اصول طراحی پایگاه داده‌ها

برای رشته‌های: مهندسی کامپیوتر
فناوری اطلاعات – علوم کامپیوتر

چاپ دوم
ویرایش دوم

تالیف :
مهندس رمضان عباس نژاد ورزی
مهندس علیرضا عظیمی
مهندس باقررحیم‌پورکامی

برخی از عناوین مهم

بیان مفاهیم پایگاه داده طبق سرفصل وزارت علوم
بیان مسائل متنوع و حل تشریحی آن‌ها
تست‌های چهار گزینه‌ای کارشناسی ارشد سراسری و آزاد
با پاسخ تشریحی از سال ۱۳۷۲ الی ۱۳۹۱
حل تست‌های چند دوره پیام‌نور

اصول طراحی پایگاه داده‌ها

تألیف:

مهندس رمضان عباس‌نژادورزی

مهندس علی‌رضا عظیمی – مهندس باقر رحیم‌پور کامی

فن‌آوری نوین

سرشناسه	: عباس‌نژاد ورزی، رمضان، ۱۳۴۸ –
عنوان و نام پدیدآور	: اصول طراحی پایگاه داده‌ها/تألیف رمضان عباس‌نژادورزی، باقر رحیم‌پورکامی،علی‌رضا عظیمی.
مشخصات نشر	: مازندران: فن‌آوری نوین، ۱۳۹۰
مشخصات ظاهری	: ۲۴۰ص.:مصور،جدول
شابک	: 978-600-92254-7-7
وضعیت فهرست نویسی	: فیپا
موضوع	: سرور اس. کیو. ال
موضوع	: پایگاه‌های اطلاعاتی -- طراحی
شناسه افزوده	: رحیم‌پور کامی، باقر، ۱۳۶۰ –
شناسه افزوده	: عظیمی، علیرضا، – ۱۳۵۹
رده بندی کنگره	: QA۷۶/۹/ پ۲ع۱۶۶ ۱۳۸۹
رده بندی دیویی	: ۰۰۵/۷۴
شماره کتابشناسی ملی	: ۳۰۶۶۴۵۳

فن‌آوری نوین بابل، کد پستی ۴۷۱۶۷-۷۳۴۴۸ تلفن: ۳۲۲۵۶۶۸۷-۰۱۱

www.fanavarienovin.net

اصول طراحی پایگاه داده‌ها

تألیف: مهندس رمضان عباس‌نژادورزی – مهندس باقر رحیم‌پور کامی- مهندس علی‌رضا عظیمی

نوبت چاپ: چاپ دوم

سال چاپ: بهار ۱۳۹۲

شمارگان: ۱۰۰۰ جلد

قیمت: ۱۱۵۰۰ تومان

نام چاپخانه و صحافی: فر نگار رنگ

شابک: ۷- ۷- ۹۲۲۵۴ - ۶۰۰ - ۹۷۸

نشانی ناشر: بابل، چهارراه نواب، کاظم بیگی، جنب حسینیه منصور کاظم بیگی، طبقه همکف

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۲۲۰-۶۶۴۰۰۱۴۴

فهرست مطالب

فصل چهارم: دستورات SQL

کتاب‌های منتشر شده انتشارات فن‌آوری نوین	
ردیف	نام کتاب
۱	حل مسائل C (مرجع کامل)
۲	حل مسائل C++ (مرجع کامل)
۳	آموزش گام به گام برنامه نویسی بانک اطلاعات با C# (مرجع کامل)
٤	حل مسائل C# (مرجع کامل)
٥	حل مسائل پاسکال (مرجع کامل)
٦	آموزش گام به گام برنامه نویسی بانک اطلاعات با ویژوال بیسیک‌نت (مرجع کامل)
۷	آموزش گام به گام LINQ با C#
۸	تجارت الکترونیکی
۹	امنیت شبکه
۱۰	اصول طراحی پایگاه‌داده
۱۱	طراحی سیستم‌های شی‌گرا با زبان C#
۱۲	مدیریت استراتیژیک (فن آوری اطلاعات)
۱۳	کاربرد رایانه در مدیریت و حسابداری
۱٤	آموزش گام به گام برنامه نویسی به زبان C++

مقدمه

امروزه حجم زیادی از داده ذخیره، پردازش و بازیابی می‌شوند. برای جلوگیری از افزونگی داده (تکرار بی‌مورد داده‌ها)، بی‌نظمی و ایجاد سازگاری بین گزارش‌ها از پایگاه داده (بانک اطلاعات) استفاده می‌شود. از آنجائی که در سیستم‌های امروزی (به ویژه سیستم‌های تجارت الکترونیک، داده کاوی و پشتیبانی تصمیم) داده‌ها جایگاه بسیار مهمی دارند، درس پایگاه داده یکی از دروس تخصصی رشته‌های مهندسی کامپیوتر و فناوری اطلاعات در نظر گرفته شده است. به همین دلیل سعی گردیده، کتاب حاضر تدوین شود. تمام مطالب کتاب با توجه به سر فصل مصوب وزارت علوم و تحقیقات، به زبانی ساده و روان بر اساس کتب آقایان دیت، سیلورشاتس و ... بیان گردید.

کتاب حاضر در ویراست دوم قرار دارد و شامل ۵ فصل و۲ پیوست است. هر فصل کتاب از سه بخش تشکیل شده است که عبارت‌اند از:

۱. تشریح مفاهیم مربوط به آن فصل به همراه مثال‌های متعدد.

۲. بیان مسائل اضافی و حل آن‌ها.

۳. بیان تست‌های ارشد سال‌های قبل دانشگاه دولتی و آزاد و حل تشریحی آن‌ها.

در فصل اول، مفاهیم اولیه بانک اطلاعات، مزایا، معایب بانک اطلاعات و غیره بیان گردید. در فصل دوم، مدل ER و چگونگی پیاده‌سازی آن بحث و بررسی گردید، در فصل سوم، مفاهیم رابطه‌ای، جبر رابطه و حساب دامنه‌ای شرح داده شده است. در فصل چهارم، وابستگی‌های تابعی و نرمال سازی بیان گردیده است و در فصل پنجم، دستورات SQL به طور کامل بیان شده، در محیط SQL تست گردیده و خروجی آن‌ها در کتاب آمده است. در پیوست الف، مسائل تکمیلی با حل آن آمده است. درعلاوه بر این دو پیوست چند دوره سوالات دانشگاه پیام نور با جواب تشریحی به عنوان پیوست تکمیلی آمده است که این پیوست را می‌توانید به صورت فایل PDF از سایت انتشارات فناوری نوین به آدرس www.fanavarienovin.net دریافت کنید.

کتابی که در پیش رو دارید، با بهره‌گیری از سال‌ها تجربه در امر تدریس (در درس پایگاه داده) و تألیف کتب کامپیوتری تدوین گردیده است.

امیدواریم این اثر نیز مورد توجه اساتید و دانشجویان عزیز واقع شود.

در پایان از تمامی خوانندگان عزیز (اساتید و دانشجویان) تقاضا داریم، هرگونه اشکال، ابهام در متن کتاب، پیشنهادات و انتقادات‌شان را به آدرس پست الکترونیک fanavarienovin@gmail.com ارسال نمایند.

بابل، بهار ۱۳۹۲

مؤلفین

فصل ۱ بخشی از آشنایی با مفاهیم اولیه بانک اطلاعات

سیستم مدیریت بانک اطلاعات، مکانیزم نگهداری رکوردها[1] است. یعنی، بانک اطلاعات مخزنی برای نگهداری داده‌ها است که کاربران می‌توانند اعمال زیر را در آن انجام دهند:

۱. افزودن جداول جدید به بانک اطلاعات
۲. افزودن رکوردهایی به جداول بانک اطلاعات
۳. تغییر ساختار جداول
۴. حذف جدول
۵. تغییر رکوردهای بانک اطلاعات
۶. حذف رکوردهای بانک اطلاعات
۷. اجرای پرس‌وجو[2] بر روی جداول

به عبارت ساده‌تر، سیستم مدیریت بانک اطلاعات (DBMS)[3]، سیستم رایانه‌ای است که هدف آن ذخیره و بازیابی داده‌ها می‌باشد. بانک اطلاعات داده را پردازش نموده، به اطلاعات تبدیل کرده، آن‌ها را ذخیره و بازیابی می‌نماید.

۱ – ۱ . دلایل نیاز به بانک اطلاعات

قبل از این که به دلایل نیاز به بانک اطلاعاتی بپردازیم، مشکلات سیستم‌های سنتی ذخیره و بازیابی اطلاعات را می‌آموزیم.

در سیستم‌های سنتی، داده‌ها به صورت فایل‌ها بر روی دیسک سخت نگهداری می‌گردند و برنامه‌هایی نوشته می‌شوند تا داده‌ها را به فایل اضافه کنند، داده‌های موجود را ویرایش نمایند، داده‌های موجود را حذف کنند و مهم‌تر از همه این اعمال، بازیابی داده‌های موجود را انجام دهند. روش‌های پردازش فایل سنتی مشکلات زیر را دارد:

۱. **افزایش افزونگی داده،** چون در سیستم ذخیره و بازیابی اطلاعات از طریق فایل‌های سنتی، داده‌ها به مرور جمع‌آوری می‌شوند. بنابراین، ممکن است داده‌ها در فایل‌ها پخش شوند. در نتیجه، ممکن است داده‌ها در فایل‌های مختلف تکرار گردند. برخی از این تکرارها می‌توانند حذف شوند (به تکرار بی‌مورد داده‌ها

[1] .Records [2].Query [3]. Database Management System

افزونگی داده می‌گویند). یکی از ویژگی‌های بانک اطلاعات به **اشتراک گذاری** و **مجتمع کردن** داده‌ها است. به اشتراک گذاری و مجتمع کردن داده موجب کاهش افزونگی داده می‌گردد.

۲. افزایش بی‌نظمی و ناسازگاری داده‌ها، از آن جایی که در روش سنتی ذخیره‌سازی، داده‌ها در فایل‌های مجزا ذخیره می‌گردند و ممکن است داده‌ها در چند فایل تکرار شوند، تغییر داده‌های تکراری شاید موجب بی‌نظمی و ناسازگاری گردد. به عنوان مثال، فرض کنید اطلاعات آدرس دانشجو در فایل‌های عمومی، حسابداری، پرونده پزشکی و غیره تکرار گردد. با تغییر آدرس یک دانشجو، ممکن است این تغییر در بعضی فایل‌ها اعمال شود، و در برخی دیگر اعمال نگردد. این امر موجب بی‌نظمی و ناسازگاری در فایل‌ها خواهد شد. بانک اطلاعاتی با به اشتراک گذاری داده‌ها بی‌نظمی و ناسازگاری را کاهش می‌دهد. چنانچه اطلاعات دانشجو در بانک اطلاعاتی ذخیره شود، اگر بخواهید آدرس دانشجو را تغییر دهید، کافی است این تغییر را در یک نقطه اعمال کنید. چون، آدرس در بانک اطلاعات در یک نقطه نگهداری می‌شود.

۳. پیچیدگی زیاد فایل‌ها، برای مدیریت فایل‌ها باید برنامه‌هایی نوشته شوند تا اعمالی از قبیل افزودن، ویرایش، حذف و بازیابی رکوردها را انجام دهند. نوشتن چنین برنامه‌هایی با توجه به نوع دست‌یابی به فایل پیچیده خواهد شد. فرض کنید اطلاعات در فایل ترتیبی ذخیره شده است و بخواهید رکورد خاصی را حذف کنید. برای این منظور، باید فرآیند زیر را انجام دهید:

۱. فایلی که رکوردی از آن را می‌خواهید حذف کنید، فقط خواندنی باز نمایید.

۲. فایل جدیدی ایجاد کنید(فایل دوم) و آن را در حالت نوشتنی باز کنید.

۳. تمام رکوردها به جزء رکوردی که باید حذف شود، از فایل اول را خوانده، در فایل دوم بنویسید.

۴. فایل اول را حذف کنید.

۵. فایل دوم را به نام فایل اول تغییر نام دهید.

این فرآیندها برای به روز رسانی و اضافه کردن داده‌ها در فایل نیز وجود دارد. بانک اطلاعاتی، **نرم‌افزار سیستم مدیریت بانک اطلاعات (DBMS)**[2] را ارائه کرده است. این نرم‌افزار مدیریت بر داده را آسان و سریع نموده است.

افزایش مشکلات جامعیت داده، محیط‌های پردازش فایل سنتی، اجازه تعریف قیود جامعیت را نمی‌دهند. به عنوان مثال، تست نمی‌کند نمره خارج از بازه 0 تا 20 نباشد یا بررسی نمی‌کند مانده حسابی منفی نشود.

---

[2] .Database Management System

برای تعریف هر یک از این قیود باید کد برنامه تغییر یابد. اگر قید جامعیت جدیدی بخواهید تعریف کنید باید این قید را به کد برنامه اضافه نمایید. انجام این کار مشکل می‌باشد. به ویژه، وقتی قیود جامعیت باید طوری اعمال گردند تا بر روی چندین داده در فایل‌های مختلف اعمال شوند، مشکل چند برابر خواهد شد. در نرم‌افزار DBMS، مدیر بانک اطلاعاتی به راحتی می‌تواند قیود جامعیت را تعریف نماید.

وجود مشکلات اتمیک، هر لحظه ممکن است سیستم رایانه‌ای خراب شود. از طرف دیگر، برخی از کارها به هم وابسته‌اند. به عنوان مثال، هنگام انتقال پول از حساب A به حساب B اعمال زیر باید انجام شود:

۱. برداشت پول از حساب A ۲. واریز پول به حساب B

فرض کنید پس از برداشت پول از حساب A سیستم ATM خراب شود. اکنون، پول از حساب A برداشت شده است و به حساب B واریز نگردید. این خرابی موجب ناسازگاری در داده می‌شود. بدیهی است که برای سازگاری پایگاه داده باید هم برداشت از حساب A و هم واریز به حساب B انجام گردد یا هیچ کدام از آن‌ها انجام نگردند. در نگهداری داده به صورت سنتی، تضمین یکپارچگی کار بسیار مشکلی است. امّا، سیستم‌های مدیریت بانک اطلاعات از تراکنش پشتیبانی می کنند. یکی از خواص تراکنش یکپارچگی است. در ادامه با مفهوم تراکنش و خواص آن آشنا خواهید شد.

وجود مشکلات دست‌یابی هم‌زمان، در برخی از محیط‌ها به ویژه اینترنت روزانه چندین میلیون خریدار به داده‌های فروشگاه‌های اینترنتی مراجعه می‌نمایند. در این مراجعات، ممکن است هم‌زمانی برای دست‌یابی به اطلاعات رخ دهد. این هم‌زمانی موجب ناسازگاری داده خواهد شد. به عنوان مثال، فرض کنید دو کاربر بخواهند به ترتیب مبالغ 5000000 و 2000000 ریال را از حساب A که دارای مانده 20000000 ریال است، برداشت کنند. در سیستم پردازش فایل سنتی، پس از برداشت‌ها از حساب A مانده حساب یکی از مقادیر 15000000 یا 18000000 ریال خواهد شد (بستگی دارد کدام یک ابتدا به کارش خاتمه دهد). در صورتی که پس از اجرای این برداشت‌ها مبلغ واقعی مانده حساب باید 13000000 ریال باشد. سیستم مدیریت بانک اطلاعات برای جلوگیری چنین ناسازگاری از **پی‌درپی پذیری تراکنش** یا **قفل گذاری** استفاده می کند.

وجود مشکلات تفکیک و جداسازی داده‌ها، چون داده‌ها در فایل‌ها به مرور زمان جمع می‌گردند و از طرف دیگر، برنامه‌نویسان مختلفی برنامه‌های کار با فایل‌ها را نوشته‌اند، ممکن است فایل‌های ایجاد شده دارای ساختار متفاوتی باشند. نوشتن برنامه‌های جدید که بتواند داده را از ساختارهای متفاوت تفکیک نماید، کار

بسیار پیچیده‌ای است. در سیستم مدیریت بانک اطلاعات، داده‌ها با یک ساختار ذخیره می‌گردند. بنابراین، تفکیک داده‌ها آسان‌تر خواهد شد.

وجود مشکلات امنیتی، همه کاربران نباید به تمام داده دسترسی داشته باشند (هر کاربر باید به بخشی از داده دسترسی داشته باشد). به عنوان مثال، در یک سیستم دانشگاه، استاد باید فقط به اطلاعات دانشجویان خودش برای همان درسی که ارائه کرده است دسترسی داشته باشد، واحد مالی باید به اطلاعات مالی و آموزش باید به اطلاعات آموزشی دسترسی داشته باشد. ایجاد این دسترسی‌ها در سیستم‌های پردازش فایل سنتی دشوار است. در نرم‌افزار DBMS با تعریف سطح خارجی، این دسترسی‌ها را به سادگی می‌توان تعریف و اعمال نمود. از طرف دیگر، داده‌ها می‌توانند در مقابل دست‌یابی غیر قانونی و غیر مجاز حفظ شوند. زیرا، اطلاعات در یک نقطه نگهداری می‌گردند.

عدم فشرده سازی اطلاعات، چون داده‌های بانک اطلاعاتی دارای ساختار هستند، نیازی به نگهداری فایل‌های متنی حجیم نیست و از ورود داده‌های بدون ساختار جلوگیری می‌کند. این امر موجب فشرده‌سازی اطلاعات می‌گردد.

دسترسی کندتر، در هر زمان اطلاعات دقیق و به هنگام شده بازیابی می‌شوند. زیرا، اطلاعات به صورت مجتمع و یک‌پارچه نگهداری می‌گردند.

نیاز به بودجه کم‌تر، خیلی از یکنواختی‌ها در نگهداری فایل‌ها به روش دستی و سنتی که به فضای زیادی نیاز دارند، حذف خواهند شد و دیگر نیاز به ساختمان‌های بزرگ برای کارمندان جهت نگهداری و پردازش اطلاعات نمی‌باشد.

۲ ـ ۱. اجزای تشکیل دهنده بانک اطلاعاتی

هر بانک اطلاعاتی از چهار جزء تشکیل می‌شود که عبارت‌اند از:

۱. سخت‌افزار ۲. داده ۳. کاربران ۴. نرم‌افزار

۱ ـ ۲ ـ ۱. *سخت‌افزار*

سخت‌افزار مورد نیاز بانک اطلاعاتی را می‌توان به بخش‌های زیر تقسیم کرد:

۱. **سخت‌افزار پردازشگر مرکزی و حافظه اصلی**، این سخت‌افزارها برای اجرای نرم‌افزار بانک اطلاعاتی مورد استفاده قرار می‌گیرند.

۲. **محیط ذخیره‌سازی داده،** دیسک‌های جانبی از قبیل دیسک سخت، نوارها و دستگاه‌های I/O نظیر درایوها، کنترل‌گرها و ابزارهای دیگر برای ذخیره‌سازی و بازیابی داده‌ها به کار می‌روند.

۳. **سخت‌افزارهای ارتباطی،** دستگاه‌های نظیر هاب‌ها، سوئیچ‌ها، کابل‌ها، فیبر نوری و غیره هستند که برای انتقال داده به کار می‌روند.

بحث درباره اجزاء سخت‌افزاری از مقوله این کتاب خارج است. زیرا، حوزه‌های سخت‌افزاری و ارتباطی وسیع و گسترده هستند. برای هر یک از این حوزه‌ها نیاز به کتاب جداگانه‌ای می‌باشد.

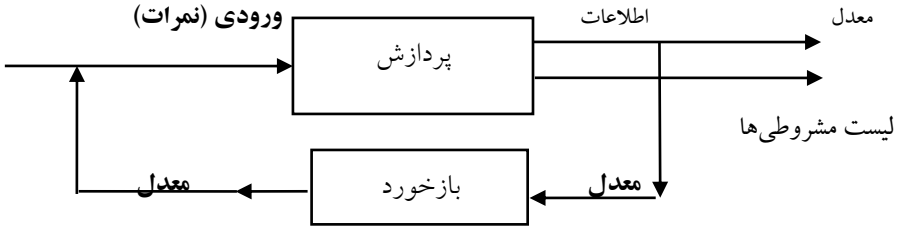
۲ ـ ۲ ـ ۱ . داده‌ها

یکی از اجزا بسیار مهم بانک اطلاعاتی داده است. هر سیستم از سه بخش تشکیل می‌گردد (شکل ۱ – ۱). این سه بخش عبارت‌اند از:

۱. **داده**[۳]، هر چیزی که وارد سیستم می‌گردد و توسط انسان قابل فهم نیست، مانند شکل ۱ – ۱، نمرات و معدل ورودی سیستم هستند.

