عنوان مقاله: ORM چیست ؟ چرا از آن استفاده می کنیم؟

نویسنده مقاله: تیم فنی نیک‌آموز

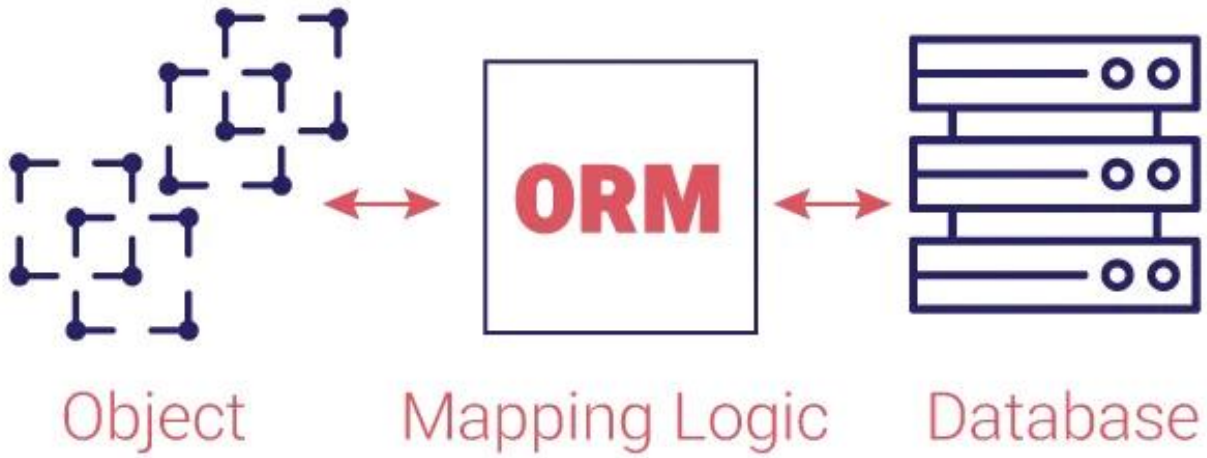
تاریخ انتشار: ۱۹ آذر ۱۴۰۲

منبع: https://nikamooz.com/what-is-orm/

مفهوم ORM و آشنایی با دلایل استفاده از آن در مسیر برنامه نویسی، حائز اهمیت است. در این مقاله، ابتدا به این پرسش پاسخ داده می‌شود که نگاشت شی رابطه ای (Object Relational Mapping) چیست و چه مزایا و معایبی دارد، سپس دلایل استفاده از آن و نحوه دسترسی به ORM مورد بررسی قرار می‌گیرند.

مفهوم Object Relational Mapping چیست ؟

در دنیای توسعه اپلیکیشن و **مدیریت پایگاه داده (RDBMS)**، مفهوم ORM به‌عنوان ابزاری قدرتمند و سازگار ارائه شده است. نگاشت شی رابطه ای یا همان ORM ، پل ارتباطی میان زبان‌های برنامه نویسی شی‌گرا و پایگاه‌های داده رابطه‌ای (RDBMS) محسوب می‌شود و برنامه‌نویسان با استفاده از آن، از روش کارآمد و قابل درک‌تری برای تعامل با سیستم‌های پایگاه داده بهره‌مند خواهند شد. در عمل، به جای آن که برنامه‌نویسان از کوئری‌های SQL خام استفاده کند، با اشیای زبان برنامه‌نویسی مدنظر کار می‌کنند و سیستم ORM، مسئولیت رسیدگی به ترجمه آن‌ها به عملیات پایگاه داده را برعهده خواهد گرفت. برای آشنایی بیشتر با SQL Server پیشنهاد می‌کنیم **مقاله SQL Server چیست؟ کاربرد، مزایا و آموزش SQL Server** را مطالعه نمایید.



مزایای ORM چیست ؟

مزیت های ORM به شرح زیر است:

