

فهرست

۵	مقدمه	۱
۶	تمرینات	2
۶	تمرین ۱	2.1
۶	معماری پایگاه داده	2.1.1
۶	معماری توزیع شده	2.1.2
۶	معماری توزیع شده همگن	2.1.3
۷	معماری توزیع شده ناهمگن	2.1.4
۷	مزایا و معایب سیستم‌های توزیع شده همگن و ناهمگن	2.1.5
۷	معماری چند پایگاهی	2.1.6
۸	نرم‌افزارهای شبکه‌ای	2.2
۸	Network monitoring	2.2.1
۸	Solarwinds	2.2.2
۸	Datalog	2.2.3
۹	تمرین ۲	2.3
۹	نرم‌افزار Rational Rose	2.3.1
۹	مشکلات نرم‌افزار	2.3.2
۱۰	تمرین ۳	2.4
۱۰	عملگرهای جبر رابطه‌ای	2.4.1
۱۰	کاربرد عملگرهای جبر رابطه‌ای در پایگاه‌های داده	2.4.2
۱۰	عملگر انتخاب (Select)	2.4.3
۱۰	عملگر فرافکنی (Project)	2.4.4
۱۰	ترتیب‌بندی (Order by)	2.4.5
۱۱	گروه‌بندی (Group by having)	2.4.6
۱۱	عملگر اجتماع (Union)	2.4.7
۱۱	عملگر اشتراک (Intersect)	2.4.8
۱۱	عملگر تفاضل (Except)	2.4.9
۱۲	عملگر ضرب دکارتی (Cross Join)	2.4.10
۱۲	عملگر الحق داخلي (Inner Join)	2.4.11
۱۲	عملگر الحق كامل (Full Outer Join)	2.4.12

۱۲.....	عملگر (AS)	2.4.13
۱۲.....	عملگر الحق یا ترکیب رشته‌ها (+)	2.4.14
۱۳.....	عملگر تقسیم (Division)	۲.۴.۱۵
۱۴.....	کاربرد و استفاده از tinyint در الگوریتم‌های یادگیری ماشین	۲.۵
۱۴.....	تعریف	۲.۵.۱
۱۴.....	Tinyint در یادگیری ماشین و کانولوشن‌های دو بعدی	2.5.2
۱۵.....	تمرین ۴	۲.۶
۱۶.....	تمرین ۵	۲.۷
۱۷.....	تمرین ۶	2.8
۱۸.....	تمرین ۷	۲.۹
۱۹.....	تمرین ۸	۲.۱۰
۲۰.....	ORM	2.11
۲۰.....	نگاشت شیء-رابطه‌ای	2.11.1
۲۰.....	پیاده‌سازی	۲.۱۱.۲
۲۱.....	پروژه	۳
۲۱.....	معرفی پایگاه داده	۳.۱
۲۱.....	معرفی جداول اصلی	۳.۲
۲۱.....	جدول driver (راننده):	۳.۲.۱
۲۲.....	جدول truck (کامیون):	۳.۲.۲
۲۲.....	جدول trailer (تریلر):	۳.۲.۳
۲۲.....	جدول cargo (بار):	۳.۲.۴
۲۳.....	جدول company (شرکت):	۳.۲.۵
۲۳.....	معرفی جداول رابطه‌ای	3.3
۲۳.....	جدول attachtrailer (اتصال تریلر):	۳.۳.۱
۲۳.....	جدول driving (رانندگی):	۳.۳.۲
۲۳.....	جدول loading (بارگیری):	۳.۳.۳
۲۴.....	جدول hiring (استخدام):	۳.۳.۴
۲۴.....	مزایا و معایب پایگاه داده	۳.۴
۲۴.....	نکات مثبت و مزایا:	۳.۴.۱
۲۴.....	نکات منفی و معایب:	۳.۴.۲

۲۵	طرح مسائل	۳.۵
۲۵	مسئله ۱	۳.۵.۱
۲۵	مسئله ۲	۳.۵.۲
۲۵	مسئله ۳	۳.۵.۳
۲۶	مسئله ۴	۳.۵.۴
۲۶	مسئله ۵	۳.۵.۵
۲۶	مسئله ۶	۳.۵.۶
۲۷	مسئله ۷	۳.۵.۷
۲۷	مسئله ۸	۳.۵.۸
۲۷	مسئله ۹	۳.۵.۹
۲۸	مسئله ۱۰	۳.۵.۱۰
۲۸	پیاده‌سازی ORM	3.6

۱ مقدمه

پایگاه داده‌ها به عنوان مخازن عظیمی از اطلاعات ساختاریافته، نقشی حیاتی در دنیای مدرن ایفا می‌کنند. این پایگاه‌ها داده‌ها را به صورت سازمان‌یافته ذخیره، مدیریت و بازیابی می‌کنند تا دسترسی سریع و آسان به اطلاعات را فراهم آورند. از وبسایت‌های تجارت الکترونیک گرفته تا سیستم‌های بانکی، پایگاه داده‌ها در پس‌صحنه‌ی تقریباً تمامی عملیات‌های دیجیتالی حضور دارند.

یک پایگاه داده از جداول تشکیل شده است که هر جدول شامل سطراها (رکوردها) و ستون‌ها (فیلدها) است. سطراها نماینده‌ی یک رکورد یا یک واحد اطلاعات هستند، در حالی که ستون‌ها آن مشخصات رکورد را تعریف می‌کنند. برای تعامل با پایگاه داده‌ها از زبان‌های پرس‌وچوی ساختاریافته (SQL) استفاده می‌شود. SQL به کاربران اجازه می‌دهد تا داده‌ها را جستجو، دستکاری و بازیابی کنند.

در این گزارش کار، به بررسی یک پروژه عملی با استفاده از پایگاه داده‌ی Microsoft SQL Server پرداخته‌ایم. در این پروژه، یک پایگاه داده طراحی و پیاده‌سازی شده است که به منظور یک شرکت حمل و نقل بار در کشور دانمارک، ایجاد شده است. برای دستیابی به این هدف، از زبان پرس‌وچوی ساختاریافته (SQL) برای نوشتن کوئری‌های مختلف استفاده شده است. کوئری‌های نوشته شده به ما امکان می‌دهند تا داده‌ها را جستجو، دستکاری و تحلیل کنیم.

در کنار پیاده‌سازی پروژه اصلی، تمرین‌های مختلفی نیز برای درک بهتر مفاهیم پایگاه داده و زبان SQL انجام شده است. این تمرین‌ها شامل تعریف معماری پایگاه‌های داده، ترسیم نمودار ER و نرم‌افزارهای شبکه‌ای می‌باشند. حل این تمرین‌ها به ما کمک کرده است تا مهارت‌هایمان را در زمینه طراحی و مدیریت پایگاه داده بهبود بخشیم.

۲ تمرینات

۲/۱ تمرین ۱

معماری توزیع شده همگن (Homogeneous) و ناهمگن (Heterogeneous) را به طور کامل تشریح و مزایای آنها را بیان نمایید.

۲/۱/۱ معماری پایگاه داده

معماری پایگاه داده به عنوان چارچوبی برای طراحی و پیاده‌سازی پایگاه داده‌ها عمل می‌کند. این معماری به سه سطح اصلی تقسیم می‌شود: ادراکی، خارجی و داخلی. هر یک از این سطوح نقش مهمی در تعریف و مدیریت داده‌ها ایفا می‌کنند.

- سطح ادراکی (Conceptual Level): این سطح بالاترین سطح انتزاع است و دید کلی از داده‌ها را ارائه می‌دهد. در این سطح، داده‌ها به صورت انتزاعی و مستقل از هرگونه پیاده‌سازی فیزیکی نمایش داده می‌شوند. سطح ادراکی معمولاً با استفاده از مدل‌های داده‌ای مانند مدل‌های موجودیت-ارتبط (ER) توصیف می‌شود. در این سطح، روابط بین موجودیت‌ها، ویژگی‌های آن‌ها و قواعد کسبوکار تعریف می‌شوند. این سطح به کاربران نهایی کمک می‌کند تا درک درستی از داده‌ها و نحوه ارتباط آن‌ها با یکدیگر داشته باشند.
- سطح خارجی (External Level): این سطح به کاربران مختلف اجازه می‌دهد تا دیدگاه‌های متفاوتی از پایگاه داده داشته باشند. هر کاربر می‌تواند یک نمای (View) شخصی‌سازی شده از داده‌ها داشته باشد که تنها شامل اطلاعات مورد نیاز او است. این سطح به امنیت داده‌ها و حفظ حریم خصوصی کمک می‌کند. به عنوان مثال، یک فروشنده ممکن است تنها به اطلاعات مربوط به محصولات و مشتریان دسترسی داشته باشد، در حالی که یک مدیر ممکن است به تمامی داده‌های پایگاه داده دسترسی داشته باشد.
- سطح داخلی (Internal Level): این سطح پایین‌ترین سطح انتزاع است و به نحوی ذخیره‌سازی فیزیکی داده‌ها در پایگاه داده می‌پردازد. در این سطح، ساختارهای داده‌ای، شاخص‌ها، فایل‌ها و روش‌های دسترسی به داده‌ها تعریف می‌شوند. هدف اصلی این سطح، بهینه‌سازی عملکرد پایگاه داده از نظر سرعت دسترسی و استفاده از فضای ذخیره‌سازی است. انتخاب ساختارهای داده‌ای مناسب و ایجاد شاخص‌های مؤثر بر روی عملکرد پایگاه داده تأثیر مستقیم دارد.

در مجموع، معماری سه سطحی پایگاه داده به ایجاد یک جداسازی بین دیدگاه‌های مختلف از داده‌ها کمک می‌کند. این جداسازی به بهبود قابلیت نگهداری، امنیت و انعطاف‌پذیری پایگاه داده‌ها منجر می‌شود. همچنین، این معماری به کاربران اجازه می‌دهد تا با توجه به نیازهای خود با پایگاه داده تعامل داشته باشند.

۲/۱/۲ معماری توزیع شده

معماری توزیع شده روشی برای طراحی سیستم‌های نرم‌افزاری است که در آن پردازش داده‌ها و منابع به چندین کامپیوتر یا سرور پراکنده در یک شبکه توزیع می‌شود. این رویکرد در مقابل معماری متمرکز قرار دارد که تمام پردازش‌ها در یک کامپیوتر واحد انجام می‌شود.

۲/۱/۳ معماری توزیع شده همگن

در یک معماری توزیع شده همگن، تمام گره‌های شبکه از نظر سخت‌افزار، سیستم‌عامل و نرم‌افزار یکسان هستند. این بدان معناست که تمام گره‌ها از یک نوع پایگاه داده، زبان برنامه‌نویسی و پروتکل ارتباطی استفاده می‌کنند. این یکپارچگی باعث می‌شود مدیریت سیستم ساده‌تر شود و احتمال بروز خطا کاهش یابد.