۲. **اطلاعات**[۲]، هر داده‌ای که روی آن پردازش انجام شده و از سیستم خارج می‌شود که توسط انسان قابل درک و فهم است. به عنوان مثال، در شکل ۱ – ۱ معدل و لیست مشروطی‌ها اطلاعات هستند.

۳. **پردازش**[۳]، فرآیندی که داده را به اطلاعات تبدیل می‌کند، **پردازش** نام دارد.



شکل ۱ – ۱ بخش‌های یک سیستم.

همان‌طور، که می‌دانید سیستم‌های رایانه‌ی می‌توانند **تک کار بره**[٤] یا **چند کاربره**[۲] باشند. در سیستم تک کار بره، در یک لحظه فقط یک کاربر می‌تواند از سیستم استفاده کند. امّا، در سیستم‌های چند کار بره، در یک زمان چند کاربر می‌توانند از سیستم استفاده کنند. در سیستم مدیریت بانک اطلاعاتی از رویکرد چند کار بره

---

[3] .Data [2].Information [3].Process
[4].Single User [2] .Multi User [3] .Users [4].End users [5].Rapid Application Development

استفاده می‌شود. لذا، در یک لحظه چند کاربر می‌توانند بـه داده‌هـا دسترسـی داشـته باشـند. بنـابراین، داده بایـد **مجتمع** باشد و به **اشتراک** گذاشته شود. مجتمع کردن و به اشتراک گذاری داده مزایای زیر را دارد:

۱. **بی نظمی و ناسازگاری را کاهش می‌دهـد،** چـون داده‌هـا در یـک نقطـه جمـع‌آوری مـی‌شـوند (پخـش نمی‌شوند)، تغییر یک فیلد در یک مکان موجب تغییر این داده در کلیه بخش‌های بانک اطلاعات خواهد شد. بنابراین، موجب کاهش بی‌نظمی و ناسازگاری داده می‌گردد.

۲. **افزونگی داده را کاهش می‌دهد،** از آن‌جائی که داده‌ها به اشتراک گذاشته مـی‌شـوند، نیـازی بـه تکـرار بی‌مورد داده‌ها در جداول مختلف نمی‌باشد. بنابراین، افزونگی نیز کاهش می‌یابد.

امّا، وقتی که داده‌ها به اشتراک گذاشته می‌شوند باید مسائل و مشکلات زیر را حل گردند:

۱. **امنیت،** وقتی داده‌ها به اشتراک گذاشته می‌شوند باید تعیین گردد کـدام کـاربر بـا چـه مجوزهـایی بـه کدام بخش از داده در چه زمانی دسترسی داشته باشد. برای این منظور می‌توان **سطوح دست‌یابی** را برای کاربران تعریف نمود.

۲. **کنترل هم‌روندی (هم‌زمانی)**، همان‌طور که بیان گردید، هم‌زمانی دسترسی بـه داده‌هـا ممکـن اسـت ناسازگاری‌های در داده ایجاد کند. به عنوان مثال، دو کاربر هم‌زمان بخواهند مانده حساب فردی را تغییـر دهند، مشخص نیست کدام یک از تغییرات باید اعمال گردد. DBMS، برای رفع این مشکل تراکنش‌ها را **پی‌درپی** انجام می‌دهد (تا یکی تمام نگردید، دیگری را شـروع نمـی‌کنـد) یـا از **قفـل گـذاری** اسـتفاده می‌کند. یعنی، داده‌هایی را که به آن‌ها می‌خواهد دست‌یابی داشـته باشـد، قفـل کـرده تـا زمـانی کـه کـار تراکنش با آن داده خاتمه نیافت، قفل را از دست نمی‌دهد. بنابراین، با این دو روش می‌تـوان هـم‌زمـانی و هم‌روندی دست‌یابی به داده‌ها را کنترل کرد.

۱۲ – ۱ . پرسش‌های چهارگزینه‌ای

۱. کدام یک از موارد زیر بیانگر اخـتلاف بـین پایگـاه داده‌هـا و پایگـاه بصـیرت (Knowledge Base) می‌باشد (ارشد- آزاد ۷۲).

الف: هر دو پایگاه از یک نوع بوده و فرقی با هم ندارند.

ب: پایگاه بصیرت دینامیک ولی پایگاه داده‌ها ایستا (Static) می‌باشد.

ج: پایگاه داده اغلب با زبان SQL و پایگاه بصرت اغلب با زبان QUEL کد می‌شود.

د: هیچ‌کدام

۲. مزایای یک پایگاه داده (Database) نسبت به فایل‌های متعارف چیست (ارشد- آزاد ۷۲)؟

الف: ب و د

ب: کنترل حساب شده مقدار افزونگی در پایگاه داده‌ها

ج: اطمینان از صحت داده‌ها (DataValidation) کمتر مورد نیاز خواهد بود.

د: دست‌یابی مشترک به داده‌ها

۳. دو مرحله از مراحل طراحی بانک اطلاعاتی عبارت‌اند از: طراحی ادراکی (Conceptual Design) و طراحی منطقی (Logical Design). این دو چه تفاوت اساسی با هم دارند (ارشد - دولتی ۷۵)؟

الف: طراحی ادراکی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد. ولی، طراحی منطقی به مدل خاصی بستگی ندارد.

ب: طراحی ادراکی منطقی است و پس از آن انجام می‌شود.

ج: طراحی منطقی به صورت کلی به سیستم می‌نگرد و با روش‌های مانند ER انجام می‌گیرد.

د: طراحی منطقی به مدل خاصی مربوط می‌شود و پس از انتخاب مدل صورت می‌گیرد. ولی، طراحی ادراکی به مدل خاصی بستگی ندارد.

۴. در یک سیستم مدیریت پایگاه داده (DBMS)، کدام یک از امکانات زیر جزء عناصر اصلی تشکیل دهنده DBMS محسوب نمی‌شوند (ارشد-دولتی ۷۲)؟

الف: امکان پردازش زبان طبیعی برای کار با پایگاه.

ب: امکان کار با داده‌ها به کمک یک DSL (Data Sublanguage).

ج: امکان تأمین جامعیت و بی نقضی (Integrity) پایگاه.

د: امکان تأمین ایمنی پایگاه.

۱۳ – ۱. پاسخ پرسش‌های چهارگزینه‌ای

۱. **گزینه (ب) صحیح است.** زیرا پایگاه داده بصیرت موجب تولید دانش می‌گردد.

۲. **گزینه (الف) صحیح است.** زیرا، از مهم‌ترین مزایای پایگاه داده کاهش افزونگی و به اشتراک‌گذاری داده‌ها است.

۳. **گزینه (الف) صحیح است.** در هنگام طراحی منطقی مدل آن نظیر، مهم نیست به چگونگی پیاده‌سازی رابطه‌ای- شبکه‌ای – سلسله مراتبی و شی گرایی فکر شود.

۴. گزینه (الف) صحیح است.

فصل ۲ بخشی از مدل‌سازی داده‌ها با استفاده از مدل ER

مدل‌سازی مفهومی، یک مرحله خیلی مهم در طراحی یک کاربرد پایگاه داده موفق است. به طور کلی، اصطلاح کاربرد پایگاه داده به یک پایگاه داده خاص و برنامه‌های مرتبط به آن اشاره دارد که پرس‌وجو از پایگاه داده و به‌روزرسانی آن را انجام می‌دهد. به عنوان مثال، کاربرد پایگاه داده در یک بانک که اطلاعات حساب مشتریان را نگهداری می‌کند، برنامه‌هایی را شامل می‌شود که به‌روزرسانی بانک اطلاعات را بر اساس ایجاد و حذف سپرده‌ها محقق می‌سازد. این برنامه‌ها، منو و فرم‌های کاربردی را برای کاربر نهایی فراهم می‌نماید (مانند تحویل‌دار بانک در مثال بانک). لذا، بخشی از کاربرد پایگاه داده به طراحی، پیاده‌سازی و تست این برنامه‌های کاربردی اختصاص داده می‌شود. در گذشته طراحی و تست برنامه‌های کاربردی بیشتر در حوزه مهندسی نرم‌افزار در نظر گرفته می‌شد(در حوزه پایگاه داده معمولاً طراحی و تست برنامه‌های کاربردی کمتر انجام می‌گردید). از آن‌جائی که بیشترین مفاهیم به کار گرفته شده در متدلوژی‌های طراحی پایگاه داده‌ها تعیین عملیات روی اشیاء آن‌ها می‌باشد و همان‌طور که متدلوژی‌های مهندسی نرم‌افزار جزئیات بیشتری از ساختار پایگاه داده‌ها را تعیین می‌کنند که برنامه‌های نرم‌افزاری قرار است از آن استفاده کنند و به آن دسترسی پیدا کنند، پر واضح است که این دو مفهوم گویا با هم مرتبط هستند.

در این فصل به ارائه یک مدل مفهومی (ادراکی) مرسوم به مدل **موجودیت- ارتباط** (ER)[5] (به عنوان یک مدل داده مفهومی سطح بالای انتزاع می‌باشد) می‌پردازیم. این مدل که توسط چن ابداع شده است، برای طراحی مفهومی کاربردهای پایگاه داده‌ها مورد استفاده قرار می‌گیرد. بسیاری از ابزارهای طراحی پایگاه داده، مدل مفهومی را به کار می‌گیرند. در این فصل نیز به تشریح مفاهیم ساختار پایگاه داده‌ای و قوانین مدل ER می‌پردازیم و چگونگی استفاده این مدل را در طراحی یک الگوریتم مفهومی برای کاربردهای پایگاه داده در قالب رسم نمودار مرسوم به نمودار ER بیان می‌نماییم. علاوه بر مدل موجودیت ارتباط ER روش‌های دیگری نظیر متدلوژی مدل‌سازی شی (مانند UML) به طور قابل ملاحظه‌ای در مهندسی و طراحی نرم‌افزار مورد توجه

---

[5]. Entity Relationship

هستند. بدیهی است برای استفاده از این متدلوژی‌ها، شناخت کافی از مفاهیم شی گرایی لازم است. اما، باید توجه نمود که روش UML عمدتاً برای استفاده در طراحی نرم‌افزار ایجاد شده است، هر چند بخشی از فرآیند طراحی نرم‌افزار، طراحی پایگاه داده‌هایی است که مورد دست‌یابی ماژول‌های نرم‌افزاری قرار می‌گیرند.

یک بخش مهم در متدلوژی‌های مدل‌سازی شی نظیر UML، **دیاگرام کلاس** نام دارد. دیاگرام کلاس در بعضی از ویژگی‌ها به مدل ER شباهت دارد.

کاربرد مدل داده مفهومی انتزاعی در طراحی پایگاه داده‌ها

شکل ۱ -۲ یک شرح خلاصه از فرآیند طراحی پایگاه داده‌ها را نشان می‌دهد. مرحله اول، جمع‌آوری نیازها و تحلیل است. در این مرحله طراح پایگاه داده برای فهم نیازمندی‌های کاربران و مستندسازی نیازهایش با کاربران پایگاه داده‌ها مصاحبه می‌کند. خروجی این مرحله در قالب مجموعه نیازمندی‌های کاربران تهیه می‌گردد. این نیازمندی‌ها در صورت امکان باید در قالب یک فرم کامل و مفصل تعیین شوند. به موازات تعیین نیازمندی‌ها، نیازهای عملیاتی باید مشخص شوند. این نیازها شامل عملیاتی (یا تراکنش‌هایی) مانند عملیات بازیابی یا به‌روزرسانی یک پایگاه داده است که قرار است روی پایگاه داده‌ها اجرا شوند. پس از تحلیل و جمع‌آوری کلیه نیازها، در مرحله بعد باید یک الگوی مفهومی برای پایگاه داده‌ها ایجاد گردد. به این مرحله، **طراحی مفهومی (ادراکی)** گفته می‌شود. الگوی مفهومی، یک شرح اجمالی از نیازهای کاربران در قالب انواع موجودیت، ارتباط بین آن‌ها و محدودیت‌های‌شان است.

بدیهی است استفاده از این مفاهیم در قالب الگوی مفهومی یک مدل داده انتزاعی را مهیا می‌کند. زیرا، این مفاهیم به جزئیاتی از پیاده‌سازی اشاره نمی‌کنند، به سادگی قابل فهم هستند و به راحتی با کاربر غیر فنی ارتباط برقرار می‌کنند.

مدل ER، در طول زمان توسعه یافته و ساختارهای جدیدی به آن اضافه گردید. امروزه، نیز مدل EER[6] یکی از ابزارهای کارآمد طراحی پایگاه داده‌ها است.

یک پایگاه داده نمونه

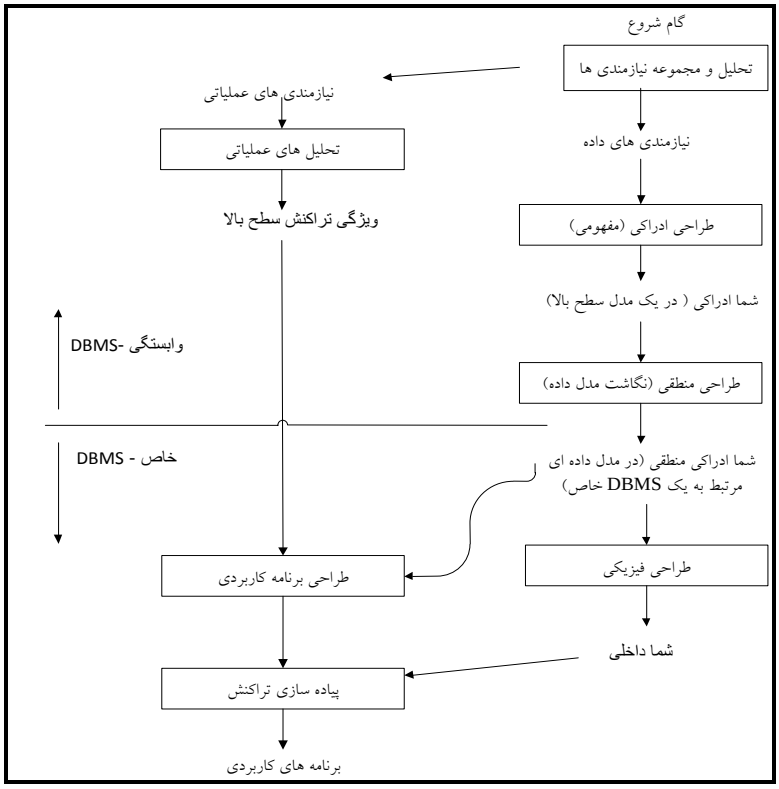
در این بخش با مثال پایگاه داده دانشگاه مفاهیم پایه‌ای مدل ER شرح داده می‌شوند. لیستی از نیازمندی‌های داده‌ای پایگاه داده دانشگاه در قالب مدل ER معرفی می‌گردند. پایگاه داده دانشگاه اطلاعاتی از

---

[6].Extended ER

قبیل دانشجویان، اساتید، نمرات، دروس و دانشکده‌ها را نگهداری می‌کند. به عنوان مثال، فرض کنید طراح پایگاه داده پس از جمع‌آوری اطلاعات مورد نیاز و تجزیه و تحلیل در فاز اول، توضیحات زیر را در مورد یک دانشگاه دریافت کرده است:

- دانشگاه از چند دانشکده تشکیل شده است. هر دانشکده یک شماره منحصر به فرد به نام کد دانشکده، یک نام یکتا و کد رئیس دانشکده دارد.
- هر دانشجو در یک دانشکده تحصیل می‌کند. دانشجو دارای یک شماره دانشجویی یکتا، نام، نام خانوادگی، کد شهر محل تحصیل، کد گروه درسی، کد دانشکده و آدرس محل سکونت است.
- هر دانشجو در هر ترم تحصیلی دروسی را انتخاب می‌نماید. درس شامل مشخصات کد درس (یکتا)، نام درس، تعداد واحد و کد نوع درس است.



شکل ۱ – ۲ یک دیاگرام داده برای نمایش فازهای اصلی طراحی پایگاه داده.

- اساتید دانشگاه به تدریس دروس می‌پردازند و هر استاد شامل مشخصاتی نظیر کد استاد، نام استاد، نام خانوادگی استاد و مرتبه علمی (مدرک) است.

به طور کلی باید کلیه اطلاعات مربوط به اساتید و دانشجو به همراه اطلاعات مربوط به دروس مرتبط به آن‌ها در هر ترم نگهداری شود.

شکل ۱ – ۲، چگونگی ایجاد یک طرح برای پایگاه داده‌ها به وسیله نمودار ER را نشان می‌دهد. در ادامه فرآیند گام به گام ایجاد یک مدل ER برای یک کاربرد نمونه از پایگاه داده‌ها (در این‌جا دانشگاه) تشریح می‌شود.

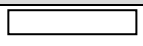
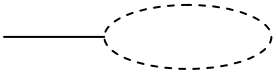
۱ – ۲. مدل ER

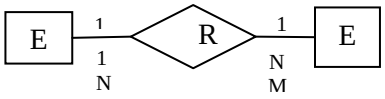
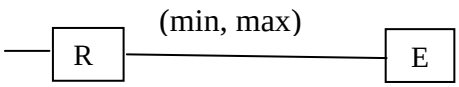
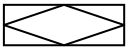
در مدل ER هر پایگاه داده، در ساده‌ترین شکل خود دارای بخش‌های زیر است:

الف) موجودیت (پدیده) ب) صفت (ویژگی پدیده) ج) ارتباط

نمادهای رسم ER و EER به طور خلاصه در جدول ۱ – ۲ آمده‌اند. در ادامه، هر یک از این نمادها و مفهوم آن‌ها را به طور کامل می‌آموزیم.

جدول ۱ – ۲ نمادهای ترسیم نمودار ER و EER.

مفهوم	نماد
موجودیت	
موجودیت ضعیف	
ارتباط	
ارتباط موجودیت ضعیف	
صفت	
صفت پیچیده	
صفت کلیدی	
صفت چند مقداری	
صفت مرکب	
صفت مشتق	

چند ارتباط R به صورت 1 : 1، 1 : N و N : M	
حد مشارکت در ارتباط R	
تخصیص صفت به ارتباط	

۱ – ۱ – ۲ . موجودیت

موجودیت[۷] هر چیزی در دنیای واقعی یک وجود مستقل دارد. پایگاه داده‌ها معمولاً شامل مجموعه‌ای از موجودیت‌های مشابه هستند. به عنوان مثال، یک دانشکده صدها دانشجو دارد. اطلاعات دانشجویان مشابه هستند. موجودیت‌های دانشجو صفات مشابهی را به اشتراک می‌گذارند. در صورتی که هر کدام مقادیر مرتبط به خودشان را برای هر صفت دارند. یک نوع موجودیت[۲] به صورت **مجموعه موجودیت**[۳] که صفات مشابهی دارند، تعریف می‌گردد. هر نوع موجودیت در پایگاه داده با نام و صفاتش توصیف می‌گردد. شکل ۲ – ۲، دو نوع موجودیت دانشجو و دانشکده را با نام و مجموعه صفاتشان نشان می‌دهد.

مجموعه همه موجودیت‌های یک نوع موجودیت خاص در یک پایگاه داده، **مجموعه موجودیت** نام دارد. معمولاً، مجموعه موجودیت به یک نام مشخص در قالب یک نوع موجودیت اشاره دارد. در مدل ER، **موجودیت** را با یک **مستطیل** و یک نام که درونش نوشته می‌شود، نشان می‌دهند. شکل۳ – ۲، نوع

دانشجو (شماره دانشجویی– معدل- نام)
S1. (۸۵٤۱۳۱۲۱- ۱٦,٤٥- احمد زیادی)
S2. (۸٤۳۱٥۱۱۰- ۱٤,۳٦- علی معصومی)
...

دانشکده (نام رئیس– شهر- نام)
C1. (دکتررضائی- تهران- فنی مهندسی)
C2. (دکتررضائی- تهران- فنی مهندسی)
...

