УДК 004.045

СПОСОБИ ВЗАЄМОДІЇ З БАЗОЮ ДАНИХ

Іванов Євгеній

Науковий керівник: канд. техн. наук, доцент кафедри інформатики, Баранюк О.Ф.

Центральноукраїнський державний педагогічний університет імені Володимира Винниченка, м. Кропивницький, Україна

У статті розглянута проблема вибору способу взаємодії з базою даних, здатного надати найкращі можливості для розроблюваної системи. Проілюстровано способи взаємодії з базою даних через програмний код за допомогою використання збережених процедур, будівельника запитів, об'єктно-реляційного відображення та мови структурованих запитів. Проаналзовано переваги та недоліки кожного із способів взаємодії, проведено їх порівняння, що дає змогу зробити вірний вибір для вирішення актуальних питань успішної розробки та підтримки системи.

Ключові слова: база даних, збережена процедура, будівельник запитів, об'єктно-реляційне відображення, мова структурованих запитів.

Ways of interaction with database

E. Ivanov

Scientific supervisor: Candidate of Technical Sciences, Associate Professor of the Department Informatics, Baranyuk O.F.

The Volodymyr Vynnychenko Central Ukrainian State Pedagogical University, Kropyvnytsky, Ukraine

The article deals with the problem of choosing a method of interaction with a database capable of providing the best opportunities for the system development. The ways of interacting with the database through program code using the stored procedures, the query builder, the object-relational mapping, and the language of structured queries. The advantages and disadvantages of each method of interaction are described, their comparison is made, which makes it possible to make the right choice for solving actual problems of successful system development and support.

Keywords: database, stored procedure, query builder, object-relational mapping, structured query language.

Постановка проблеми. Процес вибору способу взаємодії з базою даних активізує широке коло питань, серед яких: швидкість обробки запитів, завантаженість системи, розмежування доступу до даних, швидкість

розробки, рівень необхідних знань для здійснення розробки, легкість у відлагодженні та багато інших питань, які потребують вирішення для вибору вірного архітектурного рішення при побудові систем на основі баз даних. Потужний внесок у вирішення проблеми взаємодії з базою даних здійснений за допомогою створення патернів проектування та програмних засобів, які надають методи, як для підготовки запитів, так і для їх побудови. Вірний вибір засобу взаємодії з базою даних надає змогу вирішити актуальні питання розроблювальної системи.

Стан дослідження. Однією з основних праць, присвячених аналізу взаємодії об'єктно-орієнтованих програм з реляційною базою даних є робота М. Фусселя [5], що надихнула багатьох людей на створення пакетованих засобів об’єктно-реляційного відображення (ORM). В роботі М. Фусселя наводяться основні концепції необхідні для здійснення відображення реляційних даних в об’єктні. Не менше важливими є запропоновані патерни проектування для взаємодії з базою даних М. Фаулером [4]. Стюарт Тіль у своїй роботі [7] наводить способи застосування та варіанти удосконалення патернів проектування запропонованих М. Фаулером. Д. Нільссоном у книжці [6] запропонована схема декомпозиції програмної системи, що передбачає активне використання збережених процедур.

Наведенні вище дослідження основних концепцій лягли в основу ORM та будівельника запитів, також вони активно застосовуються в розробці з використанням збережених процедур та структурованих мов запитів (SQL), дозволяючи будувати нові шари програмної системи, збільшуючи рівень абстракції при взаємодії з базою даних. Можна виділити чотири основні способи взаємодії з базою даних: ORM, будівельник запитів, збережені процедури та SQL. Кожен із способів взаємодії з базою даних може надавати як переваги, так і недоліки при розробці та підтримці програмних продуктів. Невірний вибір засобу може завдати великих збитків через накладені ним обмеження, а комбінування декількох засобів може значно підвищити складність розроблюваної системи, завадивши її успішному та своєчасному

випуску. Саме тому дослідження є актуальним, адже дає змогу зрозуміти які способи, де і як краще використовувати.

Мета дослідження. Метою дослідження є огляд та порівняльний аналіз способів взаємодії з базою даних, здатних забезпечити переваги в швидкості обробки запитів, розмежування доступу до даних, швидкості розробки, рівні необхідних знань для здійснення розробки, легкості у відлагодженні.

З початку розвитку програмування було розроблено велику кількість підходів для опису різних предметних областей, які доповнювали мови програмування. Одним із найважливіших підходів є об'єктно-орієнтоване програмування (ООП), що помітно спрощує написання та читання програм. Об'єктно-орієнтоване програмування – це метод програмування, оснований на представленні програми у вигляді сукупності взаємодіючих об'єктів, кожен із яких є екземпляром певного класу, а класи являються членами певної ієрархії наслідування [3]. Іншим, не менш важливим підходом, став реляційний підхід до організації бази даних, що дозволив програмістам абстрагуватися від деталей організації фізичного зберігання даних, через структури логічного рівня та мову структурованих запитів (SQL). Згідно з назвою даного підходу основною особливістю являються зв'язки, це означає, що крім повідомлень базі даних про те, яку інформацію потрібно зберегти, також необхідно повідомляти як ця інформація буде взаємодіяти з іншими інформаційними частинами. Необхідно не лише користуватися цією мережею підключень, а також повідомляти базі даних про те як буде сконструйована дана мережа [2]. Даний підхід задовольняє потреби більшості програм в збережені даних, які зручно представити у вигляді таблиці і зв'язків між ними, через що даний підхід здобув велику популярність.