۲/۱/۴ معماری توزیع شده ناهمگن

در یک معماری توزیع شده ناهمگن، گره های شبکه می توانند از نظر سخت افزار، سیستم عامل و نرم افزار متفاوت باشند. این به این معنی است که هر گره می تواند از یک نوع پایگاه داده، زبان برنامه نویسی و پروتکل ارتباطی متفاوت استفاده کند. این نوع معماری انعطاف پذیری بیشتری را فراهم می کند، اما مدیریت آن پیچیده تر است.

۲/۱/۵ مزايا و معایب سیستم های توزیع شده همگن و ناهمگن

معماری توزیع شده همگن به دلیل یکسانی گره های شبکه، مزايا قابل توجهی دارد. مدیریت و نگهداری این سیستم ها بسیار ساده تر است. همچنان، قابلیت اطمینان بالای دارند، زیرا در صورت خرابی یک گره، می توان آن را با یک گره جایگزین کرد. هزینه های نگهداری نیز به دلیل استفاده از سخت افزار و نرم افزار یکسان، کاهش می یابد. با این حال، این نوع معماری انعطاف پذیری کمتری دارد و ممکن است برای سیستم های پیچیده مناسب نباشد.

معماری توزیع شده ناهمگن انعطاف پذیری بالایی را ارائه می دهد. امکان استفاده از بهترین فناوری برای هر بخش از سیستم وجود دارد و قابلیت استفاده مجدد از اجزای موجود در سیستم های دیگر نیز فراهم می شود. با این حال، این نوع معماری پیچیدگی بالایی دارد و مدیریت و نگهداری آن دشوار تر است. هزینه های توسعه و نگهداری سیستم نیز بالاتر است. همچنان، یکپارچه سازی اجزای مختلف سیستم دشوار تر است.

۲/۱/۶ معماری چند پایگاهی

معماری چند پایگاهی روشنی برای مدیریت داده ها است که در آن اطلاعات در چندین پایگاه داده مجزا ذخیره می شوند. این پایگاه داده ها ممکن است از نظر نوع، ساختار و مکان فیزیکی با هم متفاوت باشند. هدف اصلی از استفاده از معماری چند پایگاهی، بهبود قابلیت اطمینان، مقیاس پذیری و انعطاف پذیری سیستم است.

در این معماری، هر پایگاه داده به صورت مستقل عمل می کند و داده های خاصی را ذخیره می کند. برای مثال، یک سازمان ممکن است یک پایگاه داده برای ذخیره اطلاعات مشتریان، یک پایگاه داده برای ذخیره اطلاعات محصولات و یک پایگاه داده برای ذخیره اطلاعات سفارشات داشته باشد. با این روش، می توان به هر بخش از داده ها به صورت جداگانه دسترسی پیدا کرد و مدیریت کرد. همچنان، در صورت بروز مشکل در یک پایگاه داده، سایر پایگاه داده ها همچنان به کار خود ادامه می دهند و از دست رفتن اطلاعات به حداقل می رسد.

از مزاياي دیگر معماری چند پایگاهی می توان به افزایش عملکرد سیستم، بهبود قابلیت مدیریت و کاهش هزینه های نگهداری اشاره کرد. با توزیع داده ها در چندین پایگاه داده، می توان بار کاری را بین آن ها تقسیم کرد و از ایجاد تنگنا در سیستم جلوگیری کرد. همچنان، هر پایگاه داده را می توان به صورت مستقل به روزرسانی و نگهداری کرد. با این حال، پیاده سازی و مدیریت یک سیستم چند پایگاهی پیچیدگی های خاص خود را دارد و نیاز به طراحی دقیق و برنامه ریزی مناسب دارد.

۲/۲ نرم افزارهای شبکه‌ای

درباره یکی از نرم افزارهای شبکه‌ای Network Monitoring از جمله Datadog و SolarWinds تحقیق کرده و گزارش آن را به کلاس ارائه دهد. (تمرین اختیاری)

۲/۲/۱ Network monitoring

۲/۲/۲ Solarwinds

SolarWinds یکی از شناخته شده‌ترین و پرکاربردترین نرم افزارهای نظارت بر شبکه در جهان است. این ابزار با ارائه مجموعه‌ای جامع از ویژگی‌ها و ابزارهای مدیریت شبکه، به مدیران شبکه کمک می‌کند تا عملکرد شبکه خود را بهینه کنند، مشکلات را شناسایی کرده و از بروز اختلالات جلوگیری کنند.

SolarWinds با جمع‌آوری داده از انواع مختلف دستگاه‌های شبکه مانند روترها، سوئیچ‌ها، سرورها و ایستگاه‌های کاری، به صورت مداوم وضعیت شبکه را پایش می‌کند. این داده‌ها شامل اطلاعاتی در مورد ترافیک شبکه، استفاده از منابع، خطاهای و رویدادهای مهم است. سپس، SolarWinds این داده‌ها را تحلیل کرده و به صورت گرافیکی و جدولی نمایش می‌دهد. این نرم افزار همچنین به مدیران شبکه اجازه می‌دهد تا آلارم‌ها و هشدارهای مهم تعریف کنند تا در صورت بروز هرگونه مشکل، به سرعت مطلع شوند.

یکی از ویژگی‌های برجسته SolarWinds، قابلیت خودکارسازی بسیاری از وظایف مدیریت شبکه است. به عنوان مثال، SolarWinds می‌تواند به صورت خودکار مشکلات شبکه را تشخیص داده و اقدامات اصلاحی را انجام دهد. همچنین، این نرم افزار می‌تواند گزارش‌های دقیقی از وضعیت شبکه ایجاد کند که به مدیران شبکه کمک می‌کند تا عملکرد شبکه را تحلیل کرده و تصمیمات آگاهانه‌ای بگیرند. SolarWinds همچنین دارای مازولهای تخصصی برای نظارت بر زیرساخت‌های خاص مانند Active Directory، پایگاه داده‌های SQL Server و محیط‌های مجازی است.

در کل، SolarWinds یک ابزار قدرتمند و جامع برای نظارت و مدیریت شبکه است که به مدیران شبکه کمک می‌کند تا عملکرد شبکه خود را بهینه کنند، مشکلات را به سرعت شناسایی کرده و از بروز اختلالات جلوگیری کنند.

۲/۲/۳ Datalog

Datalog یک زبان برنامه‌نویسی اعلانی و منطقی است که به طور ویژه برای پرسش و پاسخ در پایگاه داده‌های استنتاجی طراحی شده است. این زبان اجازه می‌دهد تا با استفاده از قواعد منطقی، اطلاعات را از پایگاه داده استخراج و روابط پیچیده بین داده‌ها را بیان کند. به عبارت ساده‌تر، دیتالاگ به شما این امکان را می‌دهد تا به پایگاه داده‌تان سوالاتی بپرسید و پاسخ‌های منطقی دریافت کنید.

یکی از ویژگی‌های مهم دیتالاگ، سادگی و خوانایی آن است. سینتکس دیتالاگ بسیار شیوه به زبان طبیعی است و به همین دلیل، نوشتن پرسش‌ها و قواعد در آن بسیار آسان است. همچنین، دیتالاگ از یک مدل استنتاجی قدرتمند استفاده می‌کند که به آن اجازه می‌دهد تا اطلاعات جدید را از اطلاعات موجود استخراج کند. این ویژگی، دیتالاگ را به یک ابزار قدرتمند برای تحلیل داده‌ها و کشف الگوها تبدیل کرده است.

کاربردهای دیتالاگ بسیار گسترده است. از جمله این کاربردها می‌توان به پایگاه داده‌های استنتاجی، هوش مصنوعی، شبکه‌های معنایی و تحلیل داده اشاره کرد. در پایگاه داده‌های استنتاجی، دیتالاگ برای پرس و جو و استنتاج اطلاعات استفاده می‌شود. در هوش مصنوعی، دیتالاگ برای بیان داشش و استدلال در سیستم‌های خبره و موتورهای استنتاج به کار می‌رود. در شبکه‌های معنایی، دیتالاگ برای پرس و جو در شبکه‌های معنایی و استخراج اطلاعات از داده‌های ساختاریافته استفاده می‌شود. و در نهایت، در تحلیل داده، دیتالاگ می‌تواند برای تحلیل داده‌های پیچیده و کشف الگوها به کار رود.

به طور خلاصه، دیتالاگ یک زبان برنامه‌نویسی قدرتمند و منعطف است که برای پرسش و پاسخ در پایگاه داده‌های استنتاجی و کاربردهای مختلف هوش مصنوعی مورد استفاده قرار می‌گیرد. سادگی، قدرت بیان و قابلیت استنتاج از جمله مهم‌ترین ویژگی‌های این زبان هستند.

۲/۳ تمرین ۲

با استفاده از نرم افزار Rational Rose محیط عملیاتی مربوط به پروژه خود را مدل سازی نمایید.

۲/۳/۱ نرم افزار Rational Rose

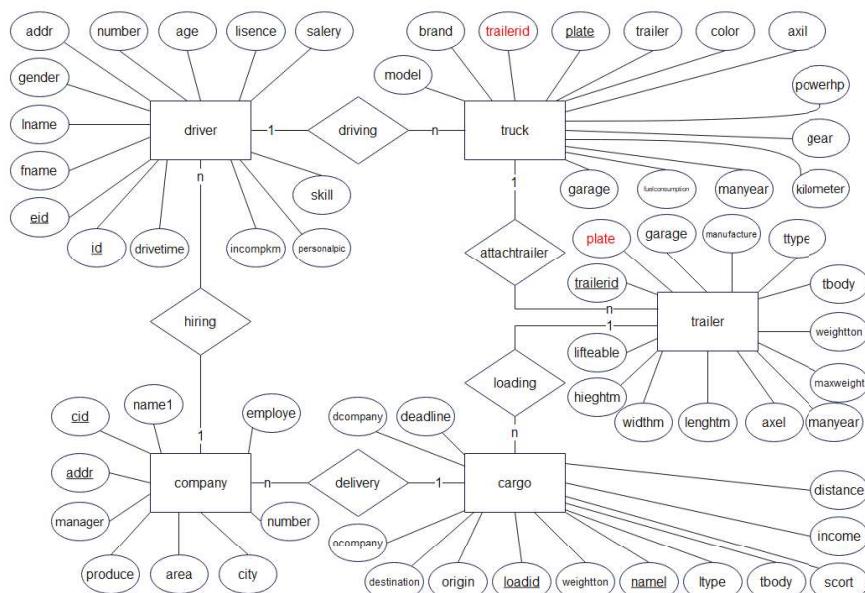
یکی از نرم افزارهای پر کاربرد در زمینه مهندسی نرم افزار است که برای مدل سازی و طراحی سیستم های نرم افزاری مورد استفاده قرار می گیرد. این ابزار با ارائه محیطی گرافیکی و امکانات متنوع، به تحلیل کردن و طراحان سیستم کمک می کند تا طرح های اولیه سیستم را به صورت تصویری و دقیق ایجاد کرده و آن ها را به سایر اعضای تیم توسعه منتقل کنند.

UML یک زبان استاندارد برای مدل سازی سیستم های نرم افزاری است که به کمک آن می توان اجزای مختلف سیستم، روابط بین آن ها و رفتار سیستم را به صورت نمودارهای مختلف نمایش داد. Rational Rose با استفاده از UML، به طراحان اجزاء می دهد تا مدل های کاملی از سیستم را ایجاد کنند که شامل نمودارهای کلاس، نمودارهای توالی، نمودارهای حالت و سایر نمودارهای مورد نیاز برای درک کامل سیستم باشند.

از جمله مزایای استفاده از Rational Rose می توان به افزایش کیفیت نرم افزار، کاهش هزینه های توسعه، بهبود ارتباط بین اعضای تیم، و تسهیل در نگهداری و توسعه سیستم اشاره کرد. با استفاده از این ابزار، می توان خطاهای طراحی را در مراحل اولیه شناسایی و رفع کرد و از این طریق از بروز مشکلات جدی در مراحل بعدی توسعه جلوگیری کرد. همچنین، Rational Rose به عنوان یک ابزار مستند سازی قوی عمل می کند و به تیم توسعه کمک می کند تا مستندات فنی کاملی از سیستم ایجاد کنند.

۲/۳/۲ مشکلات نرم افزار

با توجه به اینکه نرم افزار Rational Rose یک ابزار قدیمی محسوب می شود، برای نصب و استفاده از آن نیاز به خریداری مجوز و اشتراک بوده است. این امر باعث محدودیت دسترسی بسیاری از کاربران به این نرم افزار شده است. به همین دلیل، بسیاری از کاربران ترجیح داده اند تا از نرم افزارهای ترسیم آنلاین استفاده کنند که اغلب رایگان بوده و نیازی به نصب و پیکربندی پیچیده ندارند. این نرم افزارهای آنلاین امکانات مشابهی را برای مدل سازی و طراحی سیستم ها ارائه می دهند و به کاربران اجازه می دهند تا از هر دستگاهی با اتصال به اینترنت به پروژه های خود دسترسی داشته باشند.



شکل ۱ - نمودار ER

۲/۴ تمرین ۳**۲/۴/۱ عملگرهای جبر رابطه‌ای**

جبر رابطه‌ای، زبان رسمی برای بیان پرس‌وجوها و دستکاری داده‌ها در پایگاه‌های داده رابطه‌ای است. این جبر، مجموعه‌ای از عملگرها را ارائه می‌دهد که به کاربران امکان می‌دهند تا داده‌ها را انتخاب، فیلتر، مرتب‌سازی، ترکیب و تحلیل کنند. این عملگرها به عنوان بلوک‌های سازنده پرس‌وجوهای پیچیده در زبان‌های پرس‌وجوی ساخت‌یافته مانند SQL عمل می‌کنند.

عملگرهای جبر رابطه‌ای به دو دسته اصلی تقسیم می‌شوند: عملگرهای یکانی و عملگرهای دوتایی. عملگرهای یکانی بر روی یک رابطه عمل می‌کنند، در حالی که عملگرهای دوتایی بر روی دو رابطه عمل می‌کنند. برخی از مهم‌ترین عملگرهای جبر رابطه‌ای عبارتند از: انتخاب (Selection)، فرافکنی (Projection)، اتحاد (Union)، تفاضل (Set Difference)، ضرب دکارتی (Cartesian Product)، پیوند (Join) و گروه‌بندی (Grouping). عملگر انتخاب به کاربران اجازه می‌دهد تا سطرهایی را که شرایط خاصی را برآورده می‌کنند، از یک رابطه انتخاب کنند. عملگر فرافکنی ستون‌های خاصی را از یک رابطه انتخاب می‌کند. عملگرهای اتحاد، تفاضل و ضرب دکارتی برای ترکیب روابط به روش‌های مختلف استفاده می‌شوند. عملگر پیوند برای ترکیب سطرهای دو رابطه بر اساس یک شرط مشترک به کار می‌رود. عملگر گروه‌بندی برای گروه‌بندی سطرهای یک رابطه بر اساس یک یا چند ستون و اعمال توابع تجمعی بر روی هر گروه استفاده می‌شود.

در این بخش به تعریف و انجام جبرهای رابطه‌ای می‌پردازیم. نتیجه علمی و اصلی هر جبر رابطه‌ای در بخش ۳.۵ تست و پیاده‌سازی شده است.

۲/۴/۲ کاربرد عملگرهای جبر رابطه‌ای در پایگاه‌های داده

عملگرهای جبر رابطه‌ای نقش بسیار مهمی در مدیریت و تحلیل داده‌ها در پایگاه‌های داده رابطه‌ای ایفا می‌کنند. با استفاده از این عملگرها، کاربران می‌توانند پرس‌وجوهای پیچیده و متنوعی را برای استخراج اطلاعات مورد نظر خود از پایگاه داده طراحی کنند. به عنوان مثال، می‌توانند داده‌ها را بر اساس معیارهای خاصی فیلتر کنند، داده‌های مختلف را ترکیب کنند، گزارش‌های خلاصه‌ای ایجاد کنند و روندهای موجود در داده‌ها را شناسایی کنند. علاوه بر این، عملگرهای جبر رابطه‌ای به عنوان پایه و اساس بسیاری از زبان‌های پرس‌وجوی ساخت‌یافته مانند SQL عمل می‌کنند. بنابراین، درک عمیق این عملگرها برای هر کسی که با پایگاه‌های داده رابطه‌ای کار می‌کند، ضروری است.

۲/۴/۳ عملگر انتخاب (Select)

عملگر انتخاب، یکی از اساسی‌ترین عملگرها در جبر رابطه‌ای است که اجازه می‌دهد سطرهایی را از یک جدول که شرایط خاصی را برآورده می‌کنند، انتخاب شوند. به عبارت دیگر، می‌توان با استفاده از این عملگر، داده‌هایی را از یک جدول که دارای ویژگی‌های مشخصی هستند، فیلتر کرد. این عملگر به صورت نمادین با حرف سیگما (σ) نشان داده می‌شود.

۲/۴/۴ عملگر فرافکنی (Project)

عملگر فرافکنی، این اجازه را می‌دهد تا ستون‌های خاصی را از یک جدول انتخاب شوند. به عبارت دیگر، می‌توان با استفاده از این عملگر، داده‌های مورد نظر را از یک جدول استخراج و ستون‌های اضافی را حذف کرد. این عملگر به صورت نمادین با حرف بی (π) نشان داده می‌شود.

۲/۴/۵ ترتیب‌بندی (Order by)

عملگر Order By، یکی از عملگرها در جبر رابطه‌ای و زبان‌های پرس‌وجوی ساخت‌یافته مانند SQL است که به منظور مرتب‌سازی نتایج یک پرس‌جو بر اساس یک یا چند ستون به کار می‌رود. این عملگر به کاربر اجازه می‌دهد تا داده‌های خروجی را به ترتیب صعودی یا نزولی مرتب کند که این امر در تحلیل داده‌ها و ارائه گزارش‌های منظم بسیار مفید است. به طور مثال، می‌توان نتایج

یک پرس‌و‌جواب بر اساس ستون تاریخ به ترتیب نزولی مرتب کرد تا جدیدترین داده‌ها در ابتداء قرار گیرند یا بر اساس ستون نام به ترتیب صعودی مرتب کرد تا لیستی الفبایی از نتایج حاصل شود.

۲/۴/۶ گروه‌بندی (Group by having)

عملگرهای Having و Group By دو عملگر قدرتمند در جبر رابطه‌ای و زبان‌های پرس‌و‌جوابی ساخت‌یافته مانند SQL هستند که برای تجزیه و تحلیل داده‌ها بر اساس گروه‌ها به کار می‌روند. عملگر Group By به کاربر اجازه می‌دهد تا سطرهای یک جدول را بر اساس یک یا چند ستون گروه‌بندی کند. پس از گروه‌بندی، عملگر Having امکان اعمال شرط بر روی گروه‌ها را فراهم می‌کند تا تنها گروه‌هایی که شرط مشخصی را برآورده می‌کنند، در نتیجه نهایی نمایش داده شوند. به عنوان مثال، می‌توان با استفاده از این عملگرهای گروه‌بندی که شرط مشخصی را برآورده می‌کنند، در نتیجه نهایی نمایش داد که بیش از ۱۰۰ دانشجو داشته باشند.

۲/۴/۷ عملگر اجتماع (Union)

عملگر اجتماع یکی از عملگرهای بنیادی در جبر رابطه‌ای است که برای ترکیب دو یا چند جدول با ساختار ستونی مشابه به کار می‌رود. این عملگر سطرهایی را که در هر دو جدول یا در یکی از آن‌ها وجود دارند، در یک جدول جدید و منسجم گردآوری می‌کند. به عبارت دیگر، عملگر اتحاد، تمامی سطرهای منحصر به فرد را از دو یا چند جدول انتخاب کرده و آن‌ها را در یک جدول جدید قرار می‌دهد.

شرط لازم برای استفاده از عملگر اتحاد این است که تعداد و نوع داده‌ای ستون‌های جداول باید یکسان باشد و معمولاً سطرهای تکراری در نتیجه نهایی حذف می‌شوند.

با استفاده از عملگر اتحاد می‌توان اطلاعاتی را از منابع مختلف ترکیب کرده و یک دید کلی از داده‌ها به دست آورد. به عنوان مثال، می‌توان نتایج جستجو در دو جدول مختلف را با استفاده از عملگر اتحاد ترکیب کرد تا یک نتیجه جامع‌تر به کاربر ارائه شود.

۲/۴/۸ عملگر اشتراک (Intersect)

عملگر اشتراک (Intersect) در جبر رابطه‌ای برای یافتن سطرهای مشترک بین دو یا چند جدول استفاده می‌شود. به عبارت دیگر، این عملگر سطرهایی را بر می‌گرداند که هم در جدول اول و هم در جدول دوم (یا همه جداول مشخص شده) وجود دارند. نتیجه‌ی عملگر Intersect جدولی جدید است که شامل فقط سطرهای مشترک بین تمام جداول ورودی است.