- **سطح بالای انتزاعات و سادگی:** در ORM، سطح بالای انتزاعات به برنامه‌نویسان اجازه می‌دهد تا با کلاس‌ها و اشیا کار کنند و از دشواری‌های مربوط به کوئری‌های پیچیده SQL دور باشند. این موضوع، کد را ساده‌سازی می‌کند و به خوانایی و قابل نگهداری بودن آن منجر می‌شود.
- **سازگاری چند پایگاه داده‌ای (Cross-Database):** معمولاً فریمورک‌های ORM از چندین سیستم پایگاه داده پشتیبانی می‌کنند. به این ترتیب، برنامه‌نویس می‌تواند کدنویسی را مستقل از پایگاه داده مربوطه انجام دهد و درصورت لزوم، بدون نیاز به تغییرات اساسی در کد، بین انواع پایگاه داده جابه‌جا شود.
- **توسعه سریع:** Object Relational Mapping ، میزان کدهای تکرارشونده موردنیاز برای تعاملات پایگاه داده را کاهش می‌دهد. بنابراین، توسعه‌دهندگان می‌توانند تمرکز اصلی خود را روی ویژگی‌ها و منطق اپلیکیشن قرار دهند و فرآیند توسعه را تسریع کنند.
- **استفاده مجدد از کد:** به‌واسطه تعریف مفهوم ORM ، قابلیت استفاده مجدد از اجزای کد تقویت می‌شود و برنامه‌نویسان می‌توانند نگاشت های شی رابطه ای را تعریف و مجدداً استفاده کنند. این موضوع، ماژولاریتی کد را توسعه می‌دهد.
- **پارادایم شی گرا:** هم‌تراز بودن ORM با مفاهیم برنام نویسی شی گرا (OOP)، به برنامه‌نویسان این امکان را می‌دهد که با اشیا، ارث بری (Inheritance) و کپسوله سازی (Encapsulation) کار کنند.

معایب ORM چیست ؟

مشکلات یا کاستی‌های ORM در ادامه بررسی می‌شوند.

- **سربار کارایی و نگهداری:** ORM یک لایه انتزاعی ایجاد می‌کند که روی کارایی، به ویژه در کوئری‌های پیچیده یا سیستم‌های پرتراکنش، اثر می‌گذارد. توسعه‌دهنده باید به نحوه بهینه‌سازی کوئری‌ها و پیکربندی ORM تسلط داشته باشد. از طرفی، این فریمورک‌ها هنگام به‌روزرسانی یا تطبیق با نسخه‌های جدید دیتابیس، دارای سربار نگهداری هستند.
- **چالش‌های مربوط به آپدیت:** در شرایطی که بخواهیم اپلیکیشن را در راستای آپدیت‌های مربوط به ORM ، به‌روزرسانی کنیم، احتمالاً چالش‌هایی به وجود می‌آید.
- **انعطاف‌پذیری محدود:** انعطاف‌پذیری فریمورک‌های ORM در کار با با کوئری‌های پیچیده و ویژگی‌های خاصی از پایگاه داده محدود هستند. ممکن است برنامه‌نویسان به نوشتن کوئری‌های Custom شده روی بیاورند یا به‌طور کامل، ORM را دور بزنند.
- **عدم وجود کنترل:** فریمورک‌های ORM از لایه SQL صرف نظر می‌کنند. این موضوع زمانی مشکل ایجاد می‌کند که برنامه‌نویسان بخواهد به‌طور دقیق، روی پایگاه داده کنترل داشته باشند. به بیان دیگر، ممکن است ORM اجازه به‌کارگیری برخی کوئری‌ها یا دستورات را به برنامه‌نویسان ندهد و به دنبال آن، کنترل آن‌ها محدود شود.

اکنون که مفهوم ORM و مزایا و معایب آن را به‌خوبی درک کرده‌اید، احتمالاً این سؤال برایتان پیش می‌آید که چرا از ORM استفاده می‌شود؟ در بخش بعدی، به این پرسش پرداخته خواهد شد.

چرا از ORM استفاده می کنیم؟

دلایل مختلفی برای استفاده از ORM وجود دارد که در ادامه فهرست شده‌اند.

- **انتزاعی‌سازی پیچیدگی پایگاه داده:** وجود یک لایه انتزاعی روی پایگاه داده مورد استفاده، به توسعه‌دهندگان این امکان را می‌دهد که با اشیا و کلاس‌ها کار کنند و با جداول، سطرها و کوئری‌های پایگاه داده، سروکار نداشته باشند. این انتزاعی بودن، تعاملات با پایگاه داده را تسهیل می‌بخشد و قابلیت نگهداری (Maintainability) و خوانایی (Readability) را بهبود خواهد داد.
- **بهبود میزان بهره‌وری:** ابزارهای ORM می‌توانند کدنویسی موردنیاز برای عملیات پایگاه داده را به‌طور قابل توجه کاهش دهند. توسعه‌دهنده می‌تواند با استفاده از متدهای یک زبان سطح بالا، زمان خود را به‌صورت بهینه‌تری صرف کند.
- **استقلال پایگاه داده:** فریمورک‌های ORM ، تا حدی استقلال پایگاه داده را ارائه می‌دهند. توسعه‌دهنده می‌تواند کدی بنویسد که با ORM کار کند و ترجمه این کد به عملیات SQL ، برعهده ORM باشد. این موضوع، جابه‌جایی میان انواع پایگاه داده را بدون نیاز به بازنویسی همه لایه Data Access، تسهیل می‌دهد.