شکل ۲-۲ دو نوع موجودیت، دانشجو و دانشکده و تعدادی موجودیت‌های عضو در هر کدام.

موجودیت‌های موجود در یک پایگاه داده دانشگاه را نشان می‌دهد. همان‌طور که در این شکل می‌بینید.

---

[7].Entity [2] .Entity Type [3].Entity set

۱ – ۴ – ۲ . قواعد تبدیل نمودار ER به جدول

بعد از این که نمودار ER را رسم کردیم، برای استفاده در بانک اطلاعاتی رابطه‌ای باید آن‌ها را به جدول تبدیل کرده تا بتوانند اطلاعات را ذخیره و بازیابی نمایید. همان طور که بیان گردید، نمودار ER، از مجموعه‌ای از موجودیت‌ها و ارتباط بین آن‌ها تشکیل می‌شود. در واقع، هر یک از موجودیت‌ها و رابطه‌ها در نمودار ER را می‌توان به یک جدول تبدیل کرد. این جدول یک نام و چند ستون دارد. نام جدول می‌تواند همان نام موجودیت یا رابطه باشد. اما، ستون‌های جدول، همان نام صفات موجودیت یا رابطه می‌باشند. هر جدول تعدادی سطر دارد. کلید موجودیت یا سطر، یکتا بودن رکوردهای جدول را تضمین می‌کند. به عنوان مثال، در ER مربوط به دانشجویان، نام جداول از نام موجودیت‌ها بدست می‌آیند. یعنی، این ER به چند جدول دانشجو، استاد، درس، نمره و غیره تبدیل می‌شود. نام فیلدها نیز از صفات موجودیت در ER بدست می‌آیند. بنابراین جدول دانشجو، دارای فیلدهای شماره دانشجویی، نام، نام خانوادگی و ... می‌باشد که فیلد شماره دانشجویی یکتا بودن رکوردهای آن را تضمین می‌کند.

۲ – ۴ – ۲. تبدیل موجودیت‌های قوی به جدول

در نمودار ER موجودیت‌هایی قوی هستند که دارای صفت کلید باشند. هر موجودیت قوی در نمودار ER به یک جدول تبدیل خواهد شد. صفات موجودیت‌های قوی به فیلدهای جدول تبدیل خواهند شد. در تبدیل موجودیت قوی به جدول نباید صفات مشتق را به فیلد تبدیل کنید. چون، صفات مشتق از صفات دیگر موجودیت بدست می‌آیند. صفت کلید اصلی موجودیت قوی در ER به کلید جدول تبدیل خواهد شد.

۳ – ۴ – ۲. تبدیل موجودیت‌های ضعیف به جدول

اگر A یک موجودیت ضعیف و وابسته به موجودیت قوی B باشد، کلید اصلی موجودیت B به کلید اصلی موجودیت A اضافه می‌گردد. بنابراین، موجودیت ضعیف نیز به یک جدول مجزا تبدیل خواهد شد. صفات موجودیت ضعیف به علاوه صفت کلید موجود قوی که موجودیت ضعیف به آن وابسته است به عنوان فیلدهای این جدول در نظر گرفته می‌شوند و کلید اصلی جدولی که از موجودیت ضعیف ایجاد می‌شود، برابر با ترکیب کلید اصلی موجودیت قوی وابسته به موجودیت ضعیف و کلید اصلی موجودیت ضعیف می‌باشد.

۴ – ۴ – ۲. تبدیل رابطه‌ها به جدول

همان‌طور که بیان گردید، در نمودار ER برخی از رابطه‌ها می‌توانند دارای صفات باشند. رابطه‌هایی که دارای صفت هستند را می‌توان به جداول تبدیل کرد (رابطه‌هایی که صفات ندارند، به جدول تبدیل نمی‌شوند).

پس، رابطه‌هایی که دارای صفت هستند، صفات رابطه، فیلدهای جدول را تشکیل می‌دهند و کلید رابطه، کلید اصلی جدول می‌باشد.

۵ – ۴ – ۲. تبدیل خواص چند مقداری به جدول

یکی از مواردی که در تبدیل ER به جداول باید در نظر گرفته شود، این است که هر یک از صفات چند مقداری به یک جدول جدید تبدیل خواهند شد. این جداول دو فیلد دارند. یکی از فیلدها همان نام صفت چند مقداری می‌باشد و فیلد دیگر، کلید اصلی خواهد بود. در فصل پنجم، تبدیل صفات چند مقداری به جدول را می‌بینید.

۶ – ۴ – ۲. تبدیل موجودیت‌های تعمیم یافته به جدول

همان‌طور که بیان گردید، موجودیت‌های تعمیم یافته می‌توانند صفات خودشان را از موجودیت‌های سطح بالاتر به ارث می‌برند. بنابراین، در این موجودیت‌ها سلسله مراتب سطوح وجود دارد. در این سلسله مراتب، موجودیت‌های سطح بالا تر به یک جدول مجزا تبدیل می‌شوند که صفات آن‌ها، فیلدهای جدول را تشکیل می‌دهند و کلید اصلی موجودیت سطح بالاتر، فیلد کلید اصلی جدول می‌باشد. از طرف دیگر، موجودیت‌های سطح پایین‌تر نیز به جداول مجزا تبدیل خواهند شد که صفات آن‌ها به علاوه صفت کلید موجودیت سطح بالاتر، فیلدهای جدول را تشکیل می‌دهند و کلید اصلی این جدول همان صفت کلید موجودیت سطح پایین‌تر است. اما، فیلد کلید خارجی آن فیلد کلید اصلی که از جدول موجودیت سطح بالاتر آمده است، می‌باشد. در این صورت ممکن است در جداول سطوح بالاتر رکوردی موجود باشد که در هیچ یک از جداول سطح پایین-تر نی آمده باشد. روش دیگر این است که جداول سطوح پایین را ایجاد کرده، فیلدهای جدول سطح بالا را به جداول سطح پایین اضافه می‌کنیم. اگر مقدار فیلدهای جدول سطح بالا کم باشد، این روش مناسب‌تر خواهد بود.

۵ – ۲. مسائل حل شده

مثال ۱ – ۲. بانک اطلاعات مربوط به شرکت بیمه اتومبیل را در نظر بگیرید. موجودیت‌ها، صفات، ارتباط‌ها و کلیدهای موبوط به موجودیت‌های این بانک اطلاعات را تعیین کرده، نمودار ER آن را رسم کنید.

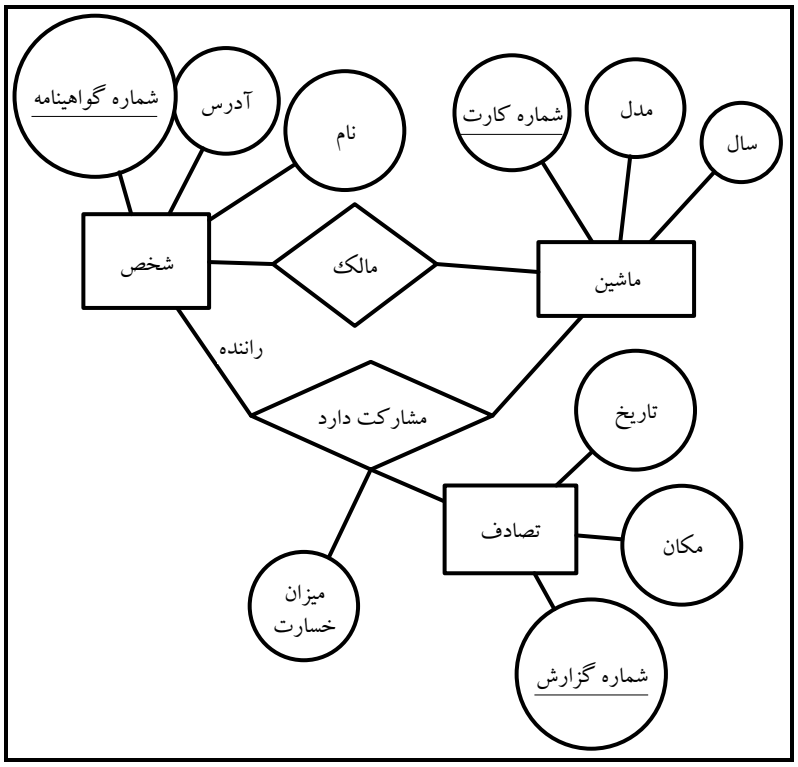
موجودیت‌های این بانک عبارت‌اند از:

۱. **شخص**، دارای صفات نام، شماره گواهی نامه و آدرس است که صفت **شماره گواهی نامه** کلید آن است.

۲. **ماشین،** دارای صفات شماره کارت ماشین، مدل و سال است که **شماره کارت ماشین** کلید آن است.

۳. **تصادف،** دارای صفات شماره گزارش، مکان و تاریخ است که **شماره گزارش** کلید آن است.

نمودار ER شرکت بیمه اتومبیل را در شکل ۱۹ – ۲ می‌بینید.



شکل ۱۹ – ۲ نمودار ER شرکت بیمه اتومبیل.

مثال ۲ – ۲. بانک اطلاعاتی انتخاب واحد دانشگاه را در نظر بگیرید. این بانک شامل موجودیت‌های زیر است:

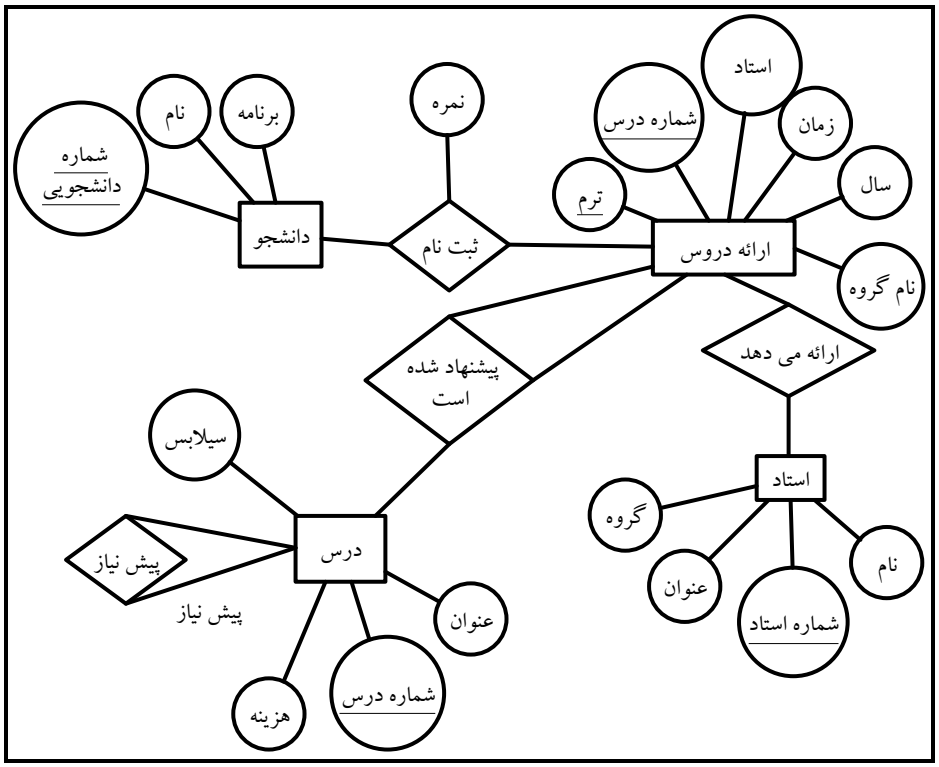
۱. **درس،** شامل صفات **شماره درس،** عنوان، هزینه، سیلابس و پیش‌نیازها است.

۲. **ارائه‌ی دروس،** شامل صفات **شماره درس،** سال، **ترم،** نام گروه، استاد و زمان است.

۳. **دانشجو،** شامل صفات **شماره دانشجویی،** نام و برنامه است.

۴. **اساتید،** شامل صفات **شماره استاد،** نام، گروه و عنوان است.

واحد ثبت نام دانشجویان، دروس و نمره را به اطلاع دانشجویان می‌رساند. نمودار ER واحد ثبت نام و امتحان را رسم کنید. حل این نمودار را در شکل ۲۰ – ۲ می‌بینید.



شکل ۲۰ – ۲ نمودار ER انتخاب واحد.

۶ – ۲. پرسش‌های چهارگزینه‌ای

۱. کدام یک از گزینه‌های زیر درست است (ارشد- سراسری ۸۳)؟

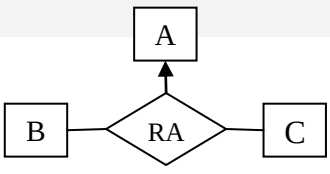
الف: وابستگی تابعی، مفهومی است که از ساختار داده‌ها به دست می‌آید، نه از معنای آن‌ها.

ب: رابطه چند به چند (M :N) در پایگاه داده رابطه‌ای، بدون نیاز به داده اضافی قابل نمایش است.

ج: استقلال داده‌ای، هدف مهمی در نظام مدیریت بانک اطلاعات (DBMS) نیست.

د: رابطه چند به چند (M : N) در پایگاه داده سلسله مراتبی، بدون نیاز به داده‌های اضافی قابل نمایش است.

۲. در نمودار موجودیت ارتباط (EER) تعریف زیر معرف کدام نوع خصیصه (Attribute) است (ارشد – آزاد ۸۵)؟



" خصیصه‌ای که در موجودیت یک و در نهایت در پایگاه داده برای آن فیلدی تعریف نشده است."

الف: چند مقداری ب: مشتق ج: ساده د: مرکب

۳. کدام گزینه از خواص رابطه (Relation) نیست (ارشد – آزاد ۸۶)؟

الف: در یک رابطه مشخص تاپل‌ها بدون ترتیب هستند.

ب: در یک رابطه مشخص تاپل‌ها تکراری وجود ندارد.

ج: در یک رابطه همه مقادیر خصیصه‌ها غیر قابل تجزیه هستند.

د: در یک رابطه مشخص، خصیصه‌ها (Attributes) دارای ترتیب هستند.

۷ – ۲. پاسخ پرسش‌های چهار گزینه‌ای

۱. گزینه (ب) صحیح است.

۲. گزینه (ب) صحیح است. چون، فیلد مشتق از حاصل فیلدهای دیگر بدست می‌آید. به عنوان مثال، معدل را می‌توان با استفاده از مجموع حاصل ضرب نمره در واحد، تقسیم بر مجموع تعداد واحدها محاسبه کرد.

۳. گزینه (د) صحیح است. ترتیب چند تایی (رکورد) و خواص (صفات یا فیلدها) در رابطه مهم نیست. فیلد کلید تضمین می‌کند که رابطه، تاپل تکراری نداشته باشد. اگر رابطه‌ای کلید نداشته باشد، تمام فیلدهای آن کلید در نظر گرفته می‌شوند.

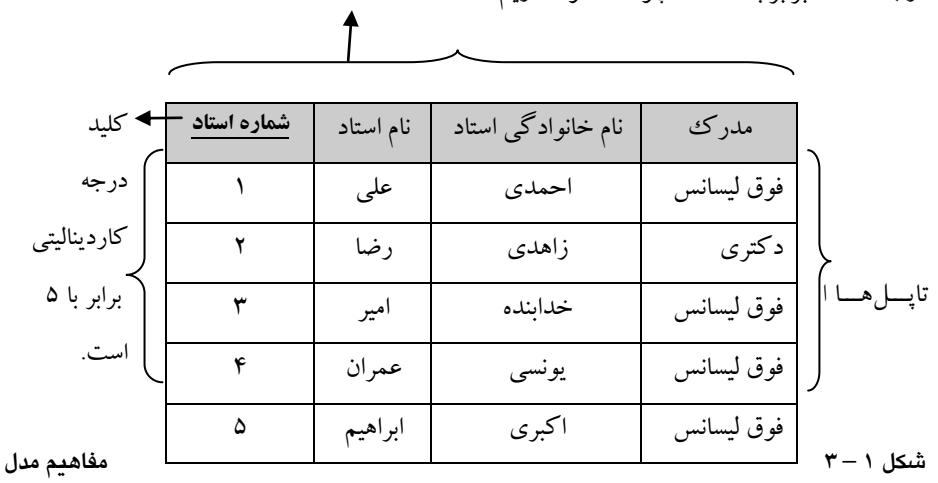
فصل ۳ بخشی از مدل رابطه‌ای و جبر رابطه-ای

در مدل رابطه‌ای[8]، داده‌ها به صورت جدول نمایش داده می‌شوند. مدل رابطه‌ای در سال ۱۹۷۰ توسط ریاضیدانی به نام Cod طراحی گردید. در بانک اطلاعاتی رابطه‌ای[2]، پایگاه داده شامل مجموعه‌ای از جداول است. در این فصل مدل رابطه‌ای، دامنه‌ای و پرس‌وجو بر روی این مدل‌ها را می‌آموزیم.

۱ – ۳ . مفاهیم اولیه مدل رابطه‌ای

در مدل رابطه‌ای مفاهیمی مانند دامنه، رابطه، تاپل، درجه رابطه، کاردینالیتی رابطه، کلید و غیره وجود دارند. این مفاهیم را در شکل ۱ – ۳ می‌بینید. مفاهیم مدل رابطه‌ای را در ادامه می‌آموزیم.

درجه صفات برابر با ۴ است. چون ۴ ستون داریم.



شکل ۱ – ۳ مفاهیم مدل رابطه‌ای در یک نگاه.

---

[8]. Relation Model [2]. Relation Database [3].Domain [4].Attribute [5].Relation

۱ ـ ۱ ـ۳. دامنه

☒ **دامنه**[۳]، مجموعه مقادیری است که صفت[۴] می‌تواند بپذیرد. به عنوان مثال، دامنه صفات مدرک (DRanking) و نام خانوادگی اساتید (DLname) در زیر آمده‌اند:

DRanking = {دکتری، فوق لیسانس، لیسانس و فوق دیپلم}

DLname = {خدابنده، عباس نژاد، جلالی، حسینی و ابوطالبی}

۱ ـ ۱ ـ ۳. رابطه

☒ **رابطه**[۵]، زیر مجموعه‌ای از ضرب دکارتی چند **دامنه** را گویند. به عنوان مثال، ضرب دکارتی دامنه‌های مدرک و نام خانوادگی اساتید در زیر آمده است:

DRanking × DLname = {(ابوطالبی، فوق دیپلم(و) حسینی، فوق دیپلم (،) جلالی، فوق دیپلم (، هاشمی)، فوق دیپلم (،) عباس نژاد، فوق دیپلم (،) خدابنده، فوق دیپلم (،... ،) عباس نژاد، دکتری (، خدابنده)، دکتری)}

زیر مجموعه‌ای از این ضرب دکارتی می‌تواند به صورت زیر باشد:

R = {(ابوطالبی، دکتری(و) حسینی، فوق لیسانس (،) جلالی، دکتری (،)هاشمی، فوق لیسانس (، عباس نژاد)، فوق لیسانس (،) خدابنده، فوق لیسانس)}

۱ ـ ۱ ـ۳. تاپل

☒ **تاپل**[۹]، هر زوج مرتب از رابطه را گویند. به عبارت دیگر، هر **سطر** یا **رکورد** جدول، تاپل نام دارد. نمونه‌ای از این زوج مرتب عبارت است از: (جلالی، دکتری)

DRanking	DLname
دکتری	ابوطالبی
فوق لیسانس	حسینی
دکتری	جلالی
فوق لیسانس	هاشمی
فوق لیسانس	عباس نژاد
فوق لیسانس	خدابنده

رابطه را به صورت جدول نیز می‌توان نمایش داد. به عنوان مثال، رابطه R به صورت مقابل نمایش داده می‌شود.

ضرب دکارتی دو دامنه، **دوتایی‌های مرتب** را ایجاد می‌کند. امّا، ضرب دکارتی سه دامنه، **سه‌تایی‌های مرتب** را تولید خواهد کرد و همین‌طور ضرب دکارتی چند دامنه، **چند تایی‌های مرتب** را تولید می‌نماید.

---

[9] .Tuple [2] . Key

۶ ـ ۱ ـ ۳ . درجه رابطه

☒ **درجه رابطه،** تعداد صفات رابطه (همان فیلدها یا ستون‌های جدول) را گویند. به عنوان مثال، درجه رابطه اساتید برابر با ۴ می‌باشد. زیرا، رابطه اساتید از ۴ صفت **کد استاد، نام استاد، نام خانوادگی و مدرک** تشکیل شده است.