برای استفاده از عملگر Intersect، تمامی جداول باید ساختار ستونی مشابهی داشته باشند. این عملگر شبیه به عملگر اشتراک در ریاضیات است و از آن برای پیدا کردن عناصر مشترک بین دو یا چند مجموعه استفاده می‌شود.

۲/۴/۹ عملگر تفاضل (Except)

عملگر تفاضل یا Except یکی از عملگرهای مجموعه‌ای در جبر رابطه‌ای است که برای یافتن تفاوت بین دو مجموعه استفاده می‌شود. به عبارت دیگر، این عملگر سطرهایی را از یک جدول انتخاب می‌کند که در جدول دیگر وجود ندارند. این عملگر شبیه به عملگر تفاضل در ریاضیات است.

برای استفاده از عملگر Except، دو جدول با ساختار ستونی یکسان نیاز است. نتیجه‌ی این عملگر، جدولی جدید است که شامل تمام سطرهایی است که در جدول اول وجود دارند، اما در جدول دوم وجود ندارند. از این عملگر می‌توان برای مقایسه‌ی دو مجموعه داده و یافتن تفاوت‌های بین آن‌ها استفاده کرد. به عنوان مثال، می‌توان از عملگر Except برای پیدا کردن مشتریانی که در یک ماه خاص خرید انجام داده‌اند اما در ماه قبل خریدی نداشته‌اند، استفاده کرد.

۲/۴/۱۰ عملگر ضرب دکارتی (Cross Join)

عملگر ضرب دکارتی یا Cross Join یکی از عملگرهای بنیادی در جبر رابطه‌ای است که برای ترکیب تمامی سطرهای یک جدول با تمامی سطرهای جدول دیگر به کار می‌رود. به عبارت دیگر، این عملگر تمامی ترکیب‌های ممکن بین سطرهای دو جدول را تولید می‌کند. نتیجه این عملگر جدولی جدید است که تعداد سطرهای آن حاصل ضرب تعداد سطرهای دو جدول اولیه است.

کاربرد اصلی عملگر ضرب دکارتی در مواردی است که می‌خواهیم تمامی ترکیب‌های ممکن بین دو یا چند جدول را بررسی کنیم. با این حال، به دلیل افزایش سییار زیاد تعداد سطرهای جدول نتیجه، استفاده از این عملگر باید با احتیاط انجام شود و معمولاً در ترکیب با سایر عملگرها مانند where برای فیلتر کردن نتایج استفاده می‌شود.

۲/۴/۱۱ عملگر الحق داخلی (Inner Join)

عملگر الحق داخلی یا Inner Join یکی از پرکاربردترین عملگرهای در جبر رابطه‌ای است که برای ترکیب سطرهای دو جدول بر اساس یک یا چند ستون مشترک به کار می‌رود. در این نوع الحق، تنها سطرهایی از دو جدول در نتیجه قرار می‌گیرند که در ستون مشترک (یا ستون‌های مشترک) مقدار یکسانی داشته باشند. به عبارت دیگر، Inner Join سطرهایی را برمی‌گرداند که در هر دو جدول وجود دارند.

از عملگر Inner Join زمانی استفاده می‌شود که بخواهیم اطلاعات مرتبط از دو یا چند جدول را با هم ترکیب کنیم. به عنوان مثال، برای یافتن نام مشتریان و سفارشات آن‌ها، می‌توانیم جدول مشتریان را با جدول سفارشات بر اساس شناسه مشترک مشتری الحق کنیم.

۲/۴/۱۲ عملگر الحق کامل (Full Outer Join)

عملگر الحق کامل یا Full Outer Join یکی از انواع الحق در جبر رابطه‌ای است که تمامی سطرهای موجود در دو جدول را در نتیجه قرار می‌دهد. به عبارت دیگر، این عملگر هم سطرهای مشترک بین دو جدول و هم سطرهایی که تنها در یکی از جداول وجود دارند را در نظر می‌گیرد.

در مواردی که بخواهیم تمامی اطلاعات مربوط به دو جدول را بدون از دست دادن هیچ سطrix ترکیب کنیم، از الحق کامل استفاده می‌شود. در نتیجه این عملگر، سطرهایی که در یک جدول وجود دارند اما در جدول دیگر وجود ندارند، با مقادیر NULL در ستون‌های جدول دیگر پر می‌شوند. این عملگر به خصوص زمانی مفید است که بخواهیم تفاوت‌های بین دو جدول را مشخص کنیم یا تمامی اطلاعات مرتبط با دو جدول را در یک جدول واحد داشته باشیم.

۲/۴/۱۳ عملگر (AS)

عملگر AS در جبر رابطه‌ای و زبان‌های پرس‌وجوی ساخت‌یافته مانند SQL برای ایجاد نام مستعار (alias) برای ستون‌ها، جدول‌ها یا نتایج عبارات استفاده می‌شود. این عملگر اجازه می‌دهد نامی دلخواه و معنادار را به جای نام اصلی یک ستون یا جدول قرار دهد تا خوانایی و درک پرس‌وجوی را افزایش دهد. همچنین، می‌توان از عملگر AS برای ایجاد نام مستعار برای یک جدول فرعی (subquery) یا یک عبارت محاسباتی استفاده شود. این کار باعث می‌شود که پرس‌وجوی شما ساختارمندتر و قابل فهم‌تر شود و از تکرار عبارات طولانی جلوگیری می‌کند.

۲/۴/۱۴ عملگر الحق یا ترکیب رشته‌ها (+)

عملگر الحق یا ترکیب رشته‌ها (+) در زبان‌های پرس‌وجوی ساخت‌یافته مانند SQL برای ترکیب دو یا چند رشته به یکدیگر استفاده می‌شود. این عملگر رشته‌های ورودی را به ترتیب از چپ به راست به هم می‌چسباند و یک رشته‌ی جدید ایجاد می‌کند.

۲/۴/۱۵ عملگر تقسیم (Division)

عملگر تقسیم (Division) یکی از عملگرهای پیچیده‌تر در جبر رابطه‌ای است که برای یافتن تمامی مقدارهای خاصی در یک ستون از یک جدول استفاده می‌شود که برای تمام مقدارهای یک ستون دیگر در جدول دوم، یک جفت مقدار متناظر وجود داشته باشد. به عبارت ساده‌تر، عملگر تقسیم به ما کمک می‌کند تا مشخص کنیم کدام مقادیر در یک جدول، به صورت کامل در جدول دیگر پوشش داده شده‌اند.

به عنوان مثال، فرض کنید جدولی از سفارش‌ها و جدولی از محصولات داریم. با استفاده از عملگر تقسیم می‌توانیم مشخص کنیم کدام محصولات در هیچ سفارشی وجود ندارند. یا به عبارتی دیگر، کدام محصولات به فروش نرفته‌اند. این عملگر در مواردی که نیاز به تحلیل‌های پیچیده‌تر و یافتن روابط خاص بین جداول داریم، بسیار مفید است. عملگر تقسیم در بسیاری از پایگاه‌های داده‌های رابطه‌ای به صورت مستقیم پشتیبانی نمی‌شود و معمولاً با استفاده از سایر عملگرها مانند `inner join` و `not exists` پیاده‌سازی می‌شود.

۲/۵ کاربرد و استفاده از `tinyint` در الگوریتم‌های یادگیری ماشین

۲/۵/۱ تعریف

Tinyint یک نوع داده عددی صحیح است که در پایگاه‌های داده برای ذخیره اعداد صحیح کوچک استفاده می‌شود. دامنهٔ مقادیر قابل ذخیره در این نوع داده معمولاً بین 0 تا 255 است. به دلیل محدود بودن دامنهٔ مقادیر، Tinyint حافظه کمتری نسبت به انواع داده‌ی عددی صحیح بزرگ‌تر مانند `int` یا `bigint` اشغال می‌کند.

۲/۵/۲ در یادگیری ماشین و کانولوشن‌های دو بعدی Tinyint

در حوزه یادگیری ماشین، Tinyint به ویژه در شبکه‌های عصبی کانولوشنی (CNN) کاربرد دارد. CNN‌ها برای پردازش داده‌های تصویری بسیار موثر هستند. هر پیکسل در یک تصویر می‌تواند به عنوان یک عدد صحیح بین 0 تا 255 (در تصاویر خاکستری) یا یک بردار سه‌تایی از اعداد صحیح بین 0 تا 255 (در تصاویر رنگی) نمایش داده شود.

در یک شبکهٔ کانولوشنی، تصاویر ورودی معمولاً به عنوان یک تنسور (tensor) سه بعدی نمایش داده می‌شوند که ابعاد آن عبارتند از: ارتفاع تصویر، عرض تصویر و تعداد کانال‌ها. هر کانال معمولاً یک ویژگی خاص از تصویر را نشان می‌دهد (مانند شدت روشنایی در تصاویر خاکستری یا کانال‌های رنگی قرمز، سبز و آبی در تصاویر رنگی). از آنجا که مقادیر پیکسل‌ها در هر کانال بین 0 تا 255 است، نوع داده‌ی Tinyint برای ذخیره این مقادیر مناسب است.

در عملیات کانولوشن، از فیلترهایی به نام ماسک استفاده می‌شود که روی تصویر حرکت می‌کنند و با پیکسل‌های زیر خود ضرب داخلی انجام می‌دهند. این ماسک‌ها نیز معمولاً به صورت ماتریس‌هایی از اعداد صحیح نمایش داده می‌شوند که مقادیر آن‌ها می‌تواند بین 0 تا 255 یا حتی مقادیر منفی باشد. در برخی موارد، از نوع داده‌ی Tinyint برای ذخیره مقادیر این ماسک‌ها استفاده می‌شود.

در لایه‌های مختلف یک شبکهٔ کانولوشنی، از توابع فعال‌سازی برای غیرخطی‌سازی خروجی استفاده می‌شود. برخی از این توابع فعال‌سازی (مانند ReLU) خروجی‌هایی بین 0 تا بین‌نهایت تولید می‌کنند. با این حال، در برخی موارد، ممکن است خروجی این توابع به یک دامنه محدود مانند 0 تا 255 محدود شود تا بتوان از نوع داده‌ی Tinyint برای ذخیره آن‌ها استفاده کرد.

به طور خلاصه، Tinyint یک نوع داده‌ی عددی صحیح کوچک و کارآمد است که در بسیاری از کاربردهای پایگاه داده و یادگیری ماشین مورد استفاده قرار می‌گیرد. در شبکه‌های عصبی کانولوشنی، Tinyint برای نمایش مقادیر پیکسل‌ها، ماسک‌های کانولوشن و برخی از فعالیت‌سازی‌ها به کار می‌رود.

