته و در شکل نشان دهید.

۳. انواع ارتباط را در شکل نشان دهید.

۴. ارتباط اجباری و اختیاری را تشریح نموده، در شکل واضح سازید.

۵. هدف استفاده از ارتباط IS-A را تشریح و در شکل واضح سازید.

۶. قواعد تبدیل ER به دیتابیس رابطه‌یی را بیان دارید.

۷. چند مثال برای جدول وابسته پیدا نموده، دلیل وابسته‌بودن آن را بیان دارید.(جست‌وجو در انترنت)

فعالیت‌ها

۱. تشخیص رابطه‌ها در یک دیتابیس مورد نظر؛ (انفرادی)

۲. تشخیص صفات به یک جدول مورد نظر؛ (انفرادی)

۳. دریافت نمونه‌یی از مشارکت اجباری و اختیاری در یک جدول؛ (انفرادی)

۴. ایجاد ER دیاگرام به یک دیتابیس مورد نظر؛ (گروپی)

۵. تبدیل ER به دیتابیس رابطه‌یی. (گروپی)

منابع و مأخذ

1. Ramakrishna ،Ragu .1997 .Database Management Systems.
2. Peter Rob ،Carlos Coronel .1997 .DATABASE SYSTEMS :Design ،Implementation, and Management.
3. C .J .Date .2004 .An Introduction to Database Systems.
4. D. M. Kroenke. 2005. DATABASE CONCEP