۵ ـ ۱ ـ ۳ .کاردینالیتی رابطه

☒ **کاردینالیتی رابطه،** تعداد تاپل‌های رابطه(رکوردها یا ردیف‌های جدول) را گویند. جدول ۱ – ۳ مفاهیم ریاضی رابطه و معادل آن‌ها را در جدول نمایش می‌دهد.

جدول ۱ – ۳ مفاهیم ریاضی رابطه و معادل آن‌ها در جدول.

مدل رابطه‌ای	رابطه	تاپل(چند تایی)	صفت	دامنه
کاربر	جدول	ردیف	ستون	مقادیری که ستون می‌تواند بپذیرد
برنامه‌نویس	فایل	رکورد	فیلد	مقادیری که فیلد می‌تواند بپذیرد

۶ ـ ۱ ـ ۳ . کلید

کلید[۲]، شناسه‌ای در یک رابطه است که منحصر به فردی رکوردهای (تاپل‌های) رابطه را تضمین می‌کند. یعنی، این شناسه در یک رابطه مقدار تکراری نمی‌پذیرد. کلید می‌تواند یک صفت یا ترکیبی از چند صفت باشد. برخی از کلیدها عبارتند از:

۱. **شماره دانشجویی برای دانشجو،** هر دانشجو یک شماره دانشجویی منحصر به فرد دارد و شماره دانشجویی دو دانشجوی یکسان نیست.

۲. **کد کارمندی برای کارمندان،** به هر کارمند فقط یک کد کارمندی یکتا تخصیص داده می‌شود.

۳. **کد ملی برای اشخاص،** به هر فردی یک کد ملی منحصر به فرد تخصیص می‌یابد و افراد نمی‌توانند کد ملی یکسانی داشته باشند.

۴. **صفت شابک برای کتاب،** هر کتاب یک شماره شابک منحصر به فرد دارد.

کلید در رابطه دارای ویژگی‌های زیر است:

۱.مقدار کلید نمی‌تواند تکراری باشد.

۲.مقدار کلید نمی‌تواند تهی[10] باشد.

اگر برای رابطه‌ای کلید تعیین نشود(یا نتوان کلید برای آن رابطه تعیین نمود)، به طور اتوماتیک کلیه صفات آن رابطه به عنوان کلید در نظر گرفته می‌شوند. زیرا، در رابطه هیچ تاپلی (رکوردی) نمی‌تواند مقدار تکراری داشته باشد.

کلید دارای انواع مختلف است که عبارتند از:

۱. سوپر کلید[2]، هر ترکیبی از فیلدها (صفات) که در رابطه دارای خاصیت کلید باشند، **ابر کلید** نام دارند. ابر کلید تنها نوع کلیدی که می‌تواند **کمینه**[3] نباشد. به عنوان مثال، شماره کارمندی و ترکیبی از نام خانوادگی و شماره کارمندی هر دو **ابر کلید** می‌باشند.

۲. کلید کاندید[4]، هر ترکیبی از صفات که خاصیت کلید داشته باشند و کمینه باشند. به عنوان مثال، برای کارمندان صفات شماره کارمندی و کد ملی می‌توانند کلید کاندید باشند. بنابراین، هر رابطه می‌تواند چند کلید کاندید داشته باشد.

۳. کلید اولیه (اصلی)[5]، کلید کاندیدی است که مدیر بانک اطلاعاتی آن را به عنوان کلید اصلی انتخاب می‌کند. به عنوان مثال، مدیر بانک اطلاعاتی صفات شماره کارمندی و شماره دانشجویی را به ترتیب برای رابطه کارمند و دانشجو کلید اصلی انتخاب می‌کند.

کلید اولیه باید طوری انتخاب گردد که مقادیر صفت آن هرگز تغییر نکند یا به ندرت تغییر کند. به عنوان مثال، فیلد آدرس محل کار افراد، نباید به عنوان کلید اولیه در نظر گرفته شود. زیرا، ممکن است تغییر کند. مدیر بانک اطلاعاتی برای تعیین کلید اصلی دو شرط را در نظر می‌گیرد:

۱.اهمیت کلید اصلی نسبت به کلید کاندید در پاسخ گویی به پرس‌وجو.

۲. تعداد بایت‌های کلید اصلی نسبت به کلید کاندید دیگر حداقل باشد.

۴. کلید خارجی[6]، صفتی است که برای برقراری ارتباط بین دو رابطه به کار می‌رود. این صفت در یک رابطه کلید اصلی یا فرعی و در رابطه دیگر کلید خارجی است. به عنوان مثال، جدول نمره را به صورت زیر در نظر بگیرید:

---

[10] .Null [2] .Super Key [3] .Minimal [4].Candidate Key [5] .Primary Key [6].Foreign Key

جدول نمرات				
نمره	ترم	شماره درس	شماره استاد	شماره دانشجویی
15	8801	10	1200	1000
13	8802	12	1201	1010
17	8801	10	1080	1009
…	…	…	…	…

همان‌طور که در این جدول می‌بینید، شماره دانشجویی، شماره استاد، شماره درس هر یک به تنهایی یک کلید خارجی هستند.

شماره درس، کلید اصلی جدول دروس، شماره استاد، کلید اصلی جدول اساتید و شماره دانشجویی، کلید اصلی جدول دانشجویان است. در ادامه با چگونگی برقراری ارتباط بین جداول (رابطه‌ها) از طریق کلید خارجی و کلید اصلی آشنا می‌شویم.

نکته:	کلید خارجی تنها کلیدی است که می‌تواند **مقدار تکراری و تهی** (**Null**) را بپذیرد.

جداول نمونه بانک اطلاعاتی

ساده‌ترین روش جهت آموزش جبر رابطه‌ای بیان یک مثال از بانک اطلاعاتی و تعیین جدول آن می‌باشد. بهترین مثال برای آموزش جبر رابطه‌ای، بانک اطلاعاتی دانشجویان است. جداول تشکیل دهنده بانک اطلاعاتی دانشجویان در جدول ۳–۳ آمده‌اند. در ادامه کتاب، پرس‌وجوهای جبر رابطه‌ای را با این مثال و جداول بیان شده در آن می‌آموزیم. وظایف این جداول را در زیر می‌بینید:

☒ **جدول Student**، اطلاعات عمومی دانشجویان از قبیل کد دانشجویی، نام، نام خانوادگی، کد شهر محل تحصیل، کد گروه و کد دانشکده محل تحصیل را نگهداری می‌کند. کلید اولیه این جدول stNo است. زیرا، هیچ دو دانشجویی شماره دانشجویی یکسان ندارند.

☒ **جدول City**، اطلاعات شهرهای محل تحصیل دانشجویان را نگهداری می‌کند. کلید اولیه این جدول فیلد cityCode است. زیرا، کد هیچ دو شهری یکسان نیست.

☒ **جدول Group1**، اطلاعات گروه‌های آموزشی دانشجویان را نگهداری می‌نماید. کلید اولیه این جدول فیلد groupCode است. چون، هیچ دو گروهی کد یکسانی ندارند.

☒ **جدول Clg**، اطلاعات دانشکده‌ها را نگهداری می‌کند. کلید اولیه این جدول فیلد clgNo است. زیرا، هیچ دو دانشکده‌ای کد دانشکده یکسان ندارند.

☒ **جدول Course**، اطلاعات دروس را نگهداری می‌کند. کلید اولیه این جدول فیلد courseNo است. چون، هیچ دو درسی کد یکسان ندارند.

☒ **جدول Type**، اطلاعات نوع درس‌ها را نگهداری می‌کند. نوع درس می‌تواند تخصصی، پایه، عمومی و ... باشد. کلید اولیه این جدول فیلد typeCode است.

جدول ۳ – ۳ جداول بانک اطلاعات دانشجویان.

نام جدول	نام فیلد	نوع فیلد	کلید اصلی	کلید خارجی	تهی	نام فارسی
Student	stNo	۸ کاراکتر	بله	–	–	شماره دانشجویی
	Fname	۱۵ کاراکتر	–	–	–	نام
	Lname	۲۰ کاراکتر	–	–	–	نام خانوادگی
	cityCode	۳ کاراکتر	–	بله	بله	کد شهر
	groupCode	۳ کاراکتر	–	بله	بله	کد گروه
	clgNo	۳ کاراکتر	–	بله	بله	کد دانشکده
City	ctiyCode	۳ کاراکتر	بله	–	–	کد شهر
	cityName	۳۰ کاراکتر	–	–	–	نام شهر
Group1	groupCode	۳ کاراکتر	بله	–	–	کد گروه
	groupName	۳۰ کاراکتر	–	–	–	نام گروه
Clg	clgNo	۳ کاراکتر	بله	–	–	کد دانشکده
	clgName	۴۰ کاراکتر	–	–	–	نام دانشکده
	teacherCode	۶ کاراکتر	–	بله	بله	کد رئیس دانشکده
Teacher	teacherCode	۶ کاراکتر	بله	–	–	کد استاد
	Fname	۱۵ کاراکتر	–	–	–	نام استاد
	Lname	۲۰ کاراکتر	–	–	–	نام خانوادگی
	Ranking	۲۰ کاراکتر	–	–	–	مدرک استاد
Course	courseNo	۶ کاراکتر	بله	–	–	کد درس
	courseName	۴۰ کاراکتر	–	–	–	نام درس
	Unit	عددی	–	–	–	تعداد واحد
	typeCode	۲ کاراکتر	–	بله	بله	کد نوع درس
Type	typeCode	۲ کاراکتر	بله	–	–	کد نوع درس
	typeName	۳۰ کاراکتر		–	–	نام نوع درس
Score	stNo	۸ کاراکتر	بله	بله	–	شماره دانشجویی
	courseNo	۶ کاراکتر	بله	بله	–	شماره درس
	teacherCode	۶ کاراکتر	–	بله	–	کد استاد ارائه دکننده درس
	Term	۴ کاراکتر	بله	–	–	ترم تحصیلی
	Score	عددی	–	–	–	نمره

☒ **جدول Score**، اطلاعات دروس انتخاب شده و نمرات دانشجویان را نگهداری می‌نماید. کلید اولیه این جدول ترکیب فیلدهای شماره دانشجویی، کد درس و ترم است.

☒ **جدول Teacher**، اطلاعات اساتید دانشکده را نگهداری می‌کند. در این جدول کد استاد کلید اولیه است، زیرا هیچ دو استادی کد یکسانی ندارند.

اطلاعات جداول بانک اطلاعاتی در شکل ۳ – ۳ آمده است.

جدول دانشجو (Student).

کد دانشکده	کد گروه	کد شهر	نام خانوادگی	نام	شماره دانشجویی
۱۰	۱	۱	رضوانی	رضا	۸۷۸۹۱۱۰۱
۱۰	۱	۲	احمدی	احمد	۸۷۸۹۱۱۰۴
۱۲	۲	۳	علوی	زهرا	۸۷۸۹۱۱۶۶
۱۱	۲	۲	رضازاده	لیلا	۸۷۹۶۱۱۲۲
۱۱	۳	۱	کمالی	مریم	۸۷۱۲۱۷۶۶
۱۰	۲	۳	یونسی	محمد	۸۷۱۳۱۶۶۵
۱۲	۳	۳	حسین زاده	جواد	۸۷۶۶۲۲۱۰

جدول دانشکده(Clg).

کد رئیس	نام دانشکده	کد دانشکده
۱۰	فنی مهندسی	۱۰
۱۱	دانشگاه آزاد	۱۱
۱۲	دانشگاه تهران	۱۲
۱۳	علوم و فنون	۱۳

جدول گروه (Group1).

نام گروه	کد گروه
کامپیوتر	۱
حسابداری	۲
ریاضی	۳
شیمی	۴

جدول شهر(City).

نام شهر	کد شهر
بابل	۱
ساری	۲
تهران	۳
یزد	۴

جدول درس (Course).

کد نوع درس	تعداد واحد	نام درس	کد درس
۱	۴	مبانی برنامه سازی	۱۰
۱	۳	اصول حسابداری	۲۰
۲	۲	اخلاق	۳۰
۳	۳	ریاضی۱	۴۰
۲	۲	تجارت الکترونیکی	۵۰
۱	۳	پایگاه داده	۶۰

جدول نوع درس (Type).

جدول استاد (Teacher).			
مدرک	نام خانوادگی	نام	کد استاد
دکتری	ابوطالبی	رضا	۱۰
فوق لیسانس	حسنی	زهرا	۱۱
دکتری	جلالی	احمد	۱۲
فوق لیسانس	هاشمی	محمد	۱۳
فوق لیسانس	عباس نژاد	رمضان	۱۴
فوق لیسانس	خدابنده	امیر	۱۵

نام نوع درس	کد نوع درس
تخصصی	۱
عمومی	۲
پایه	۳
اختیاری	۴

جدول نمره (Score).				
نمره	ترم	شماره استاد	شماره درس	شماره دانشجویی
۱۶	۸۸۰۱	۱۰	۱۰	۸۷۸۹۱۱۰۱
۹/۵	۸۸۰۱	۱۲	۴۰	۸۷۸۹۱۱۰۱
۱۲	۸۹۰۱	۱۰	۱۰	۸۷۸۹۱۱۶۶
۸	۸۹۰۱	۱۲	۴۰	۸۷۸۹۱۱۶۶
۱۶	۸۹۰۱	۱۴	۵۰	۸۷۹۶۱۱۲۲
۱۴	۸۹۰۳	۱۴	۶۰	۸۷۶۶۲۲۱۰

شکل ۳ – ۳ اطلاعات بانک اطلاعاتی دانشگاه.

مثال ۲۲ – ۳. پرس‌وجویی که اساتید دکتری که هیچ دانشجوی را نمره زیر ۱۰ نداده‌اند، نمایش می‌دهد.

حل: ابتدا کد کل اساتید دکتری را بازیابی می کنیم و تفاضل این اساتید را با اساتیدی که حداقل یک دانشجو را زیر ۱۰ داده‌اند، بازیابی می کنیم:

$$\prod_{\text{teacherCode}}(\sigma_{\text{Ranking = '}}(\text{Teacher})) - \prod_{\text{teacherCode}}(\sigma_{\text{Score < 10}}(\text{Score})) \text{دکتری} \text{T1} \quad \leftarrow$$

سپس، ارتباط بین جدول Teacher و نتیجه این پرس‌وجو را برقرار می کنیم:

$$\text{Teacher} \ltimes \text{T1}$$

خروجی ۲۲ -۳.			
مدرک	نام خانوادگی	نام	کد استاد
دکتری	ابوطالبی	رضا	۱۰

مثال ۲۳ – ۳. پرس‌وجویی که اطلاعات دانشجویان دانشگاه آزاد که نمره بین ۱٤ تا ۱۷ اخذ کرده‌اند، را نمایش می‌دهد.

حل: ابتدا، شماره دانشجویانی که نمره بین ۱۴ تا ۱۷ اخذ کرده‌اند، را بازیابی می کنیم:

$$\prod_{\text{stNo}}(\sigma_{\text{Score} \geq 14 \wedge \text{Score} \leq 17}(\text{Score}))$$

سپس، دانشجویان، دانشگاه آزاد را بازیابی می کنیم:

$$\sigma_{\text{clgName = '}}(\text{Clg}) \text{دانشگاه آزاد} \text{Student} \quad \ltimes$$

در پایان، نتیجه این دو خروجی را با هم پیوند می‌زنیم:

$$\Pi_{stNo}(\sigma_{Score \geq 14 \wedge Score \leq 17}(Score)) \bowtie (\sigma_{clgName = \text{'دانشگاه آزاد'}}(Student \bowtie Clg))$$

خروجی ۲۳ -۳.

کد دانشکده	کد گروه	کد شهر	نام خانوادگی	نام	شماره دانشجویی
۱۱	۲	۲	رضازاده	لیلا	۸۷۹۶۱۱۲۲

مثال ۲۴– ۳. پرس‌وجویی که رواسای دانشکده که درس ۳ واحدی زیر ۱۰ داده‌اند را نمایش می‌دهد.

حل: ابتدا، رواسای دانشکده را بازیابی می‌کنیم. یعنی، ارتباط بین جدول Teacher و Clg را از طریق فیلد teacherCode برقرار می‌کنیم. چون، teacherCode در جدول Clg، کد رئیس دانشکده است، پس، داریم:

$$Teacher \bowtie Clg$$

سپس، کد اساتیدی که درس‌های ۳ واحدی را نمره زیر ۱۰ داده‌اند بازیابی می‌کنیم:

$$\Pi_{TeacherCode}(\sigma_{Score < 10}(Score) \quad \sigma_{Unit = 3}(Course))$$

در پایان، نتیجه این دو پرس‌وجو را با یکدیگر پیوند می‌زنیم:

$$(Teacher \bowtie Clg) \quad _{TeacherCode}(\sigma_{Score < 10}(Score) \quad \sigma_{Unit = 3}(Course))$$

خروجی ۲۴-۳.

مدرک	نام خانوادگی	نام	کد استاد
دکتری	جلالی	احمد	۱۲

مثال ۲۵ – ۳. پرس‌وجویی که درس‌های ۳ واحدی که نمره زیر ۱۰ نداریم را نمایش می‌دهد.

حل: ابتدا، تفاضل شماره درس‌هایی که نمره زیر ۱۰ در جدول Score ندارند و ۳ واحدی هستند را بازیابی می‌کنیم:

$$T1 \longleftarrow \Pi_{courseNo}(\sigma_{Unit = 3}(Course)) - \Pi_{courseNo}(\sigma_{Score < 10}(Score))$$

چون اطلاعات درس‌ها را می‌خواهیم، پس ارتباط نتیجه این پرس‌وجو را با جدول Course برقرار می‌کنیم:

$$Course \bowtie T1$$

خروجی ۲۵ – ۳.

کد نوع درس	تعداد واحد	نام درس	کد درس
۱	۳	اصول حسابداری	۲۰
۱	۳	پایگاه داده	۶۰

۸ – ۳ -۳. توابع تجمعی

توابع تجمعی، مجموعه‌ای از مقادیر را دریافت کرده، مقداری را برمی‌گردانند. به عنوان مثال، جدول مقابل را در نظر بگیرید. اکنون، خروجی زیر را ببینید:

Count(*)	Max(Score)	Min(Score)	Sum(Score)	Avg(Score)
6	16	9	79	13.32

این جدول، اجرای توابع تجمعی را بر روی کل جدول نشان می‌دهد. امّا، ممکن است توابع تجمعی بر اساس فیلد یا فیلدهای خاص باشد. به عنوان مثال، توابع تجمعی در این جدول بر اساس فیلد courseNo به صورت زیر است:

courseNo	Count(*)	Max(Score)	Min(Score)	Sum(Score)	Avg(Score)
10	2	14	12	26	13
11	2	16	13	29	14.5
12	2	15	9	24	12

در جبر رابطه‌ای تابع تجمعی به صورت زیر به کار می‌رود:

(نام جدول یا رابطه)(صفت) تابع تجمعی n ... , (نام صفت)تابع تجمعی ۱$\mathcal{G}$صفت n ,.., صفت۲ , صفت ۱

صفات ۱ تا n، برای گروه‌بندی به کار می‌روند. این صفات می‌توانند حذف شوند. در این صورت توابع تجمعی بر روی کل رکوردهای جدول اعمال می‌شوند.

مثال ۲۶ – ۳. پرس‌وجویی که تعداد دانشجویان را بازیابی می‌کند.

$\mathcal{G}_{\text{Count (*)}}$ (Student)

خروجی ۲۶-۳.
Count (*)
7

مثال ۲۷ – ۳. پرس‌وجویی که حداکثر و حداقل نمره دروس ۳ واحدی را نمایش می‌دهد.

$\mathcal{G}_{\text{Max (Score), Min (Score)}}$ (Score $\boxtimes_{\text{Unit = 3}}$ (Course))

خروجی ۲۷ -۳.	
Max	Min
14	8

مثال ۲۸ – ۳. پرس‌وجویی که تعداد اساتید را بر اساس مدرک آن‌ها نمایش می‌دهد.

$_{\text{Ranking}}\mathcal{G}_{\text{Count (*)}}$ (Teacher)

خروجی ۲۸ – ۳.	
مدرک	تعداد
دکتری	2
فوق لیسانس	4

مثال ۲۹– ۳. پرس‌وجویی که تعداد دانشجویان را بر اساس کد گروه و کد دانشکده نمایش می‌دهد.