۲/۶ تمرین ۴

دستوراتی بنویسید که اسمی استادانی که یکی از درس‌هایی را تدریس می‌کنند که احمدی در آن درس‌ها ثبت نام کرده را مشخص نماید.

در حل این مسئله باید نام دروسی که احمدی در آن‌ها ثبت‌نام کرده است را بدست بیاوریم سپس استادانی که آن درس‌ها را ارائه می‌دهد جدا کنیم، همچنین یک استاد می‌تواند چند درس مختلف را ارائه دهد. فرض اگر احمدی دو درس ریاضی ۱ و فیزیک ۱ را برداشته باشد باید تمام استادی که حداقل یکی از این دو درس را ارائه می‌دهند را نمایش دهیم:

$$\begin{aligned} T_1 &\leftarrow \prod_{s\#} (\sigma_{sname \text{ like } \%ahmadi\%}, (Student)) \\ T_2 &\leftarrow \prod_{c\#} (T_1 \bowtie Section) \\ T_3 &\leftarrow \prod_{cname} (T_2 \bowtie Course) \\ T_4 &\leftarrow \prod_{c\#} (T_3 \bowtie Course) \\ T_5 &\leftarrow \prod_{pname} (T_4 \bowtie Section) \end{aligned}$$

با توجه به رابطه فوق یک تابع برای آن می‌نویسیم:

```

1 CREATE FUNCTION GetTeachers()
2 RETURNS TABLE
3 AS
4 RETURN
5 (
6   SELECT DISTINCT pname FROM Section
7   WHERE c# IN (
8     SELECT c# FROM Course
9     WHERE cname IN (
10       SELECT cname FROM Course
11       WHERE c# IN (
12         SELECT c# FROM Section
13         WHERE s# IN (
14           SELECT s# FROM Student
15           WHERE sname LIKE '%ahmadi%'
16         )
17       )
18     )
19   )
20 );
21
22 SELECT * FROM GetTeachers();

```

شکل ۲ - تابع تمرین ۴

۲/۷ تمرین ۵

دستوراتی بنویسید که مشخصات دانشجویانی که معدل آنها از متوسط معدل دانشجویان دانشکده شان کمتر است را تعیین نماید.

برای حل این مسئله باید معدل دانشجویان و خود دانشجو هم دانشکده‌ای را میانگین گرفته و فیلتر کنیم سپس به وسیله پروسیجر نمایش دهیم:



```

1 CREATE PROCEDURE GetStudents()
2 AS
3 BEGIN
4     SELECT s#, sname, city, avg, clg#
5     FROM student AS s1
6     WHERE avg < (
7         SELECT AVG(avg)
8         FROM student AS s2
9         WHERE s1.clg# = s2.clg#
10    );
11 END;
12
13 EXEC GetStudents;

```

شکل ۳ - پروسیجر تمرین ۵

۲/۸ تمرین ۶

اگر برای رابطه $R(A, B, C, D, E, F)$ وابستگی‌های تابعی زیر وجود داشته باشد کلید کاندید را تعیین نمایید.

$$F = \{A \rightarrow D, AE \rightarrow D, AD \rightarrow C, AF \rightarrow B, C \rightarrow E\}$$

$$F^+ = \{A \rightarrow D, AE \rightarrow D, AD \rightarrow C, AF \rightarrow B, C \rightarrow E, AD \rightarrow E\}$$

$$F_{opt} = \{A \rightarrow D, AD \rightarrow C, AF \rightarrow B, C \rightarrow E, AD \rightarrow E\}$$

$$A^+ = \{A, D\}$$

$$C^+ = \{C, E\}$$

$$AF^+ = \{A, F, B\}$$

$$AD^+ = \{A, D, E, C\} \rightarrow B, F$$

در نتیجه کلید کاندید با ADBF برابر است.

۷ تمرین ۲۹

اگر (A, B, C, D, E, F) وابستگیهای تابعی زیر را داشته باشد کلید کاندید را مشخص نمایید.

Subject: _____ Year. _____ Month. _____ Date. () _____									
$R = \{A, B, C, D, E, F\}$ حکم $F^* = \{A \rightarrow BD, E \rightarrow C, BC \rightarrow DE, E \rightarrow A\}$									
$F^t = \{A \rightarrow BD, A \rightarrow B, A \rightarrow D, E \rightarrow C, BC \rightarrow DE, BC \rightarrow D, BC \rightarrow E, E \rightarrow A, E \rightarrow BD, E \rightarrow B, E \rightarrow D, BC \rightarrow A, BC \rightarrow D\}$									
$F_{opt} = \{A \rightarrow B, A \rightarrow D, E \rightarrow C, BC \rightarrow D, BC \rightarrow E, E \rightarrow A, E \rightarrow B, E \rightarrow D, BC \rightarrow A, BC \rightarrow D\}$									
$A' = \{A, B, D\}$ $BC' = \{B, C, D, A, E\}$ $E^t = \{E, A, B, C, D\} \rightarrow FF$ $EF = \text{نیکی}$									

شکل ۴ - حل تمرین ۷

۲/۱۰ تمرین ۸

رابطه $R(A,B,C,D,E)$ با وابستگیهای تابعی زیر را در نظر بگیرید و کلید کاندید را برای R تعیین نمایید.

تمرین ۸

$$R = \{A, B, C, D, E\}$$

$$F = \{A \rightarrow B, D \rightarrow AE, B \rightarrow C\}$$

$$F^+ = \{A \rightarrow B, D \rightarrow AE, D \rightarrow A, D \rightarrow E, B \rightarrow C, A \rightarrow C, D \rightarrow C, D \rightarrow E, D \rightarrow B\}$$

$$F_{opt} = \{A \rightarrow B, D \rightarrow A, D \rightarrow E, B \rightarrow C, A \rightarrow C, D \rightarrow C, D \rightarrow E, D \rightarrow B\}$$

$$A^t = \{B, A, C\}$$

$$D^t = \{D, A, B, C, E\} \rightarrow D \text{ کلید کاندید}$$

$$R^t = \{B, C\}$$

شکل ۸- حل تمرین ۸

ORM ۲/۱۱**۲/۱۱/۱ نگاشت شیء-رابطه‌ای**

Object-Relational Mapping (نگاشت شیء-رابطه‌ای)، تکنیکی در برنامه‌نویسی است که به توسعه دهنده‌گان اجازه می‌دهد با پایگاه‌های داده رابطه‌ای (مانند MySQL، PostgreSQL و غیره) با استفاده از زبان‌های برنامه‌نویسی شیء‌گرا (مانند Java، Python، سی‌شارپ و غیره) تعامل داشته باشند، بدون اینکه نیاز به نوشتن مستقیم کدهای SQL باشد. به عبارت دیگر، ORM یک لایه میانی بین برنامه شیء‌گرای شما و پایگاه داده رابطه‌ای ایجاد می‌کند و اشیاء موجود در کد شما را به جداول و رکوردهای پایگاه داده نگاشت می‌کند. این کار باعث می‌شود که شما بتوانید با داده‌ها به صورت شیء‌گرا کار کنید و از پیچیدگی‌های نوشتن کوئری‌های SQL به طور مستقیم رها شوید.

ORM با تبدیل عملیات‌های شیء‌گرا (مانند ایجاد یک شیء جدید، تغییر خصوصیات یک شیء، یا حذف یک شیء) به کوئری‌های SQL متناظر، فرآیند تعامل با پایگاه داده را ساده می‌کند. به عنوان مثال، به جای نوشتن یک کوئری INSERT برای اضافه کردن یک رکورد جدید به جدول، شما یک شیء از کلاس مربوط به آن جدول ایجاد می‌کنید و خصوصیات آن را مقداردهی می‌کنید. سپس، ORM به طور خودکار کوئری INSERT مناسب را تولید و اجرا می‌کند. همچنین، ORM امکان بازیابی داده‌ها از پایگاه داده و تبدیل آن‌ها به اشیاء جawa را فراهم می‌کند. این کار باعث می‌شود که کد شما خواناتر، قابل نگهداری‌تر و توسعه‌پذیرتر شود.

مزایای استفاده از ORM شامل کاهش میزان کد SQL نوشته شده، افزایش سرعت توسعه، بهبود امنیت (با جلوگیری از حملات تزریق SQL در صورت استفاده صحیح)، و افزایش قابلیت حمل کد بین پایگاه‌های داده مختلف است (زیرا ORM جزئیات مربوط به هر پایگاه داده را انتزاع می‌کند). معایب آن نیز می‌تواند شامل سربار عملکردی (به دلیل تبدیل عملیات‌های شیء‌گرا به SQL)، پیچیدگی در برخی موارد خاص (مانند کوئری‌های بسیار پیچیده)، و نیاز به یادگیری نحوه کار با ORM باشد. در کد ارائه شده در سوال قبلی، استفاده از EntityManager با Hibernate به عنوان ORM نشان می‌دهد که چگونه می‌توان اشیاء Truck و Driver را با استفاده از EntityManager در پایگاه داده ذخیره کرد.

۲/۱۱/۲ پیاده‌سازی

در بخش ۳.۶ بطور کامل پیاده‌سازی در جawa شرح داده شده است.

۳ پروژه

۳/۱ معرفی پایگاه داده

این پایگاه داده با نام TransportDK برای مدیریت اطلاعات مربوط به حمل و نقل در دانمارک طراحی شده است. هدف اصلی آن ذخیره و سازماندهی داده‌های مربوط به رانندگان، کامیون‌ها، تریلرها، بارها و شرکت‌های حمل و نقل است. این ساختار به منظور تسهیل عملیات‌های مختلف مدیریت ناوگان، برنامه‌ریزی حمل و نقل، پیگیری بارها، و مدیریت پرسنل طراحی شده است.

این پایگاه داده از چندین جدول تشکیل شده است که هر کدام اطلاعات خاصی را نگهداری می‌کنند. جدول driver اطلاعات رانندگان شامل مشخصات فردی، گواهینامه، میزان مهارت، حقوق و درآمد را ذخیره می‌کند. جدول truck اطلاعات کامیون‌ها مانند پلاک، برد، مدل، مشخصات فنی و میزان مصرف سوخت را در خود جای می‌دهد. جدول trailer نیز به ذخیره اطلاعات تریلرها از جمله مشخصات فنی، وزن و ابعاد آن‌ها می‌پردازد. جدول cargo اطلاعات مربوط به بارها از جمله وزن، نوع، مبدأ، مقصد و شرکت‌های مبدأ و مقصد را نگهداری می‌کند. جدول company اطلاعات شرکت‌های حمل و نقل شامل نام، آدرس، مدیر و تعداد کارکنان را ذخیره می‌کند.

