Информационные технологии

Многокритериальный метод оценки производительности ORM-решений в различных информационных системах

Н. В. Крапухина, П. А. Курников, И. А. Тарханов

Аннотация. В статье предложен метод сравнения и оценки производительности библиотек объектно-реляционного отображения (object-relational mapping, ORM) в контексте применения их при разработке различных информационных систем (ИС). В работе предлагается способ формирования критериев при оценке ORM, которые определяются на основе потребляемых ресурсов памяти и времени выполнения множества функциональных тестов. Предложено использовать метод многокритериальной оптимизации Электра для выбора наилучшего по производительности варианта.

Ключевые слова: *ORM, производительность, метод Электра, MVC, временной ряд, временной ресурс, ресурс памяти.*

1. Проблема сравнения и оценки производительности ORM

С точки зрения разработчика, ИС должна представлять хранилище объектов, что позволяет создавать объекты и более эффективно работать с ними, а они автоматически будут сохраняться в реляционной базе данных. На практике всё не так просто и очевидно. Все ORM компоненты обычно сводят к минимуму действия, характерные для конкретной СУБД. Как следствие, код работы с транзакциями может быть медленным и неэффективным, особенно в терминах сгенерированного SQL. Все это может привести к тому, что программы будут работать медленнее и использовать больше памяти, чем программы, написанные «вручную». Но ORM избавляет программиста от написания большого количества кода, часто однообразного и подверженного ошибкам, тем самым значительно повышая скорость разработки. Кроме того, большинство современных реализаций ORM позволяют программисту при необходимости самому жёстко задать код SQL-запросов, который будет использоваться при тех или иных действиях (сохранение в базу данных, загрузка, поиск и т. д.) с постоянным объектом.

Многие компании, разработчики ПО с целью совершенствования своих продуктов проводят исследования для выбора оптимальной ORM по многим критериям. Как правило, такие исследования являются закрытыми и заканчиваются созданием нескольких прикладных тестов, решающих текущие прикладные задачи. Результаты таких тестов тяжело сравнивать друг с другом. В рамках этой статьи предлагается комплексный подход к методу оценки производительности ORM компонент. В качестве сравниваемых ORM компонент были выбраны наиболее популярные: Entity Framework [1, 2], LINQ to SQL [3], NHibernate [4, 5, 6].

2. Критерии оценки производительности ORM

Особенностью данной работы является то что в качестве критериев оценки ORM компонент предложено использовать оценочные индексы: $\boldsymbol{k_t}$ (оценка временного ресурса), $\boldsymbol{k_r}$ (оценка ресурса памяти) которые являются компонентами векторного критерия оценки производительности. Выбор данных компонентов обусловлен тем, что к ORM-компоненту должны предъявляться повышенные требования по быстродействию, минимизации используемой оперативной памяти.

Для оценки значений критериев каждой ORM-компонент был выполнен набор тестов, моделирующий функционал разрабатываемой ИС. Результаты выполнения набора тестов представлены в виде многомерного интервального временного ряда. Первый уровень — время выполнения теста, второй — выделенная под тест память.

По данным анализа временного ряда вычислены оценочные показатели, по которым можно судить о производительности ORM-решения в проекции исследуемой системы.

Вычисление оценочных показателей производительности

Индекс временного ресурса вычисляется по формуле:

$$k_t = \sum_{r=1}^{n} \left[\frac{\left(\sum_{q=1}^{m} x_{qr} \right) * p_r}{m} \right].$$

Это аддитивная свёртка математических ожиданий времен выполнения (n) типов тестов.

n — количество типов тестов;

m — количество тестов одного типа;

x_{qr} — время выполнения теста;

p_r — весовой коэффициент, вычисляемый как отношение времени работы теста r к общему времени работы всех тестов.