$_{\text{groupCode, clgNo}}\mathcal{G}_{\text{Count (*)}}$ (Student)

خروجی ۲۹ – ۳.		
کد گروه	کد دانشگاه	تعداد
1	10	2
2	12	1
2	11	1
3	11	1
2	10	1
3	12	1

۵ – ۳. مسائل حل شده

مثال ۱ – ۳. جداول R1 و R2 مقابل را در نظر بگیرید:

R1 ∩ R2

نتیجه عبارات رابطه‌ای R1 ∪ R2، R1 ∩ R2، R1 – R2 و R1 - R2 را بدست آورید.

A	B
a	1
d	3
e	4

R2 – R1	
A	B
c	2
f	2

R1 - R2	
A	B
b	2
f	4
e	5

R1 ∪ R2	
A	B
a	1
b	2
d	3
f	4
e	5
c	2
e	4
f	2

مثال ۲ –۳. با توجه به رابطه‌های T1 و T2، پاسخ عبارات رابطه‌ای زیر را بدست آورید:

T2			
A	B	C	D
a	b	c	10
z	w	g	20
f	o	p	10

T1			
A	B	C	D
a	b	c	10
f	o	p	20
k	q	r	10

الف: $T_2 \cap T_1$ ب: $T_2 \cup T_1$ ج: T_1 Join T_2

حل الف، چون فیلدهای (ستون‌های) جدول T_1 و T_2 یکی (همسان) هستند. پس، می‌توان اشتراک آن‌ها را بدست آورد. رکوردهایی از جدول T_1 و T_2 می‌آیند که مقدار فیلدهای A، B، C و D آن‌ها برابر باشند. یعنی:

A	B	C	D
a	b	c	10

حل ب، اجتماع دو رابطه T_1 و T_2، رکوردهایی است که در رابطه T1، T_2 یا در هر دو باشند. یعنی:

A	B	C	D
a	b	c	10
f	o	p	20
k	q	r	10
z	w	g	20
f	o	p	10

حل ج، چون فیلدهای جدول T_1 و T_2 یکی هستند و T_1 Join T_2 برابر با $T_1 \cap T_2$ است. پس، داریم:

A	B	C	D
a	b	c	10

مثال ۳ – ۳. با توجه به رابطه‌های R1 و R2 نتیجه جبر رابطه‌ای R1 ÷ R2 چیست؟

R1	
stNo	courseNo
10	1
20	1
10	2
20	3
30	1
10	3
30	2
20	2
40	2

R2
courseNo
1
2
3

حل: R2 ÷ R1 فقط شماره دانشجوهایی (stNoهایی) را نمایش می‌دهد که تمام شماره درس‌های ۱، ۲ و ۳ را در جدول R1 داشته باشند. یعنی:

R1 ÷ R2 =

stNo
10
20

مثال ۴ – ۳. با توجه به رابطه‌های R1، R2 و R3 نتیجه عبارات رابطه‌ای R2 ÷ R1 و R3 ÷ R1 چیست؟

R1	
S#	P#

حل: R2 ÷ R1، برابر با

S#
S1

است.

R2
P#
P1
P2
P3
P4
P5
P6

R3
P#
P2
P3

S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P3
S4	P5

زیرا، فقط S1 دارای مقادیر P1، P2، P3، P4، P5 و P6 است.

امّـا، R3 ÷ R1 برابـر بـا

S#
S1
S4

اسـت. زیـرا، S1 و S4 مقادیر P2 و P3 را دارند.

مثال ۵ – ۳. اگر T_1 و T_2 به صورت زیر باشند، $T_2 - T_1$ را بدست آورید.

T_1	
A	B
a	1
a	2
b	1

-

T_2	
A	B
a	2
b	3

⟹

$T_1 - T_2$	
A	B
a	1
b	1

۶ – ۳ . پرسش‌های چهارگزینه‌ای

۱. اگر A و B دو رابطه دارای خصیصه‌های (Attributes) یکسان باشند، حاصل الحاق A و B کدام یک از عبارات زیر است (ارشد - کامپیوتر دولتی ۸۷)؟

الف: $B \times A$ ب: $A \cap B$ ج: $A \cup B$ د: هیچ‌کدام

۲. کدام یک از گزینه‌ها در مدل رابطه‌ای صحیح است (ارشد – مهندسی کامپیوتر ۸۶)؟

الف: ترتیب چند تایی (Tuple)های یک رابطه مهم است.

ب: ترتیب خصیصه‌های (Attributes) یک رابطه مهم است.

ج: هر رابطه دارای چند کلید خارجی (Foreign key) است.

د: هر رابطه حداقل دارای یک نامزد کلیدی (Candidate key) است.

۳. اگر رابطه R (a, b) دارای تعداد r تاپل بوده و رابطه S (a, c) دارای تعداد s تاپل باشد. آن گاه تعداد کمینه و بیشینه تاپل‌های $r \cup s$ برابر است با (ارشد – IT ۸۵).

الف: r + s, Min (r, s) ب: Max (r, s), r - s

ج: r + s , Max (r, s) د: Min (r, s) و Max (r, s)

٤. فرض کنید R1 و R2 دو رابطه باشند و R3 = R1 Union R2 کدام یک از گزاره‌های زیر صحیح است (ارشد – سراسری ۸۲)؟

الف: کلید اصلی R3 اجتماع کلیدهای اصلی R1 و R2 است.

ب: کلید اصلی R3 اجتماع تمام خصیصه‌های R1 و R2 است.

ج: کلید اصلی R3 کلیدهای اصلی R1 یا کلید اصلی R2 است.

د: کلید اصلی R3 تقاطع تمام خصیصه‌های (ستون‌های) R1 و R2 است.

۵. کدام یک از عملگرهای جبر رابطه‌ای به عنوان عملگر اولیه نیست و می‌توان آن را بر اساس عملگرهای اولیه دیگر بدست آورد (ارشد IT- آزاد ۸۵)؟

الف: Union　　ب: Times　　ج: Minus　　د: Intersect

۷ – ۳ . پاسخ پرسش‌های چهارگزینه‌ای

۱. گزینه (ب) صحیح است. از آنجای که تمام خصیصه‌های رابطه‌های A و B یکسان هستند. بنابراین، اشتراک رابطه‌های A و B ($A \cap B$) برابر با پیوند آن‌ها خواهد بود.

۲. گزینه (د) صحیح است. چون هر رابطه باید حداقل یک کلید کاندید داشته باشد.

۳. گزینه (ج) صحیح است. ممکن است همه تاپل‌های (رکوردهای) یک رابطه در رابطه دیگر تکرار شوند. پس، حداقل رکوردها برابر با Max (r, s) است. ممکن است در این دو رابطه هیچ رکورد مشترک وجود نداشته باشند. بنابراین، حداکثر تاپل‌ها برابر با $r + s$ خواهد بود.

٤. گزینه (الف) صحیح است.

۵. گزینه (د) صحیح است. زیرا، عملگر Intersect (اشتراک) با استفاده از عملگر (–) بدست می‌آید. یعنی، $(A \cap B) = A – (A – B) = B – (B - A)$.

فصل ۴

بخشی از دستورات SQL

در دهه ۱۹۷۰، نسخه اصلی زبان پرس‌وجوی ساخت یافته (SQL)[11] اولین بار توسط شرکت IBM تحت نام Seguel ارائه گردید. در سال ۱۹۸۶، موسسه استانداردسازی ملی آمریکا (ANSI) و سازمان بین‌المللی استانداردسازی (ISO) استاندارد SQL تحت نام SQL-86 را معرفی نمودند. در سال ۱۹۸۹ موسسه استانداردسازی ملی آمریکا نسخه SQL-89 را ارائه نمود. سپس، نسخه‌های SQL-92، SQL-99، SQL-2003 و SQL-2008 معرفی گردیدند. اکنون، جدیدترین نسخه SQL، SQL-2012 می‌باشد. زبان SQL از چندین بخش تشکیل شده است. در این فصل بخش‌های مختلف آن را می‌آموزیم.

مثال ۱ – ۴. پرس‌وجویی که بانک اطلاعاتی به نام Student را ایجاد می‌کند که نام فایل داده آن Student_data.mdf است. این فایل در پوشه Database درایو D قرار می‌گیرد. اندازه این فایل ۱۰۰ مگابایت و حداکثر اندازه آن ۱۵۰ مگابایت می‌باشد. نام فایل کارنامه Student_log.ldf می‌باشد که در پوشه Database درایو D به اندازه ۱۲۰ و حداکثر اندازه ۲۰۰ مگابایت ایجاد می‌شود.

```
CREATE DATABASE Student
  ON PRIMARY (NAME=Student_data, FILENAME= 'D:\Database
 \Student_data.mdf',SIZE= 100MB, MAXSIZE= 150 MB)
   LOG ON(NAME=Student_log, FILENAME='D:\Database
  \Student_log.ldf', SIZE= 120MB, MAXSIZE= 200MB )
```

۱–۳–۴. تغییر خواص بانک اطلاعاتی موجود

در بخش قبل روش ایجاد بانک اطلاعاتی را دیدید. گاهی نیاز است خواص فایل‌های بانک اطلاعاتی موجود از قبیل اندازه، حداکثر اندازه و غیره را تغییر دهید. برای این منظور می‌توانید از دستور ALTER DATABASE به صورت زیر استفاده کنید:

```
ALTER DATABASE database_name
ADD FILE (ویژگی‌های فایل داده)
```

[11] . Structured Query Language

```
MODIFY FILE  (ویژگی‌های فایل موجود)
ADD LOG FILE (ویژگی‌های فایل کارنامه)
```

، نام بانک اطلاعاتی است که می‌خواهید ویژگی‌های فایل آن را تغییر دهید.database_name

مثال ۲ – ۴. پرس‌وجویی که اندازه و حداکثر اندازه فایل Student_data از بانک اطلاعاتی Student را به ترتیب به ۱۵۰ و ۲۰۰ مگا بایت تغییر می‌دهد.

```
ALTER DATABASE  Student
 MODIFY FILE (NAME= Student_data, SIZE= 150, MAXSIZE= 200)
```

۲–۳–۴. حذف بانک اطلاعاتی موجود

همان طور که دیدید، فایل‌های داده و کارنامه بانک اطلاعاتی، فضای روی دیسک را اشغال می‌کنند. بنابراین، اگر بانک اطلاعاتی را نیاز نداشته باشید باید آن را حذف کنید تا فضای اشغال‌شده توسط آن به سیستم عامل برگردد. برای حذف بانک اطلاعاتی دستور DROP DATABASE به صورت زیر استفاده می‌شود:

```
DROP DATABASE database_name [1,...,n];
```

database_name[1, …, n]، نام بانک‌های اطلاعاتی هستند که می‌خواهید حذف شوند. اگر بانک اطلاعاتی را حذف کنید، کلیه فایل‌های داده و کارنامه متعلق به آن حذف خواهند شد.

مثال ۳ – ۴. پرس‌وجویی که بانک اطلاعاتی Student را حذف خواهد کرد. یعنی، فایل‌های داده و کارنامه (فایل‌های Student_data.mdf و Student_log.ldf موجود در پوشه Database درایو D) حذف خواهند شد.

```
DROP DATABASE Student
```

در هنگام اجرای دستورات ALTER DATABASE و DROP DATABASE، اگر بانک اطلاعاتی که می‌خواهید خواص آن را تغییر دهید یا آن را حذف نمایید، موجود نباشد، بانک اطلاعاتی SQL Server پیغام خطا صادر می‌کند. ولی، اگر در هنگام اجرای دستور CREATE DATABASE، چنانچه بانک اطلاعاتی که می‌خواهید ایجاد کنید از قبل موجود باشد، پیغام خطا ظاهر می‌گردد.

۴–۴. ایجاد و تغییر ساختار جدول

همان‌طور که می‌دانید، جداول از مجموعه‌ای از رکوردها تشکیل می‌شوند، و رکوردها نیز از مجموعه‌ای از فیلدها تشکیل می‌گردند و فیلدها همچنین حاوی تعدادی خواص هستند. بنابراین، قبل از این که به ایجاد جدول بپردازیم، باید **خواص فیلدها** را بررسی کنیم. این خواص در زیر آمده‌اند:

نام فیلد: هر فیلد در جدول یک نام **یکتا**[12] دارد که از قانون نام‌گذاری شناسه‌ها در بانک اطلاعات پیروی می‌کند.

نوع فیلد: برای هر فیلد باید تعیین شود، اولاً چه نوع داده‌ای را می‌تواند ذخیره کند، ثانیاً تعداد بایت‌هایی که می‌تواند ذخیره کند، چقدر است. انواع داده‌ها و مقدار فضایی که هر یک از انواع در بانک اطلاعات نیاز دارند، در جدول ۱- ۴ آمده است.

محدودیت‌های فیلد: این ویژگی‌های تعیین می‌کنند هر فیلد چه محدودیت‌های (قیدهای) دارد. برخی از این محدودیت‌ها در زیر آمده‌اند:

۱. محدودیت کلید اولیه[2]: فیلدی کلید اولیه است که دارای شرایط زیر باشد (مانند شماره کارمندی، شماره دانشجویی، شماره وام، شماره مشتری، شماره درس، کد گروه و کد شهر):

- در یک جدول مقدار هیچ دو رکوردی از این فیلد یکسان نباشد.
- اطلاعات رکوردها بر اساس این فیلد مرتب باشند.
- مقدار این فیلد نمی‌تواند **تهی**[3] باشد.

مثال ۴ – ۴. دستوری که فیلد کد شهر را به عنوان کلید اولیه جدول City تعریف می‌کند.

```
PRIMARY KEY (CityCode)
```

مثال۵ – ۴. دستوری که فیلد شماره دانشجویی را کلید اولیه معرفی می‌کند.

```
PRIMARY KEY (stNo)
```

مثال ۶ – ۴. دستوری که فیلدهای شماره دانشجویی، شماره درس و ترم را به عنوان کلید اولیه تعریف می‌کند.

```
PRMARY KEY (stNo, courseNo, Term)
```

---

[12] .Unique [2].Primary Key [3].Null

جدول ۱- ٤ انواع داده در SQL.		
نام	تعداد بایت	هدف
Int	4	اعداد صحیح از ۲,۱۴۷,۴۸۳,۶۴۸- تا ۲,۱۴۷,۴۸۳,۶۴۷ است.
intBig	8	اعداد صحیح از $۲^{۶۳}$- تا ۱- $۲^{۶۳}$ است.
Binary(n)	n	داده‌های دودویی حداکثر ۲۵۵ بایت را اشغال می‌کنند.
Bit	1	مقادیر ۰ یا ۱ را می‌پذیرد.
Char(n)	n	حداکثر تا ۸۰۰۰ کاراکتر را می‌پذیرد و به ازای هر کاراکتر یک بایت را اشغال می‌کند.
Datetime	8	تاریخ و زمان را نگهداری می‌کند.
Decimal(p, s)	حداکثر17	اعداد اعشاری یا صحیح $۱+^{۲۸}۱۰$- تا$۱-^{۲۸}۱۰$ را نگهداری می‌کند.
Float(n)	8	از مقدار$۷۹E^{۲۸}$, ۱- تا$۷۹E^{۲۸}$, ۱ را نگهداری می‌کند و دقت آن تا ۱۵ رقم بعد از اعشار است.
Real(n)	4	تقریباً از مقدار$۳.۴۰E^{۲۸}$- تا $۳.۴۰E^{۲۸}$ را نگهداری می‌کند (با تقریب ۷ رقم).
Image		داده‌های تصویر تا 2GB را در صفحات ۸ کیلوبایتی ذخیره می‌کند.
Money	8	داده‌های ارزی از ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷.۵۸۰۸-تا۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷.۵۸۰۷ است.
Varchar(n)	n*2	داده‌های یونی کد از ۱ تا ۴۰۰۰ کاراکتر را نگهداری می‌کند.
Ntext	n*2	داده‌های یونی کد با طول متغیر که طول آن ۱۰۷۳۷۴۱۸۲۳ است.
Numeric	حداکثر17	مترادف Decimal است.
Real	4	مترادف Float است.
Smalldatetime	4	ترکیب تاریخ و زمان با طول کوتاه را نگهداری می‌کند.
Smallint	2	اعداد صحیح از ۳۲,۷۶۸- تا ۳۲,۷۶۷ را نگهداری می‌کند.
Currency	8	مقادیر پولی با حداکثر ۴ رقم اعشار را نگهداری می‌کند.
Sysname(n)	n*2	برای ذخیره کردن اسامی اشیای سیستم به کار می‌رود و به ازای هر کاراکتر دو بایت را اشغال می‌کند.
Timestamp		برای نگهداری مهر زمانی در بانک اطلاعاتی به کار می‌رود.
Tinyint(n)	n	اعداد صحیح ۰ تا ۲۵۵ بایت که n تعداد بایت‌ها را مشخص می‌کند.
Varbinary(n)	n	الگوی بیتی تا ۲۵۵ بایت را نگهداری می‌کند.
uniqueidentifier	16	عدد ۱۶ بایتی هگزا دسیمال که شناسه یکتای عمومی را نشان می‌دهد.

XML	حداکثر 2GB	برای نگهداری اسناد XML به کار می‌رود.
Varbinary(max)		اطلاعات بایتی را طوری در نظر می‌گیرد که می‌توانید تصاویر را در آن ذخیره کنید.

۲.ایجاد محدودیت کلید خارجی[۱۳]: کلید خارجی برای ایجاد ارتباط بین دو جدول به کار می‌رود. یعنی، با استفاده از کلید اولیه یا کلید فرعی یک جدول و کلید خارجی جدول دیگر می‌توان ارتباط بین این دو جدول را برقرار کرد. کلید خارجی در واقع کلید اولیه یا کلید فرعی جدول دیگر در همان بانک است. برای تعریف کلید خارجی به صورت زیر عمل می‌شود:

```
FOREIGN KEY ( فیلد۱, ... , فیلدn) REFERENCES نام جدول
```

مثال ۷ – ۴. دستوری که فیلد cityCode را بین جدول Student و Ctiy به عنوان کلید خارجی تعریف می‌کند.

```
FOREIGN KEY (cityCode) REFERENCES city
```

وقتی که محدودیت جامعیت ارجاع نقض گردد، معمولاً عملی که موجب نقض گردیده است، رد می‌شود. یعنی، اگر بخواهید در جدول Type، کد نوع درس ۰۱ را حذف کنید، در صورتی که این کد در جدول Course استفاده شده باشد، دستور حذف از جدول Type لغو خواهد شد. بخش FOREIGN KEY را می‌توان طوری تعریف کرد که تراکنش‌های حذف و به هنگام رسانی لغو نگردند. برای این منظور می‌توانند از گزینه‌های زیر استفاده کنید:

- **گزینه ON DELETE CASCADE**، موجب می‌شود تا عمل حذف لغو نگردد. ابتدا، در تمام جداولی که با این جدول از طریق این فیلد ارتباط دارند، رکوردهای مرتبط به صورت آبشاری حذف خواهند شد. سپس، در جدول اصلی مقدار مورد نظر حذف می‌گردد. یعنی، اگر بخواهید در جدول Type، کد نوع درس ۰۱ را حذف کنید، ابتدا در جدول Course، رکوردهایی با کد نوع درس ۰۱ حذف خواهند شد. سپس، در جدول Type، رکوردی با کد نوع درس ۰۱ حذف می‌گردد.
- **گزینه ON UPDATE CASCADE**، اگر در هنگام به روز رسانی در فیلدی که محدودیت جامعیت در آن تعریف شده باشد، این محدودیت نقض شود، به هنگام رسانی لغو نمی‌گردد. به عنوان مثال، اگر بخواهید در جدول Type، کد نوع درس ۰۱ را به ۰۳ تغییر دهید. ابتدا، در جدول Course، کد نوع‌های (typeCode)، ۰۱ را به ۰۳ تغییر می‌دهد. سپس، در جدول Type این تغییر را اعمال می‌کند.