علاوه بر جداول اصلی، جداول رابطی نیز برای نمایش ارتباط بین جداول ایجاد شده‌اند. جدول attachtrailer ارتباط بین کامیون‌ها و تریلرها را نشان می‌دهد، به این معنی که هر کامیون می‌تواند به یک یا چند تریلر متصل باشد. جدول driving ارتباط بین رانندگان و کامیون‌ها را مشخص می‌کند و نشان می‌دهد که هر راننده با کدام کامیون رانندگی می‌کند. جدول loading ارتباط بین تریلرها و بارها را نشان می‌دهد و مشخص می‌کند که هر تریلر چه بارهایی را حمل می‌کند. در نهایت، جدول hiring ارتباط بین رانندگان و شرکت‌ها را نشان می‌دهد و مشخص می‌کند که هر راننده در کدام شرکت استخدام شده است. این ساختار رابطه‌ای به پایگاه داده امکان می‌دهد تا اطلاعات را به صورت سازمان‌یافته و با کمترین میزان افزونگی ذخیره کند و امکان انجام پرسش‌های پیچیده و گزارش‌گیری را فراهم آورد.

در کل، پایگاه داده TransportDK با استفاده از جداول مختلف و روابط بین آن‌ها، یک سیستم جامع برای مدیریت اطلاعات حمل و نقل فراهم می‌کند. استفاده از محدودیت‌ها (CHECK) و کلیدهای اصلی و خارجی (PRIMARY KEY و FOREIGN KEY) در طراحی جداول، یکپارچگی داده‌ها و جلوگیری از ورود داده‌های نامعتبر را تضمین می‌کند. این پایگاه داده می‌تواند به شرکت‌های حمل و نقل در مدیریت بهتر منابع، بهینه‌سازی مسیرها، و بهبود کارایی عملیات کمک کند. نمودار مرتب با این محیط عملیاتی در شکل بخش ۲.۳ نشان داده شده است.

۳/۲ معرفی جداول اصلی

۳/۲/۱ جدول driver (راننده):

این جدول اطلاعات مربوط به رانندگان را در سیستم حمل و نقل ذخیره می‌کند. هدف از ایجاد این جدول، نگهداری جزئیات پرسنلی و حرفة‌ای رانندگان از جمله مشخصات فردی، اطلاعات تماس، نوع گواهینامه، میزان مهارت، سابقه رانندگی، حقوق و دستمزد و سایر اطلاعات مرتبط است. وجود این جدول امکان مدیریت کارآمد رانندگان، اختصاص وظایف به آن‌ها، محاسبه حقوق و دستمزد بر اساس عملکرد، و بررسی سوابق آن‌ها را فراهم می‌کند. فیلدی‌ای مانند rating و skill امکان ارزیابی و دستribution رانندگان بر اساس توانایی‌هایشان را فراهم می‌سازند. همچنین، ذخیره عکس پرسنلی در فیلد personalpic به مدیریت بهتر اطلاعات پرسنلی کمک می‌کند.

این جدول شامل فیلدی‌ای مانند id (شناسه یکتا)، eid (شناسه استخدامی خودکار)، fname (نام)، lname (نام خانوادگی)، gender (جنسيت)، number (آدرس)، addr (جنسیت)، age (سن)، lisence (شماره تماس)، rating (نوع گواهینامه)، skill (امتیاز)، personalpic (عکس شخصی) و income (درآمد به ازای هر کیلومتر)، drivetime (زمان رانندگی)، personalpic (مهارت) و income (حقوق). محدودیت‌های تعریف شده برای فیلدی‌ای مانند CHECKها، صحت و یکپارچگی داده‌ها را تضمین می‌کنند. به عنوان مثال،

محدودیت‌های مربوط به سن (age)، نوع گواهینامه (lisence)، امتیاز (rating) و مهارت (skill) از ورود داده‌های نامعتبر جلوگیری می‌کنند.

۳/۲/۲ جدول truck (کامیون):

این جدول اطلاعات مربوط به کامیون‌های موجود در ناوگان حمل و نقل را ذخیره می‌کند. هدف از ایجاد این جدول، نگهداری مشخصات فنی کامیون‌ها، اطلاعات مربوط به مدل و برند، وضعیت فنی، میزان کارکرد و سایر اطلاعات مرتبط است. وجود این جدول امکان مدیریت ناوگان، برنامه‌ریزی تعمیر و نگهداری، محاسبه هزینه‌های سوت و استهلاک، و اختصاص کامیون‌ها به رانندگان و بارها را فراهم می‌کند. فیلدهایی مانند kilometer و fuelconsumption امکان محاسبه دقیق هزینه‌ها و بررسی وضعیت فنی کامیون‌ها را فراهم می‌سازند.

این جدول شامل فیلدهایی مانند plate (پلاک - کلید اصلی)، tid (شناسه کامیون خودکار)، brand (برند)، model (مدل)، trailer (نوع تریلر قابل اتصال)، trailerid (شناسه تریلر)، color (رنگ)، axil (تعداد محور)، powerhp (قدرت موتور)، gear (نوع گیربکس)، kilometer (میزان کارکرد)، manyear (سال ساخت)، fuelconsumption (صرف سوت) و garage (محل استقرار) می‌باشد. محدودیت‌های تعریف شده برای فیلدها مانند CHECKها، از ورود داده‌های نامعتبر جلوگیری می‌کنند. به عنوان مثال، محدودیت‌های مربوط به تعداد محور (axil)، قدرت موتور (powerhp)، میزان کارکرد (kilometer) و سال ساخت (manyear) از صحت داده‌ها اطمینان حاصل می‌کنند.

۳/۲/۳ جدول trailer (تریلر):

این جدول اطلاعات مربوط به تریلرهای مورد استفاده در حمل و نقل را ذخیره می‌کند. هدف از ایجاد این جدول، نگهداری مشخصات فنی تریلرهای کاربری، وزن و ابعاد، و سایر اطلاعات مرتبط است. وجود این جدول امکان مدیریت تریلرهای اختصاص آن‌ها به کامیون‌ها و بارها، و محاسبه ظرفیت حمل و نقل را فراهم می‌کند. فیلدهایی مانند widthm، lengthm، maxweight، weightton و hieghtm امکان محاسبه دقیق ظرفیت و محدودیت‌های حمل بار را فراهم می‌سازند.

این جدول شامل فیلدهایی مانند trailerid (شناسه تریلر - کلید اصلی خودکار)، plate (پلاک کامیون متصل)، garage (محل استقرار)، manufacture (سازنده)، ttype (نوع تریلر)، tbody (نوع بدنه)، weightton (وزن)، maxweight (حداکثر وزن قابل حمل)، manyear (سال ساخت)، axel (تعداد محور)، widthm (عرض)، lengthm (طول)، hieghtm (ارتفاع) و lifteable (قابلیت بالابری) می‌باشد. محدودیت‌های تعریف شده برای فیلدها مانند CHECKها، از ورود داده‌های نامعتبر جلوگیری می‌کنند. به عنوان مثال، محدودیت‌های مربوط به وزن (weightton) و ابعاد (widthm، lengthm و hieghtm) از صحت داده‌ها مربوط به ظرفیت تریلر اطمینان حاصل می‌کنند.

۳/۲/۴ جدول cargo (بار):

این جدول اطلاعات مربوط به بارهای حمل شده را ذخیره می‌کند. هدف از ایجاد این جدول، نگهداری جزئیات بار از جمله وزن، نوع، مبدأ، مقصد، شرکت‌های مرتبط، زمان تحویل و سایر اطلاعات مرتبط است. وجود این جدول امکان پیگیری بارهای برنامه‌ریزی حمل و نقل، محاسبه هزینه‌ها و مدیریت لجستیک را فراهم می‌کند. فیلدهایی مانند origin، destination و deadline امکان برنامه‌ریزی دقیق حمل و نقل و پیگیری وضعیت بار را فراهم می‌سازند.

این جدول شامل فیلدهایی مانند loadid (شناسه بار - کلید اصلی خودکار)، name1 (نام بار - یکتا)، weightton (وزن)، ocompany (مبدأ)، ltype (نوع بار)، ttype (نوع تریلر مورد نیاز)، tbody (نوع بدنه تریلر مورد نیاز)، origin (مبدأ)، destination (مقصد)، dcompany (شرکت مبدأ)، distance (مسافت)، deadline (تاریخ تحویل)، income (درآمد) و scort (درآمد) می‌باشد. محدودیت‌های تعریف شده برای فیلدها مانند CHECKها، از ورود داده‌های نامعتبر جلوگیری می‌کنند. به عنوان اسکورت) می‌باشد. محدودیت‌های تعریف شده برای فیلدها مانند CHECKها، از ورود داده‌های نامعتبر جلوگیری می‌کنند. به عنوان مثال، محدودیت‌های مربوط به وزن (weightton) و درآمد (income) از صحت داده‌های مالی اطمینان حاصل می‌کنند.

۳/۲/۵ جدول company (شرکت):

این جدول اطلاعات مربوط به شرکت‌های حمل و نقل را ذخیره می‌کند. هدف از ایجاد این جدول، نگهداری اطلاعات تماس، آدرس، مدیر و سایر اطلاعات مرتبط با شرکت‌ها است. وجود این جدول امکان مدیریت ارتباط با شرکت‌ها، بررسی سوابق آن‌ها و اختصاص رانندگان به آن‌ها را فراهم می‌کند. فیلدهایی مانند `addr` و `servicee` اطلاعات جامعی در مورد شرکت‌ها ارائه می‌دهند.

این جدول شامل فیلدهایی مانند `name1` (نام شرکت - یکتا)، `cid` (شناسه شرکت - کلید اصلی)، `number` (شماره تماس)، `city` (شهر)، `addr` (آدرس - یکتا)، `area` (محدهود فعالیت)، `servicee` (خدمات ارائه شده)، `manager` (مدیر - یکتا) و `employe` (مدیر - یکتا) می‌باشد. محدودیت‌های تعریف شده برای فیلدها مانند `CHECK`، از ورود داده‌های نامعتبر جلوگیری می‌کند. به عنوان مثال، محدودیت‌های مربوط به شناسه شرکت (`cid`) و محدوده فعالیت (`area`) از صحت داده‌ها اطمینان حاصل می‌کنند.