دسترسی به ORM چگونه است؟

به منظور دسترسی به ORM ، کافی است مراحل زیر را دنبال کنید.

۱- انتخاب ORM

در مرحله اول، لازم است ابزار Object Relational Mapping مدنظر خود را انتخاب کنید. فریمورک‌های ORM گوناگونی برای **انواع زبان های برنامه نویسی** وجود دارند. به‌عنوان مثال، برای زبان برنامه‌نویسی جاوا، Hibernate، EclipseLink و MyBatis، برای زبان سی شارپ، **Entity FrameWork** و **NHibernate** و **Dapper** به‌عنوان فریمورک ORM معرفی شده‌اند. هرچند مفهوم ORM و کاربرد آن کاملاً واضح است، اما انتخاب یکی از میان گزینه‌های موجود، به زبان برنامه‌نویسی و نیازمندی‌های پروژه شما بستگی دارد.

۲- نصب فریمورک ORM

شما باید براساس زبان برنامه‌نویسی، فریم ورک ORM را برروی سیستم خود نصب کنید. این مرحله، معمولاً توسط **Package Manager** یا ازطریق کتابخانه ORM موجود در Dependency های پروژه‌تان انجام می‌شود.

۳- پیکربندی ORM

اکنون باید اقدامات لازم به منظور کانفیگ کردن ORM را انجام دهید تا به پایگاه داده متصل شود. به‌طور معمول، این مرحله مواردی همچون تعیین نوع پایگاه داده، جزئیات اتصال و نوع تنظیمات دیگری را شامل می‌شود. ممکن است پیکربندی ORM ازطریق یک فایل کانفیگ، کد یا هر دوی آن‌ها انجام شود.

۴- تعریف مدل ها و موجودیت ها

در Object Relational Mapping ، مدل‌ها یا موجودیت ها (Entities)، اشیایی را نمایش می‌دهند که در پایگاه‌های داده ذخیره خواهند شد. تعریف این مدل‌ها، با ایجاد کلاس‌ها و ساختارهایی ممکن می‌شود که به جداول پایگاه داده نگاشت می‌شوند. هر صفت (Attribute) در کلاس، با یک ستون از جدول مرتبط است.

۵- اجرای عملیات CRUD

با استفاده ORM ، می‌توان عملیات CRUD، یعنی ساخت (Create)، خواندن (Read)، به‌روزرسانی (Update) و حذف (Delete) را اجرا کرد.

۶- رسیدگی به ارتباط ها (Relationships)

در صورتی که بین جداول پایگاه داده رابطه وجود داشته باشد، بهتر است در مدل‌های پروژه، تعریف و رسیدگی به این رابطه‌ها انجام شود. معمولاً این کار، ازطریق سازوکارهای خاص مربوط به ORM انتخابی شما، امکان‌پذیر خواهد بود.

۷- رسیدگی به تراکنش ها و خطاها

زمانی که شما با پایگاه داده کار می‌کنید، بررسی صحیح تراکنش‌ها و همچنین، پیاده‌سازی Error Handling به منظور مدیریت استثناعات (Exceptions) حائز اهمیت است. درواقع، وجود Error Handling برای مواقعی مناسب است که Exception در عملیات پایگاه داده رخ دهد.

۸- بهینه سازی کوئری ها (انتخابی)

برخی از فریمورک های ORM ، سازوکارهای خاصی به‌منظور بهینه‌سازی کوئری ها دارند. به‌عنوان مثال، استفاده از Lazy Loading، ایندکس گذاری (Indexing) و Eager Loading از این گروه اقدامات محسوب می‌شوند. شما با یادگیری چنین ویژگی‌هایی، این امکان را دارید که کارایی تعاملات پایگاه داده خود را بهبود دهید.

سخن پایانی: نگاهی بر Object Relational Mapping

استفاده از Object Relational Mapping دلایلی مختلفی دارد و به‌واسطه آن، یک رویکرد سطح بالا و انتزاعی برای تعامل با پایگاه داده رابطه‌ای فراهم می‌شود. در این مقاله، مفهوم ORM و کاربردهای آن به‌همراه مزایا و معایب شرح داده شدند. هرچند ORM مزیت‌های مخصوص به خود را دارد، اما لزوماً مناسب تمام پروژه‌ها نیست. درواقع، انتخاب میان ORM و سایر روش‌ها، به نیازمندی‌ها و محدودیت‌های مخصوص پروژه بستگی دارد.