---

[1]- Foreign Key

استفاده از گزینه‌های ON DELETE CASCADE و ON UPDATE CASCADE به صورت زیر است:

```
CREATE TABLE Course
(
…
   FOREIGN KEY (typeCode) REFERENCES Type
   ON DELETE CASCADE,
   ON UPDATE CASCADE,
   …
)
```

۳. محدودیت DEFAULT: این محدودیت مقداری را تعیین می‌کند که اگر کاربر در هنگام ورود برای فیلد، مقدار وارد نکند، آن مقدار در این فیلد قرار می‌گیرد. یعنی، مقدار پیش‌فرض فیلد را تعیین می‌نماید.

۴. محدودیت UNIQUE: تضمین می‌کند صفات (فیلدهای) تعیین شده تشکیل کلید کاندید را دهند. یعنی، رکوردهای (چند تایی‌های) جدول (رابطه) **یکتا** هستند (هیچ دو رکوردی در جدول تکراری نیستند). صفات UNIQUE می‌توانند تهی (Null) باشند، مگر این که با محدودیت NOT NULL معرفی گردند. این محدودیت به صورت زیر به کار می‌رود:

```
UNIQUE (فیلد ۱, فیلد۲, . . . , فیلد n)
```

۵. محدودیت NOT NULL: تضمین می‌کند مقدار فیلد در هیچ رکوردی (چند تایی) تهی نباشد. به عنوان مثال، نام دانشجو نمی‌تواند تهی باشد. پس، به صورت زیر تعریف می‌گردد:

```
Fname   varchar (15) NOT NULL
```

۶. محدودیت CHECK: یکی از رایج‌ترین محدودیت‌ها CHECK است. این محدودیت تضمین می‌کند مقادیر صفت (فیلد)، شرایط تعیین شده را داشته باشد.

مثال ۲۱ -۴. پرس‌وجویی که نام درس و تعداد واحدهای آن را نمایش می‌دهد.

```
SELECT courseName, unit
FROM Course
```

courseName	unit
مباني برنامه سازي	4
اصول حسابداري	3
اخلاق	2
رياضي ۱	3
تجارت الكترونيك	2
پايگاه داده	3

مثال ۲۲ - ۴. پرس‌وجویی که شماره دانشجویانی که درس انتخاب کرده یا پاس کرده‌اند، را نمایش می‌دهد.

همان‌طور که در این

```
SELECT stNo
FROM Score
```

stNo
87662210
87891101
87891101
87891166
87891166
87961122

خروجی می‌بینید، برخی از شماره‌های دانشجویی تکرار شده‌اند. اگر بخواهید مقادیر تکراری را در خروجی دستور SELECT حذف کنید، باید بعد از SELECT از کلمه DISTINCT استفاده کنید (مثال زیر را ببینید):

مثال ۲۳ – ۴. پرس‌وجویی که شماره دانشجویانی را نمایش می‌دهد که درس انتخاب یا پاس کرده‌اند (شماره‌های دانشجویی تکراری را فقط یک بار نمایش می‌دهد).

```
SELECT DISTINCT stNo
FROM Score
```

stNo
87662210
87891101
87891166
87961122

همان‌طور که در این خروجی مشاهده می‌کنید، شماره‌های دانشجویی فقط یک بار نمایش داده شده‌اند.

همان‌طور که بیان گردید، بخش WHERE برای بازیابی رکوردهایی که دارای شرط خاص هستند، به کار می‌رود. در بخش WHERE می‌توانید از عملگرهای مقایسه‌ای (<، >، =<، =>، = و <>)، منطقی (AND، OR و NOT) و عملگرهای ویژه (BETWEEN، IN، LIKE ، ALL، SOME و ANY) و غیره استفاده کنید. چند نمونه از این پرس‌وجوها در زیر آمده‌اند.

مثال ۲۴ – ۴. پرس‌وجویی که دانشجویانی را نمایش می‌دهد که نام آن‌ها "زهرا" است.

```
SELECT * FROM Student
WHERE Fname='زهرا'
```

stNo	Fname	Lname	cityCode	groupCode	clgNo
87891166	زهرا	علوي	3	2	12

مثال ۲۵ – ۴. پرس‌وجویی که لیست نمرات بین ۱۵ تا ۲۰ را نمایش می‌دهد.

```
SELECT * FROM Score
WHERE Score between 15
   AND 20
```

stNo	courseNo	teacherCode	term	Score
87891101	10	10	8801	16.00
87961122	50	14	8901	16.00

مثال ۲۶ – ۴. پرس‌وجویی که نام، نام خانوادگی و مدرک اساتیدی را نمایش می‌دهد که در مدرک آن‌ها کلمه "دکتر" وجود دارد.

```
SELECT Fname, Lname, Ranking
FROM Teacher
WHERE Ranking LIKE  '%دکتر%'
```

Fname	Lname	Ranking
رضا	ابوطالبي	دكتري
احمد	جلالي	دكتري

همان‌طور که در این خروجی می‌بینید، فیلدهای نام، نام خانوادگی و مدرک اساتیدی بازیابی گردید که عبارت **"دکتر"** در مدرک آن‌ها وجود دارد.

مثال ۲۷ – ۴. پرس‌وجویی که نام و نام خانوادگی دانشجویانی را نمایش می‌دهد که نام آن‌ها "علی"، "محمد" یا "جواد" باشد.

```
SELECT Fname, Lname FROM Student
WHERE Fname IN ('علی' , 'محمد' , 'جواد')
```

Fname	Lname
محمد	یوسفی
جواد	حسین زاده

مثال ۲۸ – ۴. پرس‌وجویی که نام و نام خانوادگی دانشجویانی را نمایش می‌دهد که نام خانوادگی آن‌ها با حرف "ز" شروع شده باشد.

```
SELECT Fname, Lname  FROM Student
WHERE Lname LIKE  '%ز'
```

Fname	Lname

مثال ۲۹ – ۴. پرس‌وجویی که دانشجویانی را نمایش می‌دهد که نام آن‌ها دقیقاً سه کاراکتر باشد.

```
SELECT * FROM Student
WHERE Fname LIKE'---%'
```

stNo	Fname	Lname	cityCode	groupCode	clgNo
87891101	رضا	رضوانی	1	1	10

مثال ۳۰ – ۴. پرس‌وجویی که تمام اساتیدی را نمایش می‌دهد که نام و نام خانوادگی آن‌ها حداقل چهار حرف دارد.

```
SELECT  *  FROM Teacher
WHERE Fname LIKE  '--%'
  AND Lname LIKE  '--%'
```

teacherCode	Fname	Lname	Ranking
11	زهرا	حسنی	فوق لیسانس
12	احمد	جلالی	دکتری
13	محمد	هاشمی	فوق لیسان
14	رمضان	عباس نژاد	فوق لیسان
15	امیر	خدابنده	فوق لیسان

در هنگام استفاده از دستور SELECT می‌توان عنوان ستون‌ها را تغییر داد. برای این منظور، می‌توانید از کلمه AS استفاده کرده، بعد از عبارت AS نام جدید ستون را تایپ نمایید. چنانچه نام ستون بیش از یک کلمه باشد و بین کلمات جای خالی[14] قرار می‌گیرد باید نام ستون را در بین تک گیومه (') قرار دهید.

مثال ۳۱ – ۴. پرس‌وجویی که نام درس و تعداد واحدهای درس‌های سه واحدی را نمایش می‌دهد (در عنوان ستون courseName، نام درس و در عنوان ستون unit، تعداد واحد را نمایش می‌دهد).

```
SELECT courseName AS 'نام درس',unit AS  'تعداد واحد'
FROM Course  WHERE unit = 3
```

نام درس	تعداد واحد
اصول حسابداری	3
ریاضی ۱	3
پایگاه داده	3

۹ – ۴. مرتب سازی رکوردها (تاپل)

یکی از ویژگی‌های جالب دستور SELECT، امکان مرتب‌سازی تاپل‌های جدول است. برای مرتب‌سازی رکوردها در دستور SELECT می‌توانید از گزینه **ORDER BY** استفاده کنید. برای تعیین نوع مرتب‌سازی در بخش ORDER BY می‌توانید از گزینه‌های **DESC**، برای مرتب

[14] .Blank

سازی نزولی یا **ASC**، برای مرتب‌سازی صعودی استفاده کنید. اگر نوع مرتب‌سازی ذکر نگردد، به طور پیش‌فرض صعودی (**ASC**) در نظر گرفته می‌شود.

ORDER BY ۱ نام فیلد [نوع مرتب سازی] , نام فیلد۲ [نوع مرتب سازی] , ...

مثال ۳۲ – ۴. پرس‌وجویی که لیست دروسی را نمایش می‌دهد که تعداد واحدهای آن‌ها بیش از ۲ واحد است (اطلاعات را بر اساس تعداد واحد مرتب کرده، نمایش می‌دهد).

```
SELECT * FROM Course
WHERE unit > 2
ORDER BY  unit
```

courseNo	courseName	unit	typeCode
20	اصول حسابداري	3	1
40	رياضي ۱	3	3
60	پايگاه داده	3	1
10	مباني برنامه سازي	4	1

همان‌طور که در این خروجی می‌بینید، نتیجه بر اساس تعداد واحد به صورت صعودی مرتب گردید (نوع مرتب‌سازی ذکر نشده، پس صعودی در نظر گرفته می‌شود).

مثال ۳۳ – ۴. پرس‌وجویی که لیست اساتیدی را نمایش می‌دهد که مدرک آن‌ها فوق لیسانس است (اطلاعات را بر اساس نام خانوادگی اساتید به صورت نزولی مرتب می‌کند).

```
SELECT * FROM Teacher
WHERE Ranking = 'فوق ليسانس'
ORDER BY  Lname DESC
```

teacherCode	Fname	Lname	Ranking
13	محمد	هاشمي	فوق ليسانس
14	رمضان	عباس نژاد	فوق ليسانس
15	امير	خدابنده	فوق ليسانس
11	زهرا	حسني	فوق ليسانس

علاوه بر این، به جای یک فیلد مرتب‌سازی می‌توان چند فیلد را برای مرتب‌سازی انتخاب نمود. برای این منظور، بعد از گزینه ORDER BY، فیلدها را با کاما از یکدیگر جدا کنید. در این صورت، چنانچه رکوردهایی وجود داشته باشند که مقدار فیلد اول آن‌ها برابر باشد، بر اساس فیلد دوم مرتب‌سازی را انجام می‌دهد. اگر رکوردهایی وجود داشته باشند که مقدار فیلد اول و دوم مرتب‌سازی برابر داشته باشند، بر اساس فیلد سوم مرتب می‌نماید و این روند را ادامه می‌دهد.

مثال ۳۴ – ۴. پرس‌وجویی که لیست دروس را بر اساس تعداد واحد به صورت نزولی مرتب می‌کند. اگر تعداد واحدها مساوی بودند، بر اساس نام درس به صورت صعودی مرتب می‌کند.

```
SELECT *  FROM Course
ORDER BY unit DESC,courseName ASC
```

courseNo	courseName	unit	typeCode
10	مباني برنامه سازي	4	1
20	اصول حسابداري	3	1
60	پايگاه داده	3	1
40	رياضي ۱	3	3
30	اخلاق	2	2
50	تجارت الكترونيك	2	4

۱۰ – ۴. پرس‌وجوهای مرکب

این پرس‌وجوها از چندین پرس‌وجو تشکیل شده‌اند. عملگرهایی که برای ایجاد پرس‌وجوهای مرکب به کار می‌روند، عبارت‌اند از :

۱. عملگر UNION (همان عملگر ∪ جبر رابطه‌ای).

۲. عملگر INTERSECT (همان عملگر ∩ جبر رابطه‌ای).

۳. عملگر EXCEPT (همان عملگر − جبر رابطه‌ای).

۴. عملگر UNION – ۱۰ – ۱

Fname	Lname
احمد	احمدي
احمد	جلالي
امير	خدابنده
جواد	حسين زاده
رضا	ابوطالبي
رضا	رضوانی
رمضان	عباس نژاد
زهرا	حسني
زهرا	علوي
ليلا	رضازاده
محمد	هاشمي
محمد	يوسفي
مريم	كمالي

این عملگر نتیجه دو یا چند دستور SQL را با هم ترکیب کرده، همه رکورد-های پرس‌وجویی اول و دوم و رکوردهای تکراری را فقط یک بار نمایش می-دهد.

مثال ۳۵ – ۴. پرس‌وجویی که تمام افراد موجود در بانک اطلاعات دانشجویان را نمایش می‌دهد.

این پرس‌وجو، ابتدا دانشجویان را بازیابی می‌نماید. سپس، اساتید را بازیابی کرده، آن‌ها را با هم ترکیب می-کند. یعنی:

```
(SELECT Fname, Lname FROM Student)
 UNION
(SELECT Fname, Lname  FROM Teacher)
```

همان‌طور که در این خروجی می‌بینید، نام و نام خانوادگی دانشجویان و اساتید نمایش داده شده است.

۴. عملگر UNION ALL – ۱۰ – ۲

عمل می‌کند، با این تفاوت که رکوردهای تکراری را حذف نمی‌کند. UNIONاین عملگر همانند عملگر

مثال ۳۶ – ۴. پرس‌وجویی که کد درس‌های انتخاب شده و انتخاب نشده را نمایش می‌دهد (این پرس‌وجو، شماره درس‌های تکراری را چند بار نمایش می‌دهد).

courseNo
60
10
40
10
40
50
10
20
30
40
50
60

```
(SELECT courseNo FROM Score)
UNION  ALL
(SELECT courseNo FROM Course)
```

همان‌طور که در این خروجی می‌بینید، برخی از شماره درس‌ها تکرار (**مانند ۴۰ و ۵۰**) شده‌اند.

۳ – ۱۰ – ۴. عملگر INTERSECT

این عملگر از نتیجه چند پرس‌وجو اشتراک می‌گیرد (مانند عملگر ∩ جبر رابطه‌ای) و نتیجه اشتراک را برمی‌گرداند.

مثال ۳۷ – ۴. پرس‌وجویی که کد شهرهایی که دانشجو دارند را نمایش می‌دهد.

```
(SELECT cityCode FROM City)
INTERSECT
(SELECT cityCode FROM Student)
```

cityCode
1
2
3

مثال ۳۸ – ۴. پرس‌وجویی که شماره دروسی که توسط دانشجویان انتخاب شده‌اند را نمایش می‌دهد.

```
(SELECT courseNo FROM Course)
INTERSECT
(SELECT courseNo FROM Score)
```

courseNo
10
40
50
60

مثال ۳۹ – ۴. پرس‌وجویی که شماره اساتیدی که تاکنون درس ارائه کرده‌اند را نمایش می‌دهد.

```
(SELECT teacherCode FROM Teacher)
INTERSECT
(SELECT teacherCode FROM Score)
```

teacherCode
10
12
14

اگر فقط دستور دوم اجرا شود، کدهای بعضی از اساتید تکراری نمایش داده می‌شوند.

۴ – ۱۰ – ۴. عملگر EXCEPT

این عملگر، تفاضل بین نتیجه پرس‌وجوی اول و دوم را برمی‌گرداند (مانند عملگر – در جبر رابطه‌ای).

یعنی، رکوردهایی را برمی‌گرداند که در پرس‌وجوی اول باشند، ولی در نتیجه پرس‌وجوی دوم نباشند.

مثال ۴۰ – ۴. پرس‌وجویی که کد شهرهایی که دانشجو ندارند، را نمایش می‌دهد.

```
(SELECT cityCode FROM City)
 EXCEPT
(SELECT cityCode FROM Student)
```

cityCode
4

۲۰–۴. رویه‌های ذخیره شده

رویه‌های ذخیره شده، تعدادی از دستورات SQL هستند که با هم در یک مجموعه قرار گرفته، کامپایل می‌شوند و با یک دستور SQL قابل اجرا می‌باشند. رویه‌های ذخیره شده مزایای متعددی دارند که برخی از آن‌ها عبارت‌اند از:

۱. سرعت اجرای رویه‌های ذخیره شده بالاست. زیرا، رویه‌های ذخیره شده، به صورت کامپایل شده در حافظه نهان[۱۵] بانک اطلاعات نگهداری می‌شوند و از طرف دیگر، چون کامپایل شده‌اند، در هنگام اجرا نیاز به کامپایل شدن ندارند. پس، سرعت اجرای آن‌ها افزایش می‌یابد.

۲. رویه‌های ذخیره شده برنامه‌نویسی ماژولی[۲] را امکان‌پذیر می‌سازند. زیرا، یک بار رویه‌های ذخیره شده را می‌نویسید، کامپایل می‌کنید و چند بار آن‌ها را فراخوانی می‌نمایید تا اجرا شوند.

۳. رویه‌های ذخیره‌شده ترافیک شبکه را کاهش می‌دهند. زیرا، ممکن است رویه‌های ذخیره شده از چندین سطر تشکیل شوند. در حالت عادی، وقتی سرویس‌گیرنده[۳] چند خط را به عنوان پرس‌وجو برای سرویس‌دهنده ارسال می‌کند، ترافیک شبکه افزایش می‌یابد. در حالتی که تعداد سرویس گیرنده‌ها (کاربران) در محیط شبکه زیاد باشد، این موضوع بیشتر خودش را نشان می‌دهد. ولی، اگر از رویه‌های ذخیره شده استفاده کنید، جهت اجرا فقط کافی است یک خط را بفرستید تا رویه ذخیره شده اجرا گردد. چون، بدنه رویه ذخیره شده در سمت سرویس‌دهنده[۴] قرار دارد.

٤. رویه‌های ذخیره شده را می‌توان به طور خودکار پس از راه‌اندازی SQL Server اجرا نمود.

نام رویه‌های ذخیره شده در جدول sys.Objects و دستورات آن در جدول sys.Comments ذخیره می‌شوند.

۱-۲۰-۴. ایجاد رویه‌های ذخیره شده

رویه‌های ذخیره شده به عنوان یک شیء در بانک اطلاعات ذخیره می‌شوند که می‌توانید با دستور CREATE PROCEDURE آن‌ها را ایجاد کنید. این دستور به صورت زیر به کار می‌رود:

```
CREATE PROC[EDURE ] procedure_name [ ; number ]
[ { @parameter data_type }
 VARYING ] [ = DEFAULT ] [ OUTPUT ]
 ] [ ,...n ]
[ WITH {RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION}]
[ FOR REPLICATION ]
AS sql_statement [ ...n ]
```

پارامترهای این دستور در جدول ۹- ۴ آمده‌اند.

مثال ۷۷ – ۴. پرس‌وجویی که رویه ذخیره شده delete_City. را ایجاد می‌کند.

```
USE [Student]
GO
```

[1] .Cache Memory [2].Modular Programming [3].Client [4] . Server

```
CREATE PROCEDURE [dbo].[delete_City]
 (@cityCodeVarchar(6),
  @strOut varchar(1) OUTPUT)
AS
 IF   @cityCode IN (SELECT cityCode FROM Student)
    SET @strOut='1' - کد شهر در جدول دانشجو استفاده گردید
 ELSE  IF   @cityCode NOT IN (SELECT cityCode FROM City)
    SET @strOut='2' - کد شهر وجود ندارد
 ELSE
  BEGIN
   DELETE  FROM [City]   WHERE ( [cityCode] = @cityCode)
    SET @strOut='0' -- شهر حذف گردید
END
```

این دستورات، رویه ذخیره‌شده delete_City را ایجاد می‌کنند که پارامتر cityCode@ را از ورودی به عنوان پارامتر گرفته و پارامتر strOut@ را به صورت خروجی برمی‌گرداند. این رویه ذخیره شده، اگر کد شهر در جدول دانشجو وجود داشته باشد، strOut@ را برابر با '1' وگرنه، اگر کد شهر در جدول شهر وجود نداشته باشد، strOut@ را برابر '2' وگرنه، شهر را حذف می‌کند و مقدار strOut@ برابر '0' خواهد شد.

۲-۲۰-۴. اجرای رویه ذخیره شده

برای اجرای رویه ذخیره شده در محیط SQL می‌توانید از دستور EXECUTE استفاده کنید. این دستور به صورت زیر به کار می‌رود.

```
[ { EXEC | EXECUTE } ]
 { [ @return_status = ]
 { module_name [ ;number ] | @module_name_var }
 [ [ @parameter = ] { value | @variable [ OUTPUT ]
 | [ DEFAULT ] }][ ,...n ][ WITHRECOMPILE ]}[;]
```

جدول ۹-٤ پارامترهای دستور CREATE PROCEDURE.