۳/۳ معرفی جداول رابطه‌ای

۳/۳/۱ جدول attachtrailer (اتصال تریلر):

این جدول برای نمایش ارتباط بین کامیون‌ها و تریلرها استفاده می‌شود. از آنجایی که یک کامیون می‌تواند به چند تریلر متصل شود و یک تریلر نیز می‌تواند توسط چند کامیون مختلف کشیده شود (هرچند به طور همزمان فقط توسط یک کامیون کشیده می‌شود)، یک رابطه چند به چند بین این دو جدول وجود دارد. جدول `attachtrailer` این رابطه را مدیریت می‌کند و مشخص می‌کند که کدام تریلرها به کدام کامیون‌ها مرتبط هستند. این جدول از افزونگی داده‌ها جلوگیری می‌کند و امکان مدیریت انعطاف‌پذیر ارتباط بین کامیون‌ها و تریلرها را فراهم می‌سازد.

این جدول شامل سه فیلد است: `tid` (شناسه کامیون)، `plate` (پلاک کامیون) و `trailerid` (شناسه تریلر). فیلدهای `plate` و `trailerid` به ترتیب کلیدهای خارجی هستند که به جداول `truck` و `trailer` ارجاع می‌دهند. وجود این کلیدهای خارجی، یکپارچگی داده‌ها را تضمین می‌کند و از ایجاد رکوردهایی با اطلاعات نامعتبر جلوگیری می‌کند. حذف یک رکورد از جدول `truck` یا `trailer` باعث حذف رکوردهای مرتبط در جدول `attachtrailer` می‌شود (ON DELETE CASCADE). این ویژگی از ناسازگاری داده‌ها جلوگیری می‌کند.

۳/۳/۲ جدول driving (رانندگی):

این جدول برای نمایش ارتباط بین رانندگان و کامیون‌ها استفاده می‌شود. هر راننده می‌تواند با چند کامیون مختلف رانندگی کند و هر کامیون نیز می‌تواند توسط چند راننده مختلف رانده شود (البته نه به طور همزمان). این رابطه چند به چند توسط جدول `driving` مدیریت می‌شود. این جدول مشخص می‌کند که کدام راننده با کدام کامیون رانندگی کرده یا می‌کند. این اطلاعات برای مدیریت سوابق رانندگی، تخصیص کامیون به راننده و محاسبه عملکرد رانندگان مفید است.

این جدول شامل سه فیلد است: `eid` (شناسه استخدامی راننده)، `plate` (پلاک کامیون) و `tid` (شناسه کامیون). فیلدهای `eid` و `plate` به ترتیب کلیدهای خارجی هستند که به جداول `driver` و `truck` ارجاع می‌دهند. وجود این کلیدهای خارجی، یکپارچگی داده‌ها را تضمین می‌کند. حذف یک رکورد از جدول `driver` یا `truck` باعث حذف رکوردهای مرتبط در جدول `driving` می‌شود (ON DELETE CASCADE).

۳/۳/۳ جدول loading (بارگیری):

این جدول برای نمایش ارتباط بین تریلرها و بارها استفاده می‌شود. یک تریلر می‌تواند چندین بار مختلف را حمل کند و یک بار نیز می‌تواند در تریلرهای مختلف حمل شود. این رابطه چند به چند توسط جدول `loading` مدیریت می‌شود. این جدول مشخص می‌کند که کدام بارها در کدام تریلرها بارگیری شده‌اند. این اطلاعات برای پیگیری بارها، مدیریت ظرفیت تریلرها و برنامه‌ریزی حمل و نقل ضروری است.

این جدول شامل دو فیلد است: **loadid** (شناسه تریلر) و **trailerid** (شناسه بار). این دو فیلد به ترتیب کلیدهای خارجی هستند که به جداول **cargo** و **trailer** ارجاع می‌دهند. وجود این کلیدهای خارجی، یکپارچگی داده‌ها را تضمین می‌کند. حذف یک رکورد از جدول **cargo** یا **trailer** باعث حذف رکوردهای مرتبط در جدول **loading** می‌شود (ON DELETE CASCADE).

۳/۳/۴ جدول hiring (استخدام):

این جدول برای نمایش ارتباط بین رانندگان و شرکت‌های حمل و نقل استفاده می‌شود. هر راننده می‌تواند در یک شرکت استخدام شود و هر شرکت نیز می‌تواند چندین راننده را استخدام کند. این رابطه یک به چند از طرف شرکت به راننده است که توسط جدول **hiring** مدیریت می‌شود. این جدول مشخص می‌کند که کدام راننده در کدام شرکت مشغول به کار است. این اطلاعات برای مدیریت پرسنل، محاسبه حقوق و دستمزد و تخصیص وظایف مفید است.

این جدول شامل دو فیلد است: **eid** (شناسه استخدامی راننده) و **cid** (شناسه شرکت). این دو فیلد به ترتیب کلیدهای خارجی هستند که به جداول **driver** و **company** ارجاع می‌دهند. وجود این کلیدهای خارجی، یکپارچگی داده‌ها را تضمین می‌کند. حذف یک رکورد از جدول **driver** یا **company** باعث حذف رکوردهای مرتبط در جدول **hiring** می‌شود (ON DELETE CASCADE).

۳/۴ مزایا و معایب پایگاه داده

۳/۴/۱ نکات مثبت و مزایا:

- مدل‌سازی جامع: این پایگاه داده با داشتن جداول مختلف برای رانندگان، کامیون‌ها، تریلرهای بارها و شرکت‌ها، یک مدل جامع از سیستم حمل و نقل ارائه می‌دهد. این ساختار امکان ذخیره و مدیریت اطلاعات مرتبط با جنبه‌های مختلف این صنعت را فراهم می‌کند.
- روابط تعریف شده: استفاده از جداول رابطه‌ای (**attachtrailer**, **loading**, **driving**, **hire**), روابط چند به چند بین جداول اصلی را به درستی مدل می‌کند. این امر از افزونگی داده‌ها جلوگیری کرده و یکپارچگی داده‌ها را بهبود می‌بخشد.
- محدودیت‌های تعریف شده (Constraints): استفاده از محدودیت‌هایی مانند **NOT NULL** و **FOREIGN KEY** در تعریف جداول، از ورود داده‌های نامعتبر جلوگیری می‌کند و صحت و یکپارچگی داده‌ها را تضمین می‌کند. به عنوان مثال، محدودیت‌های مربوط به سن راننده، نوع گواهینامه، وزن بار و غیره، از ورود داده‌های پرت و نامعقول جلوگیری می‌کند.
- استفاده از کلیدهای اصلی و خارجی: استفاده از کلیدهای اصلی (PRIMARY KEY) برای شناسایی یکتای هر رکورد و کلیدهای خارجی (FOREIGN KEY) برای ایجاد ارتباط بین جداول، ساختار پایگاه داده را بهبود می‌بخشد و امکان انجام پرسش‌های پیچیده و بازیابی اطلاعات مرتبط را فراهم می‌کند.
- استفاده از **IDENTITY** برای کلیدهای خودکار: استفاده از **IDENTITY** برای فیلدهایی مانند **eid** در جدول **driver**, **truck**, **trailerid** در جدول **trailer** و **loadid** در جدول **loading**, **cargo**, فرآیند ایجاد رکوردهای جدید را ساده‌تر می‌کند و از بروز خطاهای ناشی از تکرار کلید جلوگیری می‌کند.
- قابلیت توسعه: ساختار پایگاه داده به گونه‌ای طراحی شده است که در صورت نیاز به افزودن اطلاعات جدید، می‌توان جداول و فیلدهای جدیدی را به آن اضافه کرد.

۳/۴/۲ نکات منفی و معایب:

- پیچیدگی نسبی: با وجود مزایای روابط تعریف شده، ساختار پایگاه داده نسبتاً پیچیده است و درک و مدیریت آن برای افراد غیر متخصص ممکن است دشوار باشد.

- عدم وجود جزئیات بیشتر در برخی جداول: برخی از جداول می‌توانند جزئیات بیشتری را در خود جای دهند. به عنوان مثال، جدول cargo می‌تواند اطلاعات بیشتری در مورد نوع بسته‌بندی، شرایط نگهداری و بیمه بار را شامل شود.
- جدول truck می‌تواند اطلاعات مربوط به معاینه فنی و بیمه خودرو را نیز در خود ذخیره کند.
- عدم در نظر گرفتن تاریخچه تغییرات: پایگاه داده فعلی تاریخچه تغییرات داده‌ها را نگهداری نمی‌کند. به عنوان مثال، اگر اطلاعات یک راننده تغییر کند، اطلاعات قبلی از دست می‌رود. برای رفع این مشکل، می‌توان از تکنیک‌هایی مانند ایجاد جدول‌های audit یا استفاده از ویژگی‌های خاص DBMS برای نگهداری تاریخچه تغییرات استفاده کرد.
- عدم وجود جداول برای مدیریت مسیرها و زمان‌بندی: پایگاه داده فعلی جدولی برای مدیریت مسیرهای حمل و نقل و زمان‌بندی آن‌ها ندارد. این اطلاعات می‌تواند برای بهینه‌سازی مسیرها و مدیریت زمان تحویل بارها بسیار مفید باشد.
- احتمال بروز بن‌بست (Deadlock) در تراکنش‌های پیچیده: با توجه به روابط چند به چند بین جداول وجود عملیات حذف آبشراری (ON DELETE CASCADE)، در تراکنش‌های پیچیده احتمال بروز بن‌بست وجود دارد. برای جلوگیری از این مشکل، باید از تکنیک‌های مدیریت تراکنش و قفل‌گذاری مناسب در سطح پایگاه داده استفاده کرد.

