به نام خدا

تمرین اول پایگاه داده

چمران معینی

۹۹۳۱۰۵۳

۱) بررسی انواع مدل‌های ارائه شده برای دیتابیس‌ها و مزایا و معایب آن‌ها نسبت به مدل رابطه‌ای

از انواع مختلف مدل‌های ارائه شده برای ساخت دیتابیس می‌توان به مدل $Hierarchical$ ، $DocumentOriented$ ، $Network$ ، $Relational$ ، $ObjectOriented$ یا $ObjectRelational$ اشاره کرد که آن‌ها را، با تمرکز بر مدل رابطه‌ای، بررسی خواهیم کرد.

ابتدا مدل رابطه‌ای را بررسی می‌کنیم. در این مدل، داده‌ها را در ساختمان‌داده‌ای شبیه به جدول ذخیره می‌کنیم. برای مثال جدولی از اطلاعات دانشجویان را در نظر بگیرید. در این جدول، هر سطر یک دانشجو را نشان می‌دهد و هر ستون یک ویژگی از دانشجو را نشان می‌دهد. مثلا در ستون اول، نام دانشجو، در ستون دوم، نام خانوادگی، در ستون سوم، شماره دانشجویی، در ستون چهارم، رشته تحصیلی دانشجو و در ستون پنجم دانشکده‌ی دانشجو نمایش داده می‌شود. حال فرض کنید بخواهیم اطلاعات مربوط به رشته‌ها و دانشکده‌ها را نیز ذخیره کنیم. در این صورت می‌توانیم جدول دیگری از اطلاعات رشته‌ها بسازیم که در آن، هر سطر اطلاعات مربوط به یک دانشکده و هر ستون یکی از ویژگی‌های دانشکده‌ها نشان می‌دهد، از جمله نام دانشکده، نام ریاست دانشکده، نام دانشکده‌ی مادر، آدرس دانشکده و ... و یک جدول دیگر از اطلاعات رشته‌ها که هر سطر اطلاعات مربوط به یک رشته و هر ستون یکی از ویژگی‌های رشته‌ها نشان می‌دهد، حال اگر پس از بررسی داده‌های یک دانشجو، نیاز به اطلاعات بیشتری از دانشکده یا رشته‌ی دانشجو داشته باشیم، به کمک فیلدِ مشترکی که هم در جدول دانشجوهاست و هم در جدول رشته‌ها یا دانشکده‌ها، می‌توانیم به اطلاعات بیشتری دست پیدا بکنیم.

در مدل $Hierarchical$ یا همان مدل درختی، که با نام «مدل سلسله مراتبی» نیز شناخته می‌شود، همان طور که از نام این مدل مشخص است، در آن داده‌ها در ساختمان‌داده‌ای مشابه درخت ذخیره می‌شوند. در ریشه که بالاترین سطح درخت است، $RootNode$ قرار دارد. هر یک از گره‌هایی که در سطح بعدی درخت قرار دارد، شامل یک نمونه از داده‌هایی‌ست که قصد ذخیره سازی آن را داریم. برای مثال اگر قصد ذخیره سازی داده‌های مربوط به دانشجویان را داشته باشیم، فیلدهایی نظر $S-id$ ، $S-name$ ، $S-birthday$ و $S-department$ در هر یک از گره‌های این سطح، ذخیره می‌شوند.

حال تصور کنید قصد داشته باشیم که اطلاعات مربوط به دانشکده‌ها را نیز ذخیره کنیم. در این نوع پایگاه داده، برای این کار، یک به یک سراغ گره‌هایی می‌رویم که اطلاعات دانشجویان در آن‌ها ذخیره شده، برای هریک یک گره فرزند می‌سازیم و اطلاعات مربوط به دانشکده‌ی هر یک از دانشجویان را در آن گره ذخیره می‌کنیم.

در مقایسه‌ی این مدل با مدل رابطه‌ای، می‌توان گفت که افزونگی داده‌ها در این مورد بسیار بیشتر اتفاق می‌افتد. فرض کنید ما دویست دانشجو داشته باشیم که همگی در یک دانشکده ساکن هستند. در این روش، ما دویست بار اطلاعات آن دانشکده را نوشته‌ایم، در حالی که در مدل رابطه‌ای، تنها یک بار اطلاعات هر دانشکده را می‌نوشتیم، و تنها با استفاده از کلیدِ هر دانشکده، هر دانشجو را به دانشکده‌اش مپ می‌کردیم. از مزایای آن هم می‌توان به این مورد اشاره کرد که سرعت خواندن داده‌ها در این روش،

می‌تواند بسیار بیشتر از مدل رابطه‌ای باشد. همچنین ساختار ساده‌تری نیز دارد. در مقابل، مشکلاتی نیز پیش می‌آیند، برای نمونه، هر فرزند را تنها به یک والد می‌توان مپ کرد.

از دیگر مدل‌های ارائه شده، می‌توان به مدل $DocumentOriented$ اشاره کرد. در این مدل، دیتابیس ما از تعدادی $index$ تشکیل می‌شود و در هر یک از این ایندکس‌ها، داکیومنت‌هایی قرار می‌گیرند که $mapping$ مشابهی دارند. از نمونه دیتابیس‌های معروف و پرکاربردی که از این مدل استفاده می‌کنند، می‌توان به $elasticsearch$ اشاره کرد. در این مدل که غیررابطه‌ای محسوب می‌شود $non-relational(or NoSQL)$ داده‌های به جای این که در سطرها و ستون‌های مشخصی ذخیره شوند، در داکیومنت‌های منعطفی ذخیره می‌شوند.

به طور کلی، پایگاه‌داده‌های رابطه‌ای ساختارمندتر هستند و معمولا پیش‌بینی پذیری $predictability$ بیشتری دارند، اما در مقابل پایگاه‌داده‌های سند محور، انعطاف‌پذیرتر هستند.

۲) مسئولیت $Datadictionary$ در معماری $DBMS$ دقیقا چیست؟ چه داده‌هایی را در خورد ذخیره می‌کنید؟ اگر این ماژول در معماری $DBMS$ موجود نبود چه مشکلاتی از لحاظ حفظ $dataintegrity$ به وجود می‌آمد؟

دیتا دیکشنری وظیفه‌ی ذخیره سازی متادیتا را برعهده دارد. متادیتا به داده‌هایی گفته می‌شود که توصیف و اطلاعاتی درباره‌ی دیتاهای ما را در خود دارد.

برای مثال اطلاعاتی از قبیل نام جدول‌های دیتابیس، فیلدهای موجود در هر کدام، ویژگی فیلدها از جمله تایپ آن‌ها، از جمله اطلاعاتی هستند که در دیتادیکشنری ذخیره می‌شوند.

حال فرض کنید که به داده‌هایی که در دیتادیکشنری ذخیره می‌کنیم، دسترسی نداشته باشیم. در این حالت $datacorruption$ رخ می‌دهد، چون فیلدها، تایپ مشخصی نخواهند داشت، در نتیجه ممکن است در نمونه‌های مختلف از یک موجودیت، دیتاهای مختلف و ناهماهنگی داشته باشیم. برای مثال ممکن است که در برخی از موارد در فیلدی که باید عدد قرار بگیرد، رشته ذخیره بشود، که در ادامه موجب می‌شود برای انجام عملیاتی مانند میانگین گیری به مشکل بخوریم.