Ресурс памяти $\boldsymbol{k_r}$ вычисляется как максимальное значение разностей состояний соединения до момента выполнения теста и после выполнения. Соединение хранит в себе необходимую для выполнения теста информацию. При выполнении теста, размер оперативной памяти, выделенной под соединение, меняется. Данный метод подсчёта обусловлен тем, что ORM-компоненты используемы при построении ИС, демонстрируют признаки нахождения в неравновесном состоянии, характеризующимся различными значениями параметров на входе, выходе и внутренних точках системы [7].

3. Задача многокритериальной оптимизации по методу Электра

Для найденных оценок производительности ORM предлагается применить метод векторной оптимизации ЭЛЕКТРА. Метод позволяет на основе попарного сравнения всех альтернатив выделить группу лучших альтернатив, множество которых поэтапно сокращается до приемлемого или заранее заданного количества для лица, принимающего решение (ЛПР). Множество альтернатив строится на основе последовательности бинарных отношений между любой парой альтернатив. Метод позволяет отсекать все неэффективные варианты [8].

Входная информация

$X = \{x_1, x_2, x_3\}$ — множество альтернатив ORM: Entity Framework, Linq to Sql, NHibernate,

$I = \{1, 2\}$ — множество номеров критериев $\boldsymbol{k_t}$, $\boldsymbol{k_r}$,

$z_1 = f_1(x_k)$, $z_2 = f_2(x_k)$, $k = 1,..., n$ — оценки альтернатив по критериям,

α_1, α_2 — веса критериев.

Вектор приоритета (весовой вектор) α_i для каждого из критериев строится экспертом в области ИС.

Таблица 1

Таблица отношения критериев и альтернатив

Критерии / Альтернативы	$\boldsymbol{k_t}$ (z_1)	$\boldsymbol{k_r}$ (z_2)
Entity Framework (x_1)	$z_1 = f_1(x_1)$	$z_2 = f_2(x_1)$
Linq to Sql (x_2)	$z_1 = f_1(x_2)$	$z_2 = f_2(x_2)$
Nhibernate (x_3)	$z_1 = f_1(x_3)$	$z_2 = f_2(x_3)$

Требуется выделить группу лучших альтернатив.

Этапы решения поставленной задачи

На первом этапе проводится полное попарное сравнение всех альтернатив. Для каждой пары альтернатив по критериальным оценкам вычисляются значения двух специальных индексов: согласия и несогласия. Выдвигается гипотеза о превосходстве одной альтернативы над другой. Множество номеров критериев $I = \{1,...,2\}$ разбивается на девять подмножеств (3 подмножества для каждого критерия):

$I^+(x_1, x_2) = \{i: f_i(x_1) \succ f_i(x_2); i = 1,..., 2\}$ — подмножество критериев, по которым альтернатива (x_1) предпочтительнее (x_2);

$I^{\propto}(x_1,x_2)=\{i\colon f_i(x_1)\propto f_i(x_2);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_1) эквивалентна (x_2);

$I^{-}(x_1,x_2)=\{i\colon f_i(x_1)\prec f_i(x_2);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_2) предпочтительнее (x_1);

$I^{+}(x_1,x_3)=\{i\colon f_i(x_1)\succ f_i(x_3);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_1) предпочтительнее (x_3);

$I^{\propto}(x_1,x_3)=\{i\colon f_i(x_1)\propto f_i(x_3);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_1) эквивалентна (x_3);

$I^{-}(x_1,x_3)=\{i\colon f_i(x_1)\prec f_i(x_3);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_3) предпочтительнее (x_1);

$I^{+}(x_2,x_3)=\{i\colon f_i(x_2)\succ f_i(x_3);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_2) предпочтительнее (x_3);

$I^{\propto}(x_2,x_3)=\{i\colon f_i(x_2)\propto f_i(x_3);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_2) эквивалентна (x_3);

$I^{-}(x_2,x_3)=\{i\colon f_i(x_2)\prec f_i(x_3);i=1,...,2\}$ — подмножество критериев, по которым альтернатива (x_3) предпочтительнее (x_2).