۳/۵ طرح مسائل

۱ ۳/۵/۱ مسئله

اطلاعات کامل راننده‌گانی که برنده کامیون آن‌ها اسکانیا می‌باشد.



```

1 -- Tamrin 1
2 SELECT *
3 FROM driver, truck, driving
4 WHERE
5   driver.eid = driving.eid AND
6   driving.tid = truck.plate AND
7   truck.brand = 'SCANIA'

```

شکل ۶-مسئله ۱

۲ ۳/۵/۲ مسئله

نام کامل و شماره تلفن راننده‌گانی که در شرکتی استخدام شده‌اند که در شهر البورگ است.



```

1 -- Tamrin 2
2 SELECT driver.fname + ' ' + driver.lname as fullname, driver.number
3 FROM driver, hiring, company
4 WHERE
5   driver.eid = hiring.eid AND
6   company.cid = hiring.cid AND
7   company.city = 'Aalborg'

```

شکل ۷-مسئله ۲

۳ ۳/۵/۳ مسئله

شماره کامیون، اسم کامل کامیون، قدرت موتور و تعداد دنده کامیون‌هایی که بار سنگینی دارند و وزن آن‌ها بیشتر از ۲۰ تن است همچنین قدرت موتور بالای ۴۸۰ اسب بخار و تعداد دنده بالای ۱۲ است.

```

1 -- Tamrin 3
2 SELECT tid, brand + ' ' + model AS truck_name, powerhp, gear
3 FROM truck, attachtrailer, trailer, loading, cargo
4 WHERE
5   truck.tid = attachtrailer.tid AND
6   trailer.trailerid = attachtrailer.trailerid AND
7   trailer.trailerid = loading.trailerid AND
8   cargo.loadid = loading.loadid AND
9   cargo.weightton > 20 AND
10  truck.gear > 12
11  truck.powerhp > 480

```

شکل ۸ - مسئله ۴

۳/۵/۴ مسئله ۴

اطلاعات کامل تریلرهایی که مقدار وزن بارگیری آن بیشتر ۱۵ تن است و گاراز آن‌ها در شهرهایی است که کمپانی حمل و نقل در آن وجود دارد را به ترتیب اسم سازنده تریلر گروهبندی می‌کند.

```

1 -- Tamrin 4
2 SELECT *
3 FROM trailer
4 WHERE maxweight > 15
5 GROUP BY trailer.manufacture
6 HAVING garage CONTAINS (
7   SELECT city
8   FROM company
9 )

```

شکل ۹ - مسئله ۴

۳/۵/۵ مسئله ۵

شماره پلاک و شماره تریلرهایی که از نوع کفی لودر (Low lead, Low bed) و تک باره هستند.

```

1 -- Tamrin 5
2 SELECT plate, trailerid
3 FROM trailer
4 WHERE
5   ttype like 'LOW%' AND
6   tbody = 'SINGLE'

```

شکل ۱۰ - مسئله ۵

۳/۵/۶ مسئله ۶

نام کامیون، شماره پلاک و تریلر کامیون‌هایی با برند رنو هستند.

```
1 -- Tamrin 6
2 SELECT brand + ' ' + model AS truck_name, plate, trailer
3 FROM truck
4 WHERE brand = 'Renault'
```

شکل ۱۱ - مسئله ۶

۷ مسئله ۳/۵/۷

مشخصات بارهایی که از نوع کانتینری و از مبدأ شهر اودنیس به هستند.

```
1 -- Tamrin 7
2 SELECT *
3 FROM cargo
4 WHERE
5   ttype = 'Container' AND
6   origin = 'Odense'
```

شکل ۱۲ - مسئله ۷

۸ مسئله ۳/۵/۸

نام کامل، شماره تلفن، ادرس و درآمد سالانه رانندگانی که آدرس آنها شامل شهر کوبنه‌اگن است.

```
1 -- Tamrin 8
2 SELECT fname + ' ' + driver.lname AS fullname, number, addr, salery
3 FROM driver
4 WHERE addr like '%Copenhagen%'
```

شکل ۱۳ - مسئله ۸

۹ مسئله ۳/۵/۹

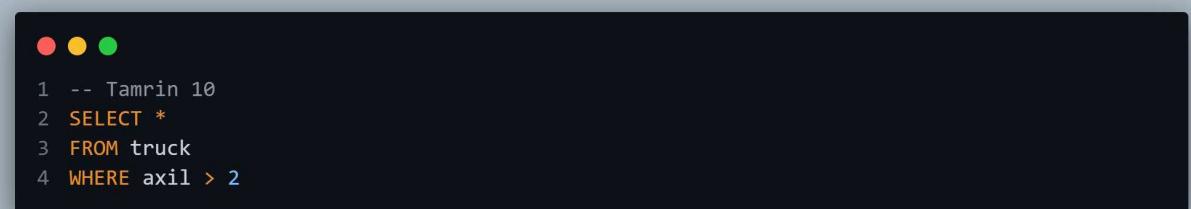
شماره ملی و درآمد سالانه رانندگانی برای دریافت مالیات که در شرکتی کار می‌کنند که بیش از ۳۰۰ کارمند دارند و درآمد آن در هر کیلومتر بیشتر از ۳۰ دلار می‌باشد.

```
1 -- Tamrin 9
2 SELECT driver.id, driver.salery
3 FROM driver, company, hiring
4 WHERE
5   company.cid = hiring.cid AND
6   driver.eid = hiring.eid AND
7   company.employe > 300 AND
8   incomepkm > 30
```

شکل ۱۴ - مسئله ۹

۱۰ ۳/۵/۱۰ مسئله

مشخصات کامل کامیون‌هایی که دارای محور بیش از ۲ است.



```

1 -- Tamrin 10
2 SELECT *
3 FROM truck
4 WHERE axil > 2

```

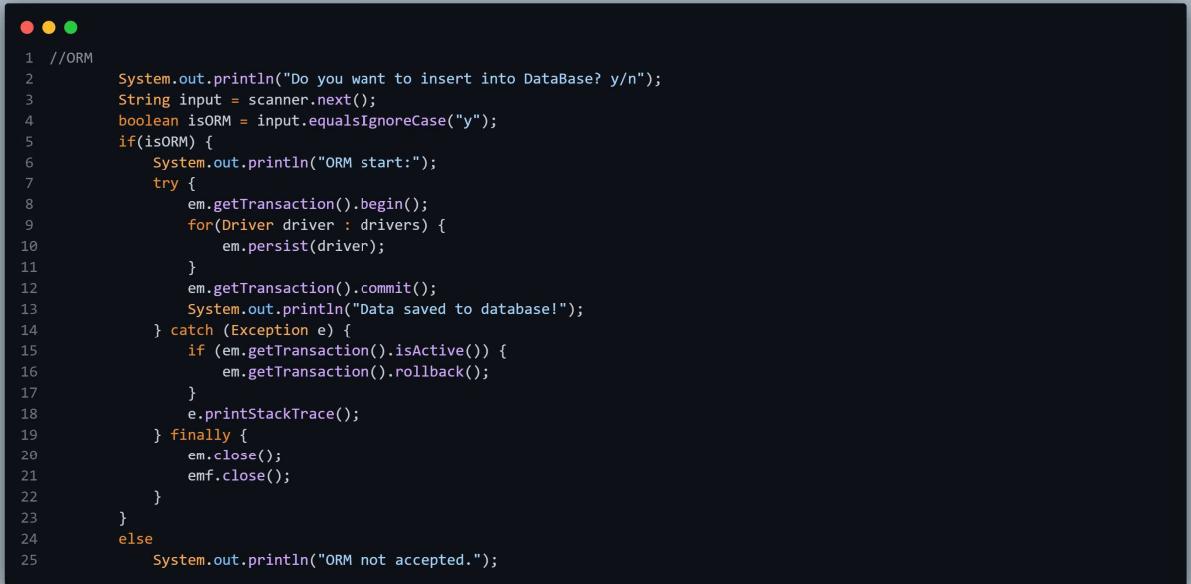
شکل ۱۵ - مسئله ۱۰

۳/۶ پیاده‌سازی ORM

در کد جاوا با توجه به تولید نمونه‌های مختلف، آن‌ها از طریق جاوا به جداول دیتابیس INSERT می‌کنیم. این قطعه کد، نحوه ایجاد و ذخیره داده‌ها در پایگاه داده با استفاده از (Java Persistence API) JPA و Hibernate را نشان می‌دهد. هدف اصلی این کد، تولید اطلاعات مربوط به رانندگان و کامیون‌ها به صورت تصادفی و سپس ذخیره آن‌ها در پایگاه داده است. این کد از ORM برای تسهیل تعامل با پایگاه داده و جلوگیری از نوشتن مستقیم کدهای SQL استفاده می‌کند.

در بخش اول کد، ابتدا یک EntityManagerFactory و سپس یک EntityManager ایجاد می‌شود. EntityManagerFactory مانند یک کارخانه برای تولید EntityManager‌ها عمل می‌کند و بر اساس تنظیمات موجود در فایل persistence.xml (که در سوالات قبلی توضیح داده شد) پیکربندی می‌شود. نیز رابط اصلی برای تعامل با پایگاه داده و انجام عملیات‌های CRUD (ایجاد، خوandن، بهروزرسانی و حذف) بر روی موجودیت‌ها (Entity‌ها) است. سپس، یک لیست از اشیاء Driver ایجاد می‌شود و در یک حلقه، اطلاعات مربوط به هر راننده (نام و سن) به صورت تصادفی تولید و به لیست اضافه می‌شود. در ادامه، در یک حلقه دیگر، برای هر راننده، یک یا دو شیء Truck با پلاک و برنده تصادفی ایجاد شده و به لیست کامیون‌های آن راننده اضافه می‌شود.

در بخش دوم کد، عملیات ذخیره‌سازی داده‌ها در پایگاه داده انجام می‌شود. ابتدا یک تراکنش با استفاده از em.getTransaction().begin آغاز می‌شود. تراکنش‌ها برای تضمین یکپارچگی داده‌ها بسیار مهم هستند. سپس، در یک حلقه، هر Driver با استفاده از متدهای em.persist(driver) در پایگاه داده ذخیره می‌شود. به دلیل تنظیم cascade = CascadeType.ALL در رابطه بین Driver و Truck، با ذخیره هر راننده، کامیون‌های مرتبط با آن نیز به طور خودکار در پایگاه داده ذخیره می‌شوند. پس از ذخیره تمام داده‌ها، تراکنش با استفاده از em.getTransaction().commit() commit() تغییرات در پایگاه داده اعمال می‌شوند. در صورت بروز هرگونه خطأ در حین انجام عملیات‌ها، تراکنش با استفاده از finally em.getTransaction().rollback() rollback منابع EntityManagerFactory و EntityManager را با استفاده از emf.close() و em.close() آزاد می‌شوند تا از نشت حافظه جلوگیری شود. متدهای generatePlate() نیز برای تولید پلاک‌های تصادفی برای کامیون‌ها استفاده می‌شود.



```
1 //ORM
2 System.out.println("Do you want to insert into DataBase? y/n");
3 String input = scanner.next();
4 boolean isORM = input.equalsIgnoreCase("y");
5 if(isORM) {
6     System.out.println("ORM start:");
7     try {
8         em.getTransaction().begin();
9         for(Driver driver : drivers) {
10             em.persist(driver);
11         }
12         em.getTransaction().commit();
13         System.out.println("Data saved to database!");
14     } catch (Exception e) {
15         if (em.getTransaction().isActive()) {
16             em.getTransaction().rollback();
17         }
18         e.printStackTrace();
19     } finally {
20         em.close();
21         emf.close();
22     }
23 }
24 else
25     System.out.println("ORM not accepted.");
```

شکل ۱۶ - پیاده‌سازی *ORM* در جاوا