پارامتر	هدف
procedure_name	نام رویه ذخیره شده‌ای را تعیین می‌کند که ایجاد می‌شود.
number	شماره‌ای را مشخص می‌کند که می‌توانید به رویه ذخیره شده تخصیص دهید تا برخی از رویه‌های ذخیره شده که در یک گروه قرار می‌گیرند را با این شماره بتوانید فراخوانی کنید.

@parameter	پارامترهایی را تعیین می‌کنند که می‌توانید مقادیر را از طریق آن‌ها برای رویه ذخیره شده ارسال کنید.
data_type	نوع هر یک از پارامترهای رویه ذخیره شده را تعیین می‌کند.
VARYING	مجموعه نتایج را به عنوان پارامتر خروجی مشخص می‌کند.
DEFAULT	مقدار پیش‌فرض پارامتر رویه ذخیره شده را تعیین می‌کند. اگر رویه ذخیره شده را بدون پارامتر فراخوانی کنید، این مقدار برای پارامتر رویه ذخیره شده در نظر گرفته می‌شود.
OUTPUT	نوع پارامتر را خروجی در نظر می‌گیرد.
WITH RECOMPILE	با هر فراخوانی رویه ذخیره شده دوباره کامپایل می‌گردد.
WITH ENCRYPTION	دستورات رویه ذخیره شده را رمزگذاری می‌کند تا قابل بازیابی نباشد.
FOR REPLICATION	رویه ذخیره شده را برای تکثیر ایجاد می‌کند که توسط سرویس گیرنده قابل اجرا نیست.
sql_statement	دستوراتی هستند که رویه ذخیره شده را ایجاد می‌کنند.

پارامترهای این دستور عبارت‌اند از:

- **متغیر return_status:** وضعیت خروجی را برمی‌گرداند.
- **متغیر @procedure_name:** نام متغیر پارامتر را تعیین می‌کند.
- **پارامتر WITH RECOMPILE:** تعیین می‌کند رویه ذخیره شده قبل از اجرا کامپایل گردد و سپس، اجرا شود. در حالت معمولی، رویه ذخیره شده قبل از اجرا کامپایل نمی‌شود.

به عنوان مثال، دستورات زیر را ببینید:

```
DECLARE @strOut varchar (1) = '0'
EXECUTE delete_City '100', @strOut output
```

این دستور، با اجرای رویه ذخیره شده delete_City، شهری با کد '100' را حذف می‌کند.

۳-۲۰-۴. تغییر رویه ذخیره شده

رویه‌های ذخیره شده موجود در بانک اطلاعات را می‌توانید با دستور ALTER PROCEDURE تغییر دهید.

ساختار و پارامترهای ALTER PROCEDURE مانند CREATE PROCEDURE است.

۲۲ – ۴. مسائل حل ش

مثال ۱ – ۴. اگر ساختار رابطه‌های R و S به صورت زیر باشند:

R = (A, B, C)
S = (D, E, F)

رابطه‌های r(R) و s(S) داده شده است. دستورات SQL معادل هر یک از عبارات جبر رابطه‌ای زیر را بنویسید:

1 - 1) $\sigma_{B = 12}(R)$ ⇨ **SELECT** * **FROM** R **WHERE** B = 12

1 – 2) $\prod_{D, E}(S)$ ⇨ **SELECT** D, E **FROM** S

1 – 3) R × S ⇨ **SELECT** * **FROM** R, S

1 – 4) $\prod_{B, E}(\sigma_{C = D}(R \times S))$ ⇨ **SELECT** D, E **FROM** R, S **WHERE** C = D

1 – 5) $\prod_{D, F}(\sigma_{E = 10}(S))$ ⇨ **SELECT** D, F **FROM** S **WHERE** E = 10

مثال ۲ – ۴. با توجه به جداول پایگاه داده "بانک" (جدول ۱۱ - ٤)، به پرس‌وجوهای زیر پاسخ دهید:

۱ – ۲ – ۴. پرس‌وجویی که نام شعبه‌های جدول Loan را بازیابی می‌کند.

```
SELECT branch_name FROM Loan
```

۲ – ۲ – ۴. پرس‌وجویی که شماره وام‌های شعبه بانک ملی مرکزی را بازیابی می‌کند که مبلغ وام آن‌ها بیش از ۱۰۰۰۰۰۰۰ تومان است.

```
SELECT loan_number FROM Loan
  WHERE branch_name = 'بانک ملی مرکزی' AND amount > 10000000
```

۳ – ۲ – ۴. پرس‌وجویی که شماره وام، نام مشتری و میزان وام آن‌ها را بازیابی می‌کند که از بانک وام گرفته‌اند.

```
SELECT customer_name, loan_number, amount FROM Borrower b
INNER JOIN Loan ON b.loan_number = loan_number
```

۴ – ۲ – ۴. پرس‌وجویی که نام مشتریان، شماره وام‌ها و میزان وام مشتریانی را بازیابی می‌کند که از بانک ملی مرکزی وام دریافت کرده‌اند.

```
SELECT customer_name, t1.loan_number, amount FROM Borrower t1
INNER JOIN Loan t2 ON t1.loan_number = t2.loan_number
WHERE branch_name = 'بانک ملی مرکزی'
```

۵ – ۲ – ۴. پرس‌وجویی که نام شعبه‌هایی را بازیابی می‌کند که دارایی آن‌ها بیش از حداقل یک شعبه در شهر "بابل" است.

```
SELECT DISTINCT t1.branch_name FROM Branch t1, Branch AS t2
```

```
WHERE   t1.assets > t2.assets  AND   t2. branch_name = 'بابل'
```

۶ – ۲– ۴. پرس‌وجویی که اسامی مشتریانی را بازیابی می‌کند که در نام آن‌ها کلمه "محمد" موجود باشد.

```
SELECT customer_name  FROM  Customer
WHERE customer_name LIKE  '%محمد%'
```

۷ – ۲– ۴. پرس‌وجویی که مشتریانی که حساب، وام یا هر دو را دارند، بازیابی می‌کند.

```
(SELECT customer_name FROM  Despositor)
    UNION
    (SELECT customer_name FROM  Borrower )
```

۸ – ۲– ۴. پرس‌وجویی که مشتریانی که هم حساب و هم وام دارند را بازیابی می‌کند.

```
(SELECT DISTINCT customer_name FROM Depositor)
INTERSECT
(SELECT DISTINCT customer_name FROM Borrower )
```

۹ – ۲– ۴. پرس‌وجویی که نام شعبه‌ها و میانگین موجودی آن‌ها را برمی‌گرداند که میانگین موجودی آن‌ها بیش از ۱۲۰۰۰۰۰۰ تومان است.

```
SELECT  branch_name, AVG (balance)  FROM  Account
GROUP  BY  branch_name
    HAVING  AVG  (balance) > 12000000
```

۱۰ – ۲– ۴. پرس‌وجویی که میانگین مانده مشتریانی که در شهر بابل زندگی می‌کنند و حداقل ۲ حساب دارند را بازیابی می‌کند.

```
SELECT d.customer_name, AVG(balance)
FROM Depositor AS d, Account AS a, Customer AS  c
WHERE d.account_number = a.account_number
 AND d.customer_name = c.customer_name AND c.customer_city = 'بابل'
GROUP BY d.customer_name
HAVING  COUNT(DISTINCT d.account_number) > = 2
```

مثال ۵ – ۴. پایگاه داده جدول ۱۲ – ٤ را در نظر بگیرید. اکنون به پرس‌وجوهای زیر پاسخ دهید:

۱ – ۵– ۴. نام و شهرهای محل اقامت کارکنانی را پیدا کنید که برای شرکت مخابرات مازندران کار می‌کنند.

```
SELECT e.employee_name, city   FROM   Employee   e, Works   w
WHERE    w.company_name =  'شرکت مخابرات مازندران'
AND     w.employee_name = e.employee_name
```

جدول ۱۲- ٤ بانک اطلاعات کارمند.

نام جدول	نام لاتین فیلد	نام فارسی فیلد

نام کارمند	employee_name	Employee
خیابان	street	
شهر	city	
نام کارمند	employee_name	Works
نام شرکت	company_name	
حقوق	salary	
نام شرکت	company_name	Company
شهر محل شرکت	city	
نام کارمند	employee_name	Manages
نام مدیر	manager_name	

۲ – ۵ – ۴. نام، خیابان و شهرهای محل زندگی کارکنانی را بازیابی کنید که برای شرکت مخابرات مازنـدران کـار می‌کنند و دریافتی آن‌ها بیش از ۱۰۰۰۰۰۰ تومان است.

```
SELECT * FROM Employee WHERE employee_name IN
(SELECT employee_name FROM Works
WHERE company_name = 'شرکت مخابرات مازندران' AND salary > 1000000 )
```

۳ – ۵ – ۴. کارکنانی را بازیابی کنید که برای شرکت مخابرات مازندران کار نمی‌کنند.

```
SELECT employee_name FROM Employee WHERE employee_name NOT IN
(SELECT employee_name FROM Works
WHERE company_name = 'شرکت مخابرات مازندران')
```

۴ – ۵–۴. کارکنانی را بازیابی کنید که از تمام کارکنان شرکت مخابرات مازندران بیشتر حقوق می‌گیرند.

ابتدا تمام افرادی که بیش از یک شرکت کار می‌کنند را به دست می‌آوریم. یعنی، داریم:

```
SELECT employee_name FROM Works WHERE salary > ALL
(SELECT salary FROM Works
WHERE company_name = 'شرکت مخابرات مازندران' )
```

اگر فردی برای چندین شرکت کار کند، می‌خواهیم جمع کل حقوق را برای این شخص به دست آوریـم، مسئله پیچیده‌تر می‌شود. بنابراین، می‌توانیم با پرس‌وجوی فرعی زیر و استفاده از WITH آن را حل کنیم:

```
WITH emp_total_salary AS
  SELECT employee_name, SUM (salary) AS  total_salary
  FROM    Works
  GROUP  BY employee_name)
  SELECT  employee_name
  FROM   Emp_total_salary   WHERE total_salary > ALL
  (SELECT total_salary
    FROM Works, Emp_total_salary
    WHERE Works.company_name = 'شرکت مخابرات مازندران '
AND Emp_total_salary.employee_name = Works.employee_name)
```

۱.در SQL شرط R EXISTS چه موقع صحیح (True) است (ارشد – مهندسی کامپیوتر ۸۸)؟

الف: اگر R در دید VIEW باشد. ب: اگر R یک رابطه پایه (base relation) باشد.

ج: اگر R حداقل یک سطر داشته باشد. د: اگر R حداقل یک ستون داشته باشد.

۲. دستور زیر چه کاری انجام می‌دهد (ارشد- آزاد ۸۷)؟

GRANT UPDATE ON Stud **TO** Ahmadi

الف: اجازه انجام UPDATE روی Stud را به کاربری به نام Ahmadi می‌دهد.

ب: اجازه انجام UPDATE روی Stud را از کاربری به نام Ahmadi باز پس می گیرد.

ج: فایل شاخص را روی Stud از کاربری به نام Ahmadi به روز می کند.

د: باعث می‌شود فایل سابقه (Log File) برای دستورات انجام شده از طرف کاربری به نام Ahmadi بروز شود.

۳.جدول زیر برای یک بانک مفروض است :

مشتری	Customer (**custName** , address)
حساب	Account (**accNo**, custName)
وام	Loan (**loanNo**, custName)

کدام دستور SQL زیر برای تعیین مشتریانی که وام نگرفته‌اند، کافی است (ارشد – سراسری ۸۲)؟

الف: **SELECT** * **FROM** Customer, Loan **WHERE** Customer.custName <> Loan.custName

ب: **SELECT** * **FROM** Customer, Loan **WHERE** Customer.custName = Loan.custName

ج: **SELECT** * **FROM** Customer **WHERE** custName **NOT IN** (**SELECT** custName **FORM** Loan)

د: هیچ کدام

۴. دو جدول X و Y زیر هستند. در صورت اجرای دستور SQL، زیر کاردینالیتی جدول حاصل کدام است (ارشد – آزاد ۸۵).

SELECT * **FROM** Y **RIGHT OUTER JOIN** X **ON** Y.B = X.B **WHERE** Y.C **LIKE** 'M%'

X:

F	B
f1	b1
f2	b2

Y:

A	B	C
a1	b1	MK
a2	b4	MTK
a1	b2	LN
a2	b1	RP

۲۴– ۴ . پاسخ پرسش‌های چهارگزینه‌ای

۱. گزینه (ب) صحیح است. چون تقسیم را پیاده سازی می‌کند.

۲. گزینه (ج) صحیح است. چون، EXISTS موجود بودن حداقل یک رکورد در R را بیان می‌کند.

۳. گزینه (الف) صحیح است. دستور GRANT، برای اعطا مجوز و دستور REVOKE برای باز پس گیری مجوز است.

٤. گزینه (ج) صحیح است. چون، مشتریانی که وام نگرفته‌اند را می‌خواهد، گزینه NOT IN، برای پیاده‌سازی عملگر تفاضل (–) به کار می‌رود.

پیوست الف بخشی از مسائل تکمیلی

بانک اطلاعات قطعات تهیه‌کنندگان و عرضه‌کنندگان را در نظر بگیرید (مانند جدول زیر).

نام جدول	نام فیلد	
S	S#	شماره تهیه‌کننده
	Sname	نام تهیه‌کننده
	City	شهر شهر تهیه‌کننده
	Status	وضعیت تهیه کننده
P	P#	شماره قطعه
	Color	رنگ قطعه
	Pname	نام قطعه
	Type	نوع قطعه
	City	شهر قطعه
	Weight	وزن قطعه
SP	S#	شماره عرضه‌کننده
	P#	شماره قطعه
	QTY	تعداد

اکنون به سوالات ۱و ۲ با توجه به این جدول پاسخ دهید.

مثال ۱. پرس‌وجوهای زیر را به SQL پاسخ دهید.

۱. رنگ و شهر قطعاتی که وزن آن‌ها بیش از ۱۰ کیلو و شهر تولید آن‌ها بابل نباشد را نمایش دهد.

```
SELECT T1.Color, T1.City    FROM P AS T1
WHERE T1.City < > 'بابل'    AND    Weight > 10.0
```

۲. شماره و وزن تمام قطعات را به گرم نمایش دهد.

```
SELECT P#, Weight * 1000    FROM P
```

۳. تعداد کل تهیه کنندگان را نمایش دهد

```
SELECT        COUNT(*) AS Tedad    FROM   S
```

٤. بیشترین و کم‌ترین قطعه P3 را نمایش دهد.

```
CT  MAX (QTY)  MAXQ, MIN (QTY)  AS  MINQ
    SP      WHERE   P#  = 'P3'
```

٥. شماره و وزن قطعاتی را بر حسب گرم نمایش دهد که وزن آن‌ها بیش از ۲۰۰۰۰ گرم است.

```
SELECT  P#,   Weignt  *  1000       FROM     P
     WHERE   Weight * 1000  >  20000
```

٦. شماره قطعاتی را نمایش دهید که وزن آن‌ها بیش از ۱۰۰ کیلو است یا توسط عرضه‌کننده S3 عرضه می‌گردند.

```
(SELECT  P#    FROM     P   WHERE   Weight  >  100.0)
 UNION
   (SELECT  P#    FROM    SP  WHERE   S#  = 'S3')
```

٧. نام عرضه کنندگانی را مشخص کنید که حداقل یک قطعه قرمز رنگ تولید کرده‌اند.

```
SELECT   DISTINCT        Sname     FROM     S
      WHERE  S#     IN  (SELECT    S#     FROM     SP
   WHERE   Color = ' قرمز ')
```

٨. نام عرضه‌کنندگانی را نمایش می‌دهد که قطعه P4 را عرضه نمی‌کنند.

```
SELECT   DISTINCT    Sname        FROM     S
   WHERE      NOT   EXISTS    (SELECT  *  FROM    SP
   WHERE     SP.S# = S.S#     AND      SP.P# = 'P4')
```

٩. شماره قطعاتی را نمایش می‌دهد که بیش از چند عرضه کننده آن‌ها را عرضه می‌کنند.

```
SELECT      P#       FROM     SP
   GROUP    BY    P#
   HAVING    COUNT  (S#)  >  1
```

١٠.کم‌ترین، بیشترین، میانگین و مجموع هر قطعه عرضه شده را نمایش می‌دهد.

```
SELECT  P#, SUM (QTY), AVG (QTY), MIN (QTY), MAX (QTY)
FROM    SP    GROUP   BY    P#
```

١١.نام عرضه‌کنندگانی را نمایش می‌دهد که قطعه P3 را تولید نمی‌کنند.

```
SELECT   DISTINCT    Sname       FROM    S
WHERE  S#   NOT   IN  (SELECT   S#   FROM  SP   WHERE P# = 'P3')
```

١٢.شماره قطعاتی که در بابل تولید شده‌اند را نمایش می‌دهد.

```
 SELECT  P#     FROM   P WHERE  City = ' بابل'
```

١٣.نام تهیه‌کنندگان قطعه P4 را نمایش می‌دهد.

```
SELECT  Sname  FROM  S, SP  WHERE P# = 'P4'  AND  SP.P# = S.P#
```

۱٤.مشخصات قطعاتی را نمایش دهد که در نام آن‌ها عبارت "هن" وجود داشته باشد

```
SELECT  *  FROM  P  WHERE  Pname  LIKE  '% هن%'
```

۱۵.مشخصات قطعاتی را نمایش می‌دهد که در نام آن‌ها کلمه "هن" نباشد.

```
SELECT  *  FROM  P WHERE Pname  NOT  LIKE  '% هن%'
```

۱٦.مشخصات قطعاتی را نمایش می‌دهد که نام آن‌ها مقدار تهی (null) باشد.

```
SELECT  * FROM  P  WHERE  Pname  IS NULL
```

۱۷.مشخصات قطعاتی را نمایش می‌دهد که شهر تولید و عرضه آن‌ها یکی باشد.

```
SELECT  P.* FROM  P, S  WHERE  S.City = P.City
```

۱۸.جفت شماره تهیه‌کنندگانی را نمایش می‌دهد که از یک شهر هستند.

```
SELECT  T1.S#,  T2. S#  FROM  S T1, S T2  WHERE  T1.City  = T2.City
```

۱۹.شماره تهیه‌کنندگانی را نمایش می‌دهد که در شهر تهیه‌کننده "S3" هستند.

```
SELECT S#  FROM  S  WHERE  City =
  (SELECT  City  FROM  S  WHERE  S# = 'S3')
```

۲۰.نام تهیه‌کنندگانی را نمایش می‌دهد که قطعه P2 یا P3 را تهیه می‌کنند.

```
SELECT  Sname  FROM  S  WHERE  S# IN
  (SELECT  S#  FROM  SP  WHERE  P#  IN ('P2', 'P3'))
```

۲۱.نام تهیه‌کنندگانی را نمایش می‌دهد که هیچ قطعه‌ای تولید نکرده‌اند.

```
SELECT Sname  FROM  S  WHERE S# NOT  IN
        (SELECT  S#  FROM  SP)
```

۲۲.شماره قطعه تمام قطعات را نمایش می‌دهد که توسط بیش از ۲ تولید کننده تهیه شده‌اند.

```
SELECT  P# FROM  SP  GROUP BY  P#
HAVING  COUNT (P#)  > 2
```

۲۳.مشخصات قطعاتی را نمایش می‌دهد که وزن آن‌ها ۱۲، ۱۵ یا ۲۰ کیلو است.

```
SELECT  *  FROM  P  WHERE  Weight  IN  (10, 15, 20)
```

فصل ۵ بخشی از وابستگی‌های تابعی و نرمال سازی رابطه‌ها

در فصل دوم، مراحل طراحی ادراکی بانک اطلاعاتی را به کمک مدل ER/EER دیدید. در این فصل فرآیند نگاشت رابطه و طراحی منطقی بانک اطلاعات را می‌بینید. در این فرآیند صفات[۱۶] دسته‌بندی شده‌اند تا رابطه‌ها را ایجاد نمایند. اما، سوالی که هر طراح بانک اطلاعات از خود می‌پرسد، این است که چرا یک دسته-بندی انجام شده از صفات در یک رابطه از رابطه دیگر بهتر است؟ یا چقدر بهتر است؟

بنابراین، به یک معیار سنجش قانون‌مندی[۲] نیاز است تا بر اساس آن بتوان دسته‌بندی صفات را انجام داد. این معیار سنجش به طراح بانک اطلاعات کمک می‌کند که میزان مطلوبیت[۳] یک طراحی را اندازه‌گیری و تست کند. این معیار سنجش قانون‌مند در دو سطح ادراکی[۴] و ذخیره‌سازی[۵]، رابطه را بررسی می‌کند. **در سطح ادراکی** منجر به ایجاد پرس‌وجوهای بهینه‌تر و قانونمندتر می‌گردد و در **سطح ذخیره‌سازی** نیز تعداد دفعات ذخیره و بازیابی را بهبود می‌بخشد. این فرآیند قانون‌مند، **نرمال سازی**[۶] رابطه نام دارد. در ادامه، مراحل نرمال سازی را می‌آموزیم.