На втором этапе вводятся $C_{x_nx_m}$ — индекс согласия с гипотезой о превосходстве x_n над x_m и $d_{x_nx_m}$ — индекс несогласия с гипотезой о превосходстве x_n над x_m для $n, m = 1,\ldots, 3; n \neq m$;

Индекс согласия подсчитывается на основе весов критериев как отношение суммы весов критериев подмножеств $I^{+}(x_n,x_m)$ и $I^{\propto}(x_n,x_m)$ к общей сумме весов.

Индекс несогласия определяется на основе учета относительных величин проигрышей альтернативы x_n альтернативе x_m. Для каждого критерия z_i из подмножества $i\in I^{-}(x_n,x_m)$ вычисляются разности значений критерия для альтернатив x_n, x_m. Полученная величина делится на длину шкалы этого критерия (L_i), затем за значение индекса несогласия принимается наибольшая относительная величина:

$$C_{x_1x_2}=\frac{\sum\limits_{i\in I^{+}(x_1,x_2),I^{\propto}(x_1,x_2)}\alpha_i}{\sum\limits_{i=1}^{2}\alpha_i},$$

$$d_{x_1x_2}=\max_{i\in I^{-}(x_1,x_2)}\frac{\left|f_i(x_1)-f_i(x_2)\right|}{L_i};i=1,...,2,$$

$$C_{x_1x_3}=\frac{\sum\limits_{i\in I^{+}(x_1,x_3),I^{\propto}(x_1,x_3)}\alpha_i}{\sum\limits_{i=1}^{2}\alpha_i},$$

$$d_{x_1x_3}=\max_{i\in I^{-}(x_1,x_3)}\frac{\left|f_i(x_1)-f_i(x_3)\right|}{L_i};i=1,...,2,$$

$$C_{x_2x_3}=\frac{\sum\limits_{i\in I^{+}(x_2,x_3),I^{\propto}(x_2,x_3)}\alpha_i}{\sum\limits_{i=1}^{2}\alpha_i}$$

$$d_{x_2x_3}=\max_{i\in I^{-}(x_2,x_3)}\frac{\left|f_i(x_2)-f_i(x_3)\right|}{L_i};i=1,...,2.$$

Далее задаются пороговые значения: уровни согласия C_1 ($0\le C_1\le 1$) и несогласия d_1 ($0\le d_1\le 1$), с которыми сравниваются значения вычисленных индексов для каждой пары альтернатив. Если $C_{x_nx_m}\ge C_1$ и $d_{x_nx_m}\le d_1$, то альтернатива x_n объявляется лучшей по сравнению с альтернативой x_m, т. е. альтернатива x_m — доминируемая. В противном случае альтернативы несравнимы. Из множества альтернатив удаляются доминируемые. Оставшиеся альтернативы образуют ядро, в которое входят доминирующие и несравнимые альтернативы.

Следующим этапом вводятся более «слабые» пороговые значения: уровни согласия и несогласия, удовлетворяющие условиям $C_{r+1}\le C_r$, $d_{r+1}\ge d_r$, $r=2,3,...$, при которых выделяются ядра с меньшим количеством альтернатив и процедура сравнения значений полученных индексов вновь повторяется до тех пор, пока число альтернатив в ядре становится приемлемым для ЛПР.

Ядра упорядочены по признаку улучшения качества (производительности) альтернатив (ORM). В последнее ядро входит наилучшая альтернатива.

4. Результаты применения метода

В рамках совместного проекта кафедры Инженерной Кибернетики НИТУ МИСиС и компании Cognitive Technologies была разработана тестовая среда для проведения измерений. В программе реализован следующий функционал:

1) создание профиля с некоторым набором и распределением тестов, имитирующих работу систем и последующим их запуском;
2) сериализация результатов выполнения набора тестов в виде интервального многомерного временного ряда, пригодным для дальнейшего анализа.