Ці два підходи відносяться до логічного рівня проектування програмних систем, що значить те, що вони представляють дві різні точки зору на одну й ту ж сутність і використовуються в більшості випадків при створені веб-додатків, у яких необхідно виводити динамічно змінювану

інформацію. Через це потрібно виконувати реляційне відображення таблиць на об’єкти та навпаки. Отримання, зміну, оновлення або видалення об'єктів, можна здійснювати через ORM, будівельник запитів, збережені процедури та SQL.

Збережена процедура – підпрограма, доступна застосункам, які мають доступ до системи керування реляційними базами даних (СКРБД). Такі процедури зберігаються у словнику даних бази. Використання збережених процедур дозволяє здійснювати гнучке управління правами доступу, таким чином користувач буде мати доступ лише до можливостей, які надаються збереженою процедурою і не зможе здійснювати недозволені дії в базі даних через помилки в серверному або клієнтському коді. Також покращується безпека завдяки усуненню можливості здійснювати SQL-ін’єкції, при умові що у збереженій процедурі не будуть виконуватися динамічні запити з параметрами, переданими від клієнта без попередньої обробки та екранування, або видалення небезпечних фрагментів. При статичному виконанні коду в збереженій процедурі захист від SQL-ін’єкцій досягається за рахунок того, що параметри які передаються до збереженої процедури представляють собою звичайні змінні на відмінну від інших методів, де параметри з’єднуються з іншою частиною запиту і можуть містити в собі шкідливий код, який стане частиною запиту, та буде виконаний. Щоб завадити його виконанню, потрібно виконати міри захисту – екранувати або видалити небезпечні фрагменти. Більшість засобів можуть виконувати екранування автоматично або за допомогою певних, додаткових засобів. Крім того використання збережених процедур надає можливість приховувати структури даних, що дозволить при необхідності здійснювати зміни в базі даних, не виконуючи зміни в програмному коді. Користь від приховання структури даних особливо помітні коли серверним кодом і базою даних займаються різні люди. При використанні збережених процедур з’являється можливість повторного використання коду, що дозволяє значно зекономити час на розробку. Також однією з найбільших переваг використання

збережених процедур є швидкість їх виконання завдяки тому, що всі необхідні дії будуть виконані СУБД сервера, і лише необхідні дані будуть повернені у відповідь на запит, а також завдяки компіляції, під час якої буде здійснена семантична та синтаксична перевірка коду. Крім пришвидшення виконання, попередня семантична та синтаксична перевірка покаже можливі помилки, що може допомогти зекономити час при відлагодженні. Крім плюсів, використання збережених процедур надає і мінуси, з яких основним є розмиття бізнес логіки та дублювання коду з невеликими відмінностями. Дану проблему можна вирішити за допомогою ускладнення збереженої процедури завдяки передачі додаткових параметрів, але таке рішення проблеми може призвести до того, що швидкість виконання збереженої процедури впаде, а також стане важко вносити зміни через занадто складну логіку. Також використання збережених процедур значно збільшують складність розробки, що впливає на швидкість розробки всієї системи та вартість програмного продукту. Основною причиною даної проблеми є те, що мова для написання збереженої процедури може надавати значно менше можливостей в порівнянні з мовою, що використовується для написання додатка, який здійснює запити на отримання інформації з бази даних. Також збільшення складності відбувається і через необхідність перетворювати данні до необхідного вигляду на відміну від ORM, що виконують перетворення даних автоматично.

Будівельник запитів надає зручний інтерфейс для створення та виконання запитів до бази даних. Він може бути використаний для виконання більшості типів операцій [1]. За допомогою будівельника запитів можна виконувати ініціалізацію структур даних більш наглядно, ніж за допомогою ORM. Будівельник запитів представляє собою засіб для доступу до бази даних, проміжний між звичайними SQL запитами та ORM. На відмінну від ORM, за допомогою будівельника запитів взаємодія з базою даних відбувається швидше. Покращення в швидкості роботи може бути помітним, коли відбувається читання великої кількості записів. Через

покращення швидкості роботи, будівельник запитів не містить багатьох можливостей які надають ORM, серед яких можливість завантаження даних по необхідності, обробка та структурування результату взаємодії, події моделі, часові мітки.

Будівельник запитів створюється за допомогою об'єктів доступу до даних (DAO) та дозволяє конструювати SQL вираз в програмованому і незалежному від СУБД вигляді. У порівнянні з написанням чистого SQL виразу, використання будівельника допомагає отримати кращу читабельність коду та здійснювати генерування більш безпечних SQL виразів.

ORM – технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних». Застосування об'єктно-реляційних відображень ORM в даний час є загальним засобом в процесі розробки систем, що дозволяє об'єднати об'єктно-орієнтовану модель представлення даних з реляційною [3].

ORM бере на себе складність прямого та зворотного перетворення об’єктів в реляційну модель. При використанні ORM для реалізації простого функціоналу, який не включає в себе складних відношень між великою кількістю таблиць та складних умов фільтрації даних необхідно писати значно менше коду, що підвищує швидкість розробки та зрозумілість коду для подальшої підтримки. Не потрібно самому виконувати перетворення поверненого результату та виконувати групування якщо необхідно отримати результат з декількох таблиць. Використання ORM зменшує кількість часу, що програміст витрачає на виконання тестування та відлагодження коду, через позбавлення від необхідності опрацьовувати значну кількість програмного коду, який часто схожий та схильний до помилок, надає можливість змінити реалізацію бази даних, через незалежність коду додатка від бази даних. Крім того використання ORM може забезпечити опис схеми бази даних, представленої за допомогою мови програмування якою виконується розробка. Розвинені ORM можуть допомогти у вирішенні проблеми відображення успадкованих об’єктів на таблиці бази даних.

Найбільший недолік ORM, у порівнянні з іншими підходами, це низька швидкодія, спричинена тим, що ORM призначені для обробки великої кількості сценаріїв різного роду, більшість з яких ніколи не буде використано. Крім того ORM можуть відрізнятися для різних фреймворків і тим паче мов програмування, через що навантаження на програміста при використанні нового засобу може збільшитися. Має місце підвищена складність відлагодження, якщо сама ORM містить помилку.

В основі ORM може лежати патерн проктування Active Record або Data Maper. ORM, які для здійснення відображення з реляційних даних в об’єкти використовують патерн Active Record, є більш простими та потребують значно менших зусиль для роботи з ними через те, що цей патерн використовує найбільш очевидний підхід – зберігання логіки доступу до даних в об'єкті [4]. Патерн Data Mapper – це програмний шар, що розділяє об'єкт і БД. Його обов'язок – пересилати дані між ними та ізолювати їх один від одного. При використанні даних Mapper'а об'єкти не потребують знання про існування БД. Вони не потребуються в SQL-коді, і інформації про структуру БД [4]. ORM, що використовують патерн проектування Active Record найкраще використовувати при розробці за концепцією швидкої розробки додатків (RAD). Концепція розробки RAD передбачає, що спочатку проекту створюється база даних а потім програмний код, через що досить просто розгорнути доменну модель по існуючій схемі бази даних.

SQL – декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних. Сама по собі SQL не є ані системою керування базами даних, ані окремим програмним продуктом.

Використання SQL надає незалежність від того, які засоби використовуються для створення додатка, на відміну від будівельника запитів та ORM, а незважаючи на наявність діалектів, використання SQL може надати незалежності від баз даних на відміну від збережених процедур

які використовують певну мову програмування. За допомогою SQL можна більш точно задавати, які дані необхідно отримати з бази даних, на відміну від ORM.

Недоліком при використанні SQL у порівняння з іншими способами доступу до бази даних є значне дублювання коду, велика ймовірність виконання шкідливого коду, низька зрозумілість коду при збільшенні складності предметної області. Для складних систем при використанні SQL застосовують такі патерни проектування як: Row Data Gateway, Active Record, Table Data Gateway, Data Mapper для відмежування джерела даних від бізнес логіки.

Висновки. В результаті проведеного порівняльного аналізу способів здійснення взаємодії з базою даних за допомогою використання ORM, будівельника запитів, збережених процедури та SQL зазначено переваги одного способу над іншим. Виявлено, що вибір вірного способу взаємодії з базою даних може позбавити від проблем або надати переваги у вирішенні проблем, що виникають при розробці та підтримці програмних продуктів.

Враховуючи вищесказане, для систем, у яких необхідна швидкість розробки, невисокий рівень знань для здійснення розробки, легкість у відлагодженні, краще обирати ORM. Для систем, у яких важлива швидкість обробки запитів, завантаженість системи, розмежування доступу до даних, краще обирати для взаємодії з базою даних збережені процедури.

Список літератури

1. Конструктор запросов [Електронний ресурс] // Laravel – Режим доступу до ресурсу: http://laravel.su/docs/5.0/queries.

2. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство / Бретт Маклафлин. – Санкт-Петербург: Питер, 2013. – 544 с.

3. Объектно-ориентированный анализ и проектирование с примерами приложений / [Г. Буч, М. Роберт, М. Энгл и другие]. – Хьюстон: Вильямс, 2008. – 720 с.

4. Fowler M. Patterns of Enterprise Application Architecture / Martin Fowler. – Boston: Addison-Wesley Professional, 2011. – 560 с.

5. Fussell M. Foundations of Object-Relational Mapping [Електронний ресурс] / Mark L. Fussell // ChiMu Corp. – 1997. – Режим доступу до ресурсу: http://markfussell.emenar.com/blog/object-relational/.

6. Nilsson J. .NET Enterprise Design with Visual Basic .NET and SQL Server 2000 / Jimmy Nilsson. – Indianapolis: Sams Publishing, 2001. – 384 с.

7. Thiel S. Enterprise application design patterns : improved and applied / Thiel Stuart – Canada, 2010. – 145 с