۱ – ۵. دلایل نرمال سازی رابطه‌ها

فرض کنید تمام جداول بانک اطلاعاتی Student فقط به یک جدول (رابطه) تبدیل شوند. در این صورت، هر دانشجو در زمان ورود به دانشگاه (همان زمان ثبت نام) باید تمام دروس ارائه شده توسط همه اساتید را اخذ کند (یعنی، ضرب دکارتی تمام جداول باید در یک جدول قرار گیرد). در این حالت، مشکلات زیر ایجاد خواهد شد:

۱. افزونگی داده‌ها[۷]، همان‌طور که بیان گردید، افزونگی داده تکرار بی‌مورد داده‌ها است. اگر جداول بانک اطلاعات ترکیب گردند، موجب ایجاد افزونگی داده می‌شوند. ایجاد افزونگی داده اولاً، موجب به هدر رفتن

---

[16] .Attributes [2] .Formal [3].Goodness [4].Conceptual [5].Storage [6].Normalization
[7].Data Redundancy [8] .Anomaly

حافظه می‌گردد و ثانیاً، سرعت پرس‌وجوها را پایین می‌آورد. چون، حجم داده افزایش می‌یابد. بنابراین، کار کردن با داده کندتر انجام خواهد شد. **در نتیجه، نرمال سازی موجب کاهش افزونگی داده** می‌شود.

۲. ایجاد بی‌نظمی[۸]، همان‌طور که بیان گردید، یکی از مهم‌ترین عملیات در بانک اطلاعات دست‌کاری داده (ذخیره یا بازیابی داده) است. دست‌کاری داده از سه بخش زیر تشکیل شده است:

افزودن یک رکورد، فرض کنید بخواهید نمره یک دانشجو را وارد کنید. برای این منظور باید برای هر نمره کلیه اطلاعات دانشجو از قبیل نام درس، تعداد واحد، نوع درس، نام استاد، نام دانشکده، نام دانشجو و ... را وارد نمایید. اگر بخواهید نمره صدها دانشجو را وارد نمایید، این کار نه‌تنها زمان‌بر است، بلکه، گاهی اوقات ممکن نیست. از طرف دیگر، ممکن است موجب ناسازگاری در داده‌ها شود. یعنی، نام استاد، نام دانشکده یا اطلاعات دیگر در رکوردهای مختلف نادرست وارد شوند. این عمل، **بی‌نظمی** نام دارد.

حذف رکوردها، فرض کنید بخواهید یک دانشکده را حذف نمایید. در این صورت باید تمام دانشجویان، همه دروس، تمام اساتید و اطلاعات دیگر که شامل داده‌های مربوط به این دانشکده هستند را حذف کنید. این کار موجب ناسازگاری و بی‌نظمی می‌شود. **اما، اگر جداول نرمال شوند بی‌نظمی و ناسازگاری در حذف رکوردها نیز کاهش می‌یابند.**

ویرایش رکوردها، فرض کنید بخواهید اطلاعات یک درس را تغییر دهید. در این صورت باید تمام رکوردهای مربوط به آن درس را ویرایش نمایید. این امر موجب ناسازگاری و بی‌نظمی خواهد شد. **اما با نرمال سازی جداول، ناسازگاری و بی‌نظمی در ویرایش رکوردها نیز کاهش خواهد یافت.**

۳. ایجاد داده‌های تهی[۱۷]، همان‌طور که می‌دانید یک دانشکده در یک‌ترم همه دروس را ارائه نمی‌کند. بنابراین، برخی از فیلدهای مربوط به دانشجویان و ترم مقدار تهی را می‌پذیرند و داده‌های تهی افزایش می‌یابند. وجود داده‌های تهی برای توابع ریاضی از قبیل Count، Sum، Avg و غیره و عملیات پیوند[۲] موجب ایجاد نتایج غیر قابل پیش‌بینی یا ناخواسته می‌گردد. بنابراین، باید تا حد امکان از ورود داده‌های تهی در جدول اجتناب کرد. **یکی از راه‌های کاهش داده‌های تهی نرمال سازی جداول است.**

۴. مفهوم صفات، هرگاه صفتی در یک جدول (رابطه) قرار می‌گیرد، باید به نوعی بیان کننده صفتی از پدیده مورد نظر باشد. به عبارت دیگر، این صفت باید به همراه صفات دیگر رابطه بیان کننده صفات موجودیت مورد بحث باشد. در حالت کلی می‌توان گفت هر سطر (تاپل) از یک رابطه شامل صفاتی است که یک

---

[17] .Null [2] .Join [3] .Functional dependency

موجودیت کارمند موجودیت سازمان

موجودیت (پدیده) خاص را تعریف می‌کند. به عنوان مثال، رابطه Emp_Dep (واحد کار) را با صفات زیر را در نظر بگیرید:

eName	eNo	depNo	depName	depMgrNo

این رابطه بیان کننده محل کار یک کارمند است. همان‌طور که در این رابطه می‌بینید، صفت depName بیان کننده نام یک سازمان است که نیازی نیست در این رابطه بیان شود. زیرا، هر سطر این رابطه فقط به کد کارمند و کد سازمان برای معرفی یک موجودیت نیاز دارد.

۲ ـ ۵. وابستگی تابعی

قبل از این که به مفهوم وابستگی تابعی[۳] بپردازیم، تعریف تابع را بیان می‌کنیم.

نکته	تابع مجموعه‌ای از زوج‌های مرتب است که اگر مؤلفه اول آن‌ها برابر باشد، مؤلفه دوم آن‌ها نیز برابر باشد.

تعریف ریاضی تابع به صورت زیر است:

فرض کنید $X,Y \in IR$ باشند، در این صورت، $Y = F(X)$ تابع است، **اگر و فقط اگر به ازای هر مقدار X یک مقدار Y ایجاد گردد.** به عنوان مثال، $Y = F(X) = X^2+1$ را در نظر بگیرید. در این رابطه مقدار Y وابسته به مقدار X است.

۱ – ۲ – ۵. تعریف رابطه‌ای وابستگی تابعی

فرض کنید مجموعه صفات (فیلدهای) R به صورت زیر تعریف شوند:

$R = \{ A_1, A_2, A_3, \ldots, A_n\}$

$A_n, \ldots, A_3, A_2, A_1$، تمام فیلدها (صفات) رابطه R هستند. وابستگی تابعی به صورت زیر تعریف می‌شود:

نکته	اگر X و Y دو زیر مجموعه از صفات R باشند، آنگاه، وابستگی تابعی $X \rightarrow Y$، وقتی برقرار است، اگر برای تمام رابطه‌ها در R، به ازای هر مقدار X فقط یک مقدار Y موجود باشد.

به عبارت دیگر، اگر t1 و t2 دو رکورد در R باشند، اگر مجموعه X رکورد t1 برابر مجموعه X رکورد t2 باشد. در این صورت، حتماً مجموعه Y از t1 با مجموعه Y از رکورد t2 برابر است. یعنی:

$t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y]$

به عنوان مثال، چند وابستگی تابعی را در زیر می‌بینید:

چون، هیچ دو دانشجویی شماره دانشجویی یکسان ندارند. **1.**stNo → Fname

۲. courseNo → courseName چون، هیچ دو درس کد یکسان ندارند.

چون، هیچ دو دانشکده‌ای کد یکسان ندارند. **3.**clgNo → clgName

اما، رابطه‌های زیر وابستگی تابعی را تشکیل نمی‌دهند:

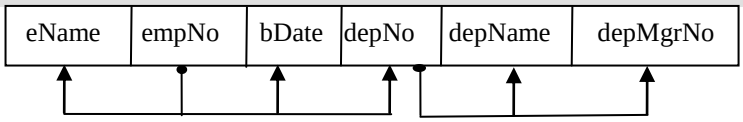
۱. stNo → Score چون، یک دانشجو ممکن است چندین نمره داشته باشد.

۲. Fname → stNo چون، چند دانشجو ممکن است یک نام داشته باشند، اما شماره دانشجوی آن‌ها متفاوت باشد.

۳. stNo → cityCode چون، چند دانشجو ممکن است در یک شهر تحصیل کنند.

۴. stNo, courseNo → Score چون، یک دانشجو ممکن است یک درس را در یک‌ترم پاس نکند، ولی در ترم‌های بعد آن را پاس نماید.

مثال ۱ – ۵. رابطه Emp_Dep را به صورت زیر در نظر بگیرید:

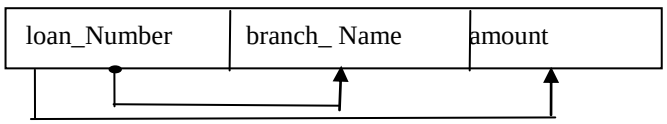


در این رابطه، وابستگی‌های زیر را داریم:

FD1: {empNo}→{eName, bDate, depNo}

FD2: {depNo}→ {depName, depMgrNo}

مثال ۲ – ۵. رابطه Loan را به صورت زیر در نظر بگیرید:



در این رابطه، وابستگی زیر را داریم:

FD3: {loan_Number}→{amount}

FD4: { loan_Number}→ {branch_Name}

۲-۲ – ۵. قوانین استنتاج وابستگی‌های تابعی

معمولاً وابستگی‌های تابعی در ابتدا با استفاده از ارتباط معنایی موجود در تعریف موجودیت‌ها و رابطه‌ها ایجاد می‌گردند. اما، این مجموعه وابستگی‌های تابعی محدود نیستند و با استفاده از **قوانین استنتاج**[۱۸] امکان تولید وابستگی‌های تابعی دیگر هم است.

با فرض این که FD به عنوان مجموعه‌ای از وابستگی تابعی اولیه باشد، در این صورت، مجموعه تمام وابستگی‌های قابل تولید بر مبنای اعضای FD را **توسعه/ بستار**[۲] می‌نامند و با FD^+ نمایش می‌دهند. تولید مجموعه پوششی FD بر اساس قوانین استنتاج است که این قوانین عبارت‌اند از:

۱. **قانون بازتاب**[۳]، اگر x زیر مجموعه y ($\{y\} \subseteq \{x\}$) باشد، آنگاه $\{y\} \rightarrow \{x\}$ برقرار خواهد بود.

۲. **قانون انتقال**[۴]، اگر $\{y\} \rightarrow \{x\}$ و $\{z\} \rightarrow \{y\}$ باشد، آنگاه $\{z\} \rightarrow \{x\}$ برقرار خواهد بود.

۳. **قانون افزایشی**[۵]، اگر $\{y\} \rightarrow \{x\}$ باشد و z یک صفت دیگر در R باشد، آنگاه $\{y, z\} \rightarrow \{x, z\}$ برقرار خواهد بود.

۴. **قانون تجزیه**[۶]، اگر $\{x\} \rightarrow \{y, z\}$ باشد، آنگاه $\{x\} \rightarrow \{y\}$ و $\{x\} \rightarrow \{z\}$ برقرار خواهد بود.

۵. **قانون اجتماع**[۷]، اگر $\{x\} \rightarrow \{y\}$ و $\{x\} \rightarrow \{z\}$ باشد، آنگاه $\{x\} \rightarrow \{y, z\}$ برقرار خواهد بود.

۶. **قانون ترکیب**[۸]، اگر $\{x\} \rightarrow \{y\}$ و $\{w\} \rightarrow \{z\}$ باشد، آنگاه $\{y, w\} \rightarrow \{x, z\}$ برقرار خواهد بود.

۷. **قانون بسته‌بندی**[۹]، اگر $\{x\} \rightarrow \{y\}$ و $\{x, y\} \rightarrow \{z\}$ باشد، آنگاه $\{x\} \rightarrow z$ برقرار خواهد بود.

مثال ۳ – ۵. وابستگی‌های تابعی مثال ۲ – ۵ را در نظر بگیرید. اکنون می‌توان با قوانین استنتاج وابستگی‌های زیر را تولید نمود:

FD = { loan_Number → Amount, loan_Number → branch_Name}

اکنون، قوانین زیر را اعمال می‌کنیم تا FD^+ بدست آید.

FD^+ = { loan_Number → Amount, loan_Number → branch_Name}

loan_Number →{ Amount , branch_Name, loan_Number} قانون اجتماع

loan_Number →{ Amount , branch_Name } قانون اجتماع

loan_Number →{ loan_Number, Amount } قانون افزایش

loan_Number →{ loan_Number, branch_Name } قانون افزایش

---

[18].Inferr deduce [2].Closure [3].Reflexivity [4].Transitivity [5].Augmentation
[6].Decomposition [7].Union [8].Composition [9].Pseudotransitive

نکته	مجموعه وابستگی F مجموعه **وابستگی پوششی** برای وابستگی E است، اگر هر FD از E توسط بستار F بدست آید. در مثال ۳ – ۵، FD^+ به عنوان مجموعه وابستگی پوششی FD است.

نکته	مجموعه وابستگی F و مجموعه وابستگیE، **هم ارز**[19] هستند، اگر مجموعه بستار F و مجموعه بستار E برابر باشند. به عبارت دیگر، هر FD از E را بتوان به کمک FDهای F تولید کرد و بالعکس.

۶ – ۵. پرسش‌های چهارگزینه‌ای

۱. اگر جدول در فرم نرمال BCNF باشد، ممکن است در فرم نرمال دیگر نباشد (ارشد - مهندسی کامپیوتر ۸۶)؟

الف: 1NF　　ب: 2NF　　ج: 3NF　　د: 4NF

۲. در رابطه R(x, y, z, w) ، اگر $x \rightarrow z$ و $y \rightarrow w$ باشد، نامزد کلیدی رابطه R کدام است (ارشد - مهندسی کامپیوتر ۸۷)؟

الف: نامزد کلیدی (x, y) است.　　ب: نامزد کلیدی فقط x یا فقط y است.

ج: نامزد کلیدی (x, y, z, w) است.　　د: x یا y هر یک نامزد کلیدی هستند.

۳. رابطه XYZ به شکل زیر را در نظر بگیرید. این رابطه در چه سطحی از نرمال می‌باشد (ارشد مهندسی IT- سال ۸۵)؟

X	Y	Z
X1	Y1	Z1
X2	Y2	Z2
X3	Y3	Z3
X4	Y4	Z4

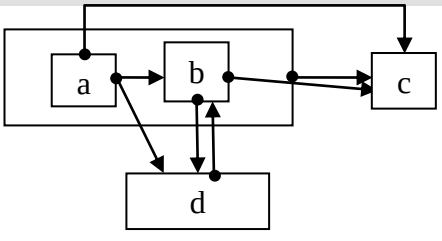
الف: 1NF

ب: 2NF

ج: 3NF

د: BCNF

۴. نمودار FD زیر را در نظر بگیرید. کدام FD متعلق به حداقل FDها است (ارشد مهندسی IT – سال ۸۵)؟



الف: b→c

ب: a→c

ج: a→d

د: ab→c

[19].Equivalent

۵. در رابطه R (A, B, C, D, E) **با مجموعه وابستگی** S، **کدام یک از گزینه‌های زیر نادرست است (ارشد -مهندسی کامپیوتر ۸۵)؟**

$S = \{A \rightarrow C,\ B \rightarrow D,\ AB \rightarrow E\}$

الف: رابطه R، نرمال 2NF است.

ب: رابطه R، نرمال 3NF است.

ج: رابطه R، نرمال 3NF و BCNF می‌باشد.

د: اگر رابطه R به رابطه R1 (A, B, C)، R2 (B, D) و R3 (A, C) تفکیک شود، هر سه رابطه حاصل 3NF و BCNF می‌باشند.

۶. جدول R(x, y, z) **را به دو جدول** R1(x, y) **و** R2(x, y, z) **تجزیه و نرمال شده است. کدام گزینه از شرط‌های صحت این عمل می‌باشد؟** (x→R1، **یعنی، x کلید اصلی** R1 **است) (ارشد - مهندسی کامپیوتر ۸٤).**

الف: x→R1 ب:x→R2 ج:x→R2 و x→R1 د: x→R2 یا x→R1

۷. کدام گزینه از اهداف کلی نرمال سازی (Normalization) **نیست (ارشد- دولتی ۷۵)؟**

الف: تسهیل اعمال بعضی از قواعد جامعیتی (integrity rules).

ب: تسهیل پیاده‌سازی دید کاربر (User view)

ج: حذف بعضی از انواع قواعد وابستگی تابع (Functional dependency).

د: کاهش بعضی از انواع افزونگی (redundency).

۸. کدام یک از گزاره‌های زیر در رابطه با << عمده‌ترین اهداف نرمال سازی رابطه‌ها>> نادرست است (ارشد- مهندسی کامپیوتر ۷۲)؟

الف: کاهش بعضی انواع افزونگی (Redundency).

ب: کاهش سربار (Overhead) سیستم در پاسخ‌گویی در عمل بازیابی.

ج: تسهیل در اعمال بعضی محدودیت‌های جامعیتی (Integrity Constraints).

د: اجتناب از انواع آنومالی در عملیات روی پایگاه داده

۷ - ۵. پاسخ پرسش‌های چهار گزینه‌ای

۱. گزینه (د) صحیح است. زیرا، اگر جدولی فرم نرمال BCNF باشد، حتماً فرم‌های 1NF، 2NF و 3NF است. ولی ممکن است 4NF و 5NF نباشد.

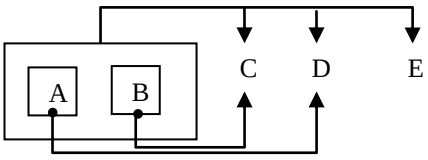
۲. گزینه (الف) صحیح است. زیرا، داریم:

$$\{x \rightarrow z,\ x \rightarrow x,\ y \rightarrow w,\ y \rightarrow y\} \Rightarrow (x, y) \rightarrow (x, y, z, w)$$

۳. گزینه (د) صحیح است. چون، این جدول فاقد عنصر کلیدی می‌باشد. یعنی، هیچ یک از صفات و ترکیب آن‌ها کلید نیست. بنابراین ترکیب XYZ کلید کاندید می‌باشد. پس این وابستگی کامل است.

۴. گزینه (ج) صحیح است. چون، در FD داریم: $FD = \{a \rightarrow b , a \rightarrow c , a \rightarrow d , b \rightarrow d , c \rightarrow d\}$. از طرف دیگر، $a \rightarrow b$ و $b \rightarrow d$ را داریم. پس، $a \rightarrow d$ می‌تواند حذف شود.

۵. گزینه (ج) صحیح است. در این سوال کلید اصلی AB است. چون، A→C و B→D، آنگاه AB→CD نتیجه خواهد شد و AB→E را نیز داریم. پس دیاگرام به صورت زیر بدست می‌آید:



این دیاگرام را ملاحظه کنید، جدول نرم فرم ۲ نمی‌باشد.

پس گزینه‌های الف، ب و ج نادرست است.

۶. گزینه (ج) صحیح است. چون x کلید اصلی R است. پس، با تبدیل R به R1 و R2، وابستگی‌های $x \rightarrow y$ و $x \rightarrow z$ برقرار است. بنابراین، x کلید R1 و R2 است.

۷. گزینه (ب) صحیح است. چون از اهداف نرمال سازی کاهش داده‌های تهی، کاهش بی‌نظمی، کاهش افزونگی داده و تسهیل اعمال برخی از قوانین جامعیت است.

۸. گزینه (ب) صحیح است. چون، نرمال سازی موجب شکستن جدول به چند جدول می‌گردد. پس برای بازیابی اطلاعات باید این جدول با عملگر پیوند با هم Join شوند. این عمل سرعت را پایین می‌آورد.