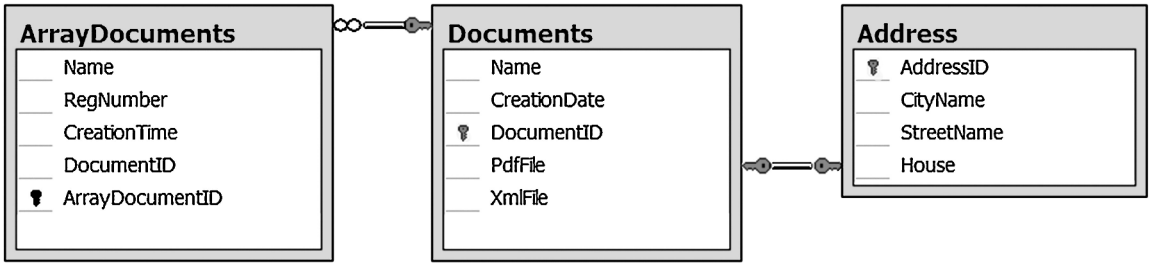

Рис. 1. Диаграмма реляционной базы данных для тестов

В качестве эксперимента для апробации метода были выбраны 3 ORM-системы: Entity Framework, LINQ to SQL, NHibernate, как наиболее распространённые. Разработаны базовые тесты различного функционала с возможностью вариационной компоновки набора тестов, имитирующего работу различных ИС [9]. Осуществлена программная реализация 6 базовых тестов для 3-х ORM.

Список тестов был разработан с учётом наиболее часто встречающихся операций работы над СУБД с точки зрения ИС.

1. SearchSimple — поиск по таблице документов (без ассоциированных с ним адреса и массива, и без файлов) по дате.
2. GetFile — получение PDF и XML файлов из таблицы документов.
3. PutFile — запись PDF и XML файлов в таблицу документов.
4. GetStructObject — получение документа и связанного с ним по ключу адреса из другой таблицы.
5. PutStructObject — запись структурированного объекта в несколько таблиц базы: документа, адреса, связанных строк массива документа.
6. SearchStructObject — поиск документов по связанной таблице по дате.

Описанная программа реализована на языке C# с применением технологии ASP MVC 3 [10]. Благодаря архитектуре шаблона MVC, а также использование одной и той же схемы реляционной СУБД во всех тестовых прогонах, программа позволяет получить результаты тестов разных ORM, которые можно сравнивать друг с другом.

Таблица 2

Распределение и количество выполненных тестов для профиля документооборота

Тип теста	Распределение	Количество выполненных тестов		
		Entity Framework	LINQ to SQL	NHibernate
GetStructObject	30	10800	14400	10800
PutStructObject	40	14400	19200	14400
SearchStructObject	25	9000	12000	9000
SearchSimple	5	1800	2400	1800

Экспертом был создан профиль тестирования, соответствующий системам документооборота. Было выполнено 5 тестовых прогонов для 3-х ORM. Время каждого прогона составило 300 минут.

Временные ряды были проанализированы описанным выше методом, и полученные результаты приведены в табл. 3. Значение индекса k_r выражается в байтах, но при подсчете результатов значения переведены в мегабайты.

Таблица 3

Результаты индексов производительности отдельных тестов для профиля документооборота

Вид теста	Entity Framework		LINQ to SQL		NHibernate	
	k_t	k_r	k_t	k_r	k_t	k_r
GetStructObject	1,72	2,88	0,71	2,42	3,44	2,36
PutStructObject	187,24	4,97	212,76	2,37	219,33	2,38
SearchStruct Object	121,11	30,42	115,95	32,64	252,65	58,46
SearchSimple	0,41	2,33	36,92	30,31	0,04	0,02

Таблица 4

Результаты индексов производительности в рамках профиля тестирования документооборота

Вид ORM	Индекс ресурса времени (меньше — лучше)	Индекс ресурса памяти (меньше — лучше)
Entity Framework	310,4883	SearchStructObject 30,42
LINQ to SQL	366,3288	SearchStructObject 32,64
NHibernate	475,455	SearchStructObject 58,46

Так же была создана еще одна конфигурация ИС (табл. 5) и протестирована на тех же ORