УДК 681.004.4

М.Ю. Лосєв, Ю.І. Скорін

Харківський національний економічний університет, Харків

ОСОБЛИВОСТІ ДОСТУПУ ДО ДАНИХ SQL SERVER ЗА ДОПОМОГОЮ ADO.NET ENTITY FRAMEWORK

У роботі розглядається особливості використання технології ADO.NET Entity Framework (EF) для полегшення побудови застосувань доступу до даних. При цьому визначаються основні компоненти платформи Entity Framework, які реалізують принцип проектування в моделюванні даних, що полягає в розділенні моделі даних на концептуальну модель, логічну модель і фізичну модель.

Ключові слова: *застосування, реляційна схема, концептуальна модель, середовище програмування, компонент, семантика, обмеження.*

Вступ

ADO.NET Entity Framework (EF) – об'єктно-орієнтована технологія доступу до даних, є object-relational mapping (ORM) рішенням для .NET Framework від Microsoft [1]. Надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL. Для полегшення побудови застосувань використовується як ADO.NET Data Services, так і зв'язка з Windows Communication Foundation і Windows Presentation Foundation, що дозволяє будувати багаторівневі застосування, реалізовуючи один з шаблонів проектування MVC, MVP або MVVM.

Платформа ADO.NET Entity Framework дозволяє розробникам створювати застосування для доступу до даних, працюючі з концептуальною моделлю застосування, а не безпосередньо з реляційною схемою зберігання. Її метою є зменшення об'єму коду і зусиль по обслуговуванню застосувань, орієнтованих на обробку даних. Застосування Entity Framework дають наступні переваги [1]:

застосування можуть працювати з концептуальною моделлю в термінах наочної області – зокрема з успадкованими типами, складними елементами і зв'язками;

застосування звільняються від жорстких залежностей від конкретного ядра СУБД або схеми зберігання;

зіставлення між концептуальною моделлю і схемою, специфічною для конкретного сховища, можуть мінятися без зміни коду застосування;

розробники мають можливість працювати з узгодженою моделлю об'єктів застосування, яка може бути зіставлена з різними схемами зберігання, які, можливо, реалізовані в різних системах управління даними;

декілька концептуальних моделей можуть бути зіставлені з єдиною схемою зберігання;

підтримка інтегрованих в мову запитів (LINQ) забезпечує під час компіляції перевірку синтаксису запиту щодо концептуальної моделі.

Основний матеріал

Платформа Entity Framework є набір технологій ADO.NET, які забезпечують розробку застосувань, пов'язаних з обробкою даних [1].

Архітекторам і розробникам застосувань, орієнтованих на обробку даних, доводиться враховувати необхідність досягнення двох абсолютно різних цілей. Вони повинні моделювати суттевості, зв'язок і логіку вирішення бізнес-завдань, а також працювати з ядрами СУБД, які використовуються для збереження і отримання даних.

Дані можуть розподілятися по декількох системах зберігання даних, в кожній з яких застосовуються свої протоколи, але навіть в застосуваннях, що працюють з однією системою зберігання даних, необхідно підтримувати баланс між вимогами системи зберігання даних і вимогами написання ефективного і зручного для обслуговування коду застосування.

У Entity Framework розробники дістають можливість працювати з даними, представленими у формі об'єктів, що відносяться до конкретних доменів, і властивостей, таких як клієнти і їх адреси, не будучи вимушеними звертатися до базових таблиць і стовпців бази даних, де зберігаються ці дані. Така можливість з'являється завдяки переходу на вищий рівень абстракції, на якому розробники можуть працювати з даними, застосовуючи менший об'єм коду для створення і супроводу застосувань, орієнтованих на роботу з даними.

Платформа Entity Framework є компонентом .NET Framework, тому застосування Entity Framework можна запускати на будь-якому комп'ютері, на якому встановлений .NET Framework 3.5 з пакетом оновлення SP1.

Давно відомий і широко вживаний принцип проектування в моделюванні даних полягає в розділенні моделі даних на наступні три частини: концептуальна модель, логічна модель і фізична модель.

Концептуальна модель визначає суттевості і зв'язки в модельованій системі.

Логічна модель для реляційної бази даних забезпечує нормалізацію суттевостей і зв'язків в цілях створення таблиць з обмеженнями зовнішнього ключа.

У фізичній моделі враховуються можливості конкретної системи обробки даних шляхом визначення залежних від ядра бази даних докладних відомостей про зберігання даних, які торкаються секціонування і індексування.

Фізична модель удосконалюється адміністраторами бази даних з метою підвищення продуктивності, але програмісти, які розробляють код застосування, в основному вимушені обмежуватися роботою з логічною моделлю, готуючи SQL-запити і викликаючи процедури, що зберігаються.

Концептуальні моделі в основному використовуються як інструмент для уявлення і обміну думками з приводу вимог до застосування, тому найчастіше служать як майже незмінні схеми, які розглядаються і обговорюються на ранніх стадіях проекту, після чого перестають бути предметом уваги.

Платформа Entity Framework додає значущість концептуальним моделям, дозволяючи розробникам виконувати запити до суттевостей і зв'язків в концептуальній моделі, при цьому для перекладу цих операцій в команди, залежні від джерела даних, застосовується сама платформа Entity Framework. Це дозволяє відмовитися від використовування в застосуваннях жорстко заданих залежностей від конкретного джерела даних.

Концептуальна модель, модель зберігання даних і їх зіставлення виражаються в зовнішній специфікації, відомій також як модель Entity Data Model (EDM).

Модель зберігання і зіставлення можуть змінюватися за необхідністю, не вимагаючи змін в концептуальній моделі, класах даних і коді застосування.

Моделі зберігання даних залежать від постачальника, тому можна працювати з узгодженою концептуальною моделлю через різні джерела даних.

Модель EDM визначається наступними трьома файлами моделі і зіставлення, які мають відповідні розширення імен файлів:

файл на мові CSDL (з розширенням CSDL) визначає концептуальну модель;

файл на мові SSDL (з розширенням SSDL) визначає модель зберігання даних, яка називається також логічною моделлю;

файл на мові MSL (з розширенням MSL) визначає зіставлення моделі зберігання і концептуальної моделі.

У Entity Framework ці файли моделі і зіставлення на основі XML використовуються для перетворення операцій створення, читання, оновлення і видалення, що виконується над суттевістю і зв'язками концептуальної моделі, в еквівалентні операції в джерелі даних (рис. 1).

Модель EDM підтримує навіть зіставлення суттевості в концептуальній моделі з процедурами, що зберігаються, в джерелі даних.

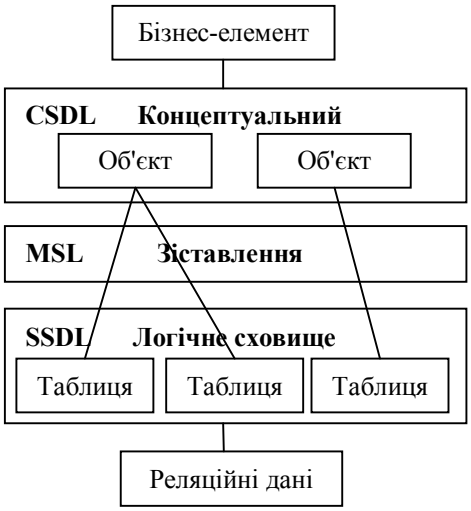

Рис. 1. Розробка сутнісної моделі даних

У моделі EDM сутвості і їх зв'язок моделюються з використанням двох основних типів:

EntityType - абстрактна специфікація докладних відомостей про структуру даних в домені застосування;

AssociationType - логічне з'єднання між типами.

Схема проектування моделі EDM визначає структуру, семантику, обмеження і зв'язки суттевостей в домені застосування. У реалізації служб об'єктів моделі EDM концептуальна схема зіставляється з EDM-схемою, яка містить метадані, що описують модель зберігання (звичайне це таблиці в базі даних). Концептуальна схема використовується для

формування класів програмованої моделі об'єктів, яка застосовується в коді застосування. Концептуальна схема і схема зберігання використовуються також в технології Entity Framework для перевірки, виконання запитів і оновлення даних застосування під час виконання.

При використанні об'єктно-орієнтованого програмування для взаємодії з системами зберігання даних виникають складнощі. Безумовно, організація класів часто нагадує організацію таблиць реляційної бази даних, але така відповідність не ідеальна. Декілька нормалізованих таблиць часто відповідають єдиному класу, а зв'язки між класами представлені інакше, ніж зв'язки між таблицями.

В існуючих рішеннях була зроблена спроба усунути цей розрив, часто званий "невідповідністю типів даних" (impedance mismatch), шляхом зіставлення з реляційними таблицями і стовпцями тільки об'єктно-орієнтованих класів і властивостей. Замість даного традиційного підходу в Entity Framework реляційні таблиці, стовпці і обмеження зовнішнього ключа логічних моделей перетворюються у суттевості і зв'язки концептуальних моделей. Це дозволяє досягти більшої гнучкості при визначенні об'єктів і оптимізації логічної моделі. За допомогою інструментів Entity Data Model формуються розширювані класи даних, засновані на концептуальній моделі. Ці класи є класами, що розділяються, які можуть бути розширені за допомогою додаткових членів, доданих розробником. Класи, сформовані для певної концептуальної моделі, є похідними від базових класів, що надають служби об'єктів для матеріалізації суттевостей у вигляді об'єктів, а також для відстежування і збереження змін. Розробники можуть використовувати ці три класи для роботи з суттевостями і зв'язками як з об'єктами, зв'язаними властивостями навігації.

Наступні функції і компоненти платформи Entity Framework працюють спільно для забезпечення середовища програмування.

Модель Entity Data Model (EDM) служить центром застосування Entity Framework. Вона задає схему проекту, яка використовується для побудови програмованих класів, що використовуються кодом застосування.

Компонент Object Services дозволяє програмістам працювати з класами CLR, створеними з концептуальної моделі. Він також забезпечує інфраструктурну підтримку для застосування Entity Framework, надаючи такі служби, як управління станом, відстежування змін, дозвіл ідентифікаторів, завантаження і перехід по зв'язках, розповсюдження змін об'єктів у модифікації бази даних, а також підтримку запитів для Entity SQL.

Компонент LINQ to Entities забезпечує підтримку LINQ при запитах до суттевостей. Цей компонент дозволяє розробникам писати запити до бази даних на одній з підтримуваних мов програмування .NET Framework, наприклад Visual Basic або Visual C#.

Мова Entity SQL подібна мові SQL і не залежить від типу сховища. Вона призначена для створення запитів до складних об'єктів, заснованих на моделі EDM, а також для управління ними [2].

Постачальник EntityClient розширює модель постачальника ADO.NET шляхом доступу до даних в термінах суттевостей і зв'язків концептуальної моделі. Він виконує запити, які використовують мову Entity SQL. Entity SQL надає базову мову запитів, яка дозволяє постачальнику EntityClient зв'язуватися з базою даних.

Компонент метаданих ADO.NET управляє метаданими для всієї платформи Entity Framework як під час розробки, так і під час виконання програм. Всі метадані, пов'язані з моделями і зіставленнями, доступні через інтерфейси метаданих, які не залежать від механізму, який використовується для зберігання метаданих. Поточний механізм зберігання використовує файли, які засновані на трьох діалектах XML: мові CSDL, мові SSDL і мові MSL.

Застосування Entity Framework включає набір засобів, які створюють зіставлення і класи, що розділяються, представляють суть концептуальної моделі, що змінюється. Застосування Entity Framework включає оновлений постачальник даних SqlClient, який підтримує канонічні дерева команд.

Разом з середовищем виконання Entity Framework .NET Framework 3.5 з пакетом оновлення 1 включає генератор моделей EDM (EdmGen.exe). Програма командного рядка з'єднується з джерелом даних і формує модель EDM на основі зіставлення типу "один до одного" між суттевостю і таблицями. У цій програмі використовується також файл концептуальної моделі (з розширенням CSDL) для формування файлу рівня об'єктів, що містить класи, які представляють типи суті і контекст ObjectContext.

Visual Studio 2010 включає обширний набір інструментів для створення і обслуговування моделі EDM в застосуванні. Конструктор Entity Data Model підтримує створення вдосконалених сценаріїв зіставлення (таких як спадкоємство типу "одна таблиця на тип" і "одна таблиця на ієрархію"), а також розділення суттевості, яка зіставлена з декількома таблицями.

Створення сутнісної моделі даних можна виконувати з допомогою майстра EDM, що надає список об'єктів, які можливо змоделювати. При цому допускається додавання всіх таблиць в модель, тим самим провівши взаємно-однозначне зіставлення таблиць суттевостям [3]. Крім створення файлів CSDL, SSDL і MSL для всіх таблиць бази даних, майстер

також створить набір класів, грунтуючись на CSDL, що представляє модель.

Доступ до концептуальної моделі Entity Framework може бути організований трьома способами:

з допомогою EntityClient;
з допомогою Object Services;
з допомогою LINQ to Entities.

EntityClient абстрагований від логічного сховища, оскільки він взаємодіє з концептуальною моделлю за допомогою своєї власної текстової мови, що називається Entity SQL. Всі запити Entity SQL, що виконуються через EntityClient, компілюються в дерева команд, що посилаються до сховища. Перетворення запитів Entity SQL через концептуальну модель і далі в сховищі забезпечується Entity Framework.

Класи EntityClient схожі на класи поширених постачальників ADO.NET [3]. Наприклад, запити EntityClient виконуються в об'єкті EntityCommand, якому необхідний об'єкт EntityConnection для підключення до EDM. Хоча EntityClient взаємодіє з суттевостю EDM, він не повертає екземпляри суттевості, а натомість повертає всі результати у вигляді об'єкту DbDataReader. За допомогою DbDataReader EntityClient може повертати стандартний набір рядків і стовпців, або представлення складнішої ієрархії даних.

Іншим способом взаємодії з даними, представленими EDM, є використання служб Object Services. Служби Object Services надають можливість завантажувати об'єкти і слідувати будь-яким зв'язкам, визначеним в EDM. Служби Object Services використовують EntityClient для отримання даних. Служби Object Services додають дозвіл ідентифікаторів, що при використанні DataSet доводиться робити уручну. Вони також забезпечують незмінність об'єктів і відстежування змін через події, щоб дозволяти явні завантаження і збереження даних. За рахунок цього знижується число звернень до сервера.

Служби Object Services дозволяють повертати списки об'єктів безпосередньо. При цьому можна повертати як проекції суттевості, так і певну суттевість. Наприклад, користуючись Object Services, можна одержати об’єкт List<Customers>, як визначено в EDM. Об'єкти Customers можна проглянути, змінити значення, а потім зберегти дані в базі даних.

При використанні проекцій за допомогою служб Object Services дані, які повертаються не будуть оновлюваним об'єктом. Оскільки проекції повертають конкретні властивості, а не суттевості цілком. Служби Object Services не можуть зберегти зміни в спроектованих даних назад в базу даних. Якщо необхідно відновити дані, кращим варіантом повертатиме суттевість цілком і не використовувати проекцію [4].

За допомогою служб Object Services можна виконувати динамічні запити, написані на Entity SQL, для взаємодії з суттю EDM. Але в Entity Framework також можна працювати з класами EDM, що строго типізуються, використовуючи LINQ to Entities.

Висновки

Можна виділити основні достоїнства і недоліки технологій ADO.NET і LINQ.

Головні достоїнства ADO.NET:

дозволяє працювати з різними джерелами даних, розробник застосування може і не знати, яке СУБД буде у бази даних, з яким працюватиме його застосування, йому досить буде поміняти постачальника даних;

наявність автономних об'єктів дозволяють підвищити продуктивність і понизити навантаження на СУБД.

Головні недоліки ADO.NET:

обмежені можливості для роботи із запитами одного з найбільш широко використовуваного компоненту - DataSet.

Головні достоїнства LINQ:

за певних умов дозволяє провести глибоку інтеграцію бази даних і застосування;

значно прискорює процес написання запитів до бази даних, за рахунок, розширення синтаксису мов C# і Visual Basic;

надає компоненти для зручної роботи не тільки з базами даних, але і об'єктами, XML-документами і т.д.

Головні недоліки LINQ:

багато функціональних можливостей, які ограничуються тільки СУБД SQL Server.

Обидві технології відмінно поєднуються, і таким чином практично закривають недоліки один одного. Таким чином, можно зробити висновок, що у разі, коли застосування працюватиме тільки з СУБД SQL Server і надалі не планується мігрувати на іншу СУБД, то доцільно буде використовувати LINQ to SQL, з повним mapping бази даних. Це при не великих тимчасових витратах дозволить використовувати не реляційну, а об'єктну модель даних, яка знайома багатьом програмістам. А так само, якщо застосування розробляється в середовищі Microsoft Visual Studio, то ще і скористатися повною мірою перевагами написання коду з технологією IntelliSense, яка дописує назву функції при введенні початкових букв, використовується для доступу до документації і для усунення неоднозначності в іменах змінних, функцій і методів, використовуя рефлексію.

У сукупності це дозволить значно прискорити процес написання застосування .

Якщо розробнику не відомо наперед, яка СУБД використовуватиметься або надалі можлива зміна

СУБД, то кращим варіантом буде використання технології ADO.NET для доступу до джерела даних, і зберігання даних в DataSet. А маніпуляції з даними проводити за допомогою технології LINQ to DataSet. Однак дати відповідь однозначно, звичайно ж, ніхто не зважиться. Все залежить від предметної області, а також конкретних обставин і вимог до застосувань, що розробляються.

Краще використовувати Linq To SQL, якщо необхідно виконати такі вимоги:

написати ORM застосування з «маппінгом» об'єктів бази і об'єктів бізнесу 1 до 1;

написати ORM застосування з ієрархічним спадкоємством (inheritance hierarchies), яке зберігається в одній таблиці;

використовувати свої класи, а не що генеруються, або успадковуватися від базового классу;

використовувати LINQ для своїх запитів;

використовувати ORM, але також необхідно, щоб все це було високопродуктивним, і була можливість оптимізувати продуктивність методом редагування процедур, що зберігаються, і компільованих запитів.

Краще використовувати Linq To Entities, якщо необхідно виконати такі вимоги:

написати застосування, яке працюватиме з різними базами даних;

визначити моделі для застосування для конкретної предметної області;

написати ORM-застосування, яке матиме «маппінг» бази даних і об'єктів бізнесу 1 до 1 чи структуру, що відрізняється;

використовувати ORM-проект з ієрархічним спадкоємством, яке може мати альтернативні джерела зберігання схем (окрема таблиця для ієрархії, окрема таблиця для кожного класу, окрема таблиця для всіх даних, які відносяться до специфічного типа) використовувати ORM, але також необхідно, щоб все це було високопродуктивним, і була можливість оптимізувати продуктивність методом редагування процедур, що зберігаються, і компільованих запитів.

Як бачимо, в простих ситуаціях краще використовувати LINQ to SQL, в складніших - LINQ to Entities.

Список літератури

1. Запросы в LINQ to Entities [Електронний ресурс]. – Режим доступу: – http://msdn.microsoft.com/ ru-ru/library/bb896340.aspx.

2. Генератор модели EDM [Електронний ресурс]. – Режим доступу: –http://msdn.microsoft.com/ru-ru/library/ bb397926.aspx.

3. Здебский Р.Н. Технический обзор LINQ [Електронний ресурс]. – Режим доступу: – http://www.techdays. ru/videos/1163.html.

4. Жеребцов А.Н. LINQ и ADO.NET: различные подходы к организации доступа к данным [Електронний ресурс]. – Режим доступу: – http://www.yavnauke.ru/ users/jerebcov_a_n.

Надійшла до редколегії 19.03.2012

Рецензент: д-р техн. наук, доц. К.О. Метєшкін, Харківська національна академія міського господарства, Харків.

ОСОБЕННОСТИ ДОСТУПА К ДАННЫМ SQL SERVER С ПОМОЩЬЮ ADO.NET ENTITY FRAMEWORK

М.Ю. Лосев, Ю.И. Скорин

В работе рассматриваются особенности использования технологии ADO.NET Entity Framework (EF) для обеспечения построения приложений доступа к данным. При этом определяются основные компоненты платформы Entity Framework, которые реализуют принцип проектирования в моделировании данных, что заключается в разделении модели данных на концептуальную модель, логическую модель и физическую модель.

Ключевые слова: *приложение, реляционная схема, концептуальная модель, среда программирования, компонент, семантика, ограничение..*

FEATURES OF THE DATA ACCESS SQL SERVER USING ADO.NET ENTITY FRAMEWORK

M.Y. Losev, Y.I. Skorin

Particularly using of technology ADO.NET Entity Framework (EF) is considered in the paper. The major components of the platform Entity Framework are identified. These major components implement the principle of design in the simulation data. It is to separate the data model to the conceptual model, logical model and physical model.

Keywords: *application, relational schema, a conceptual model, a programming environment, a component of the semantics of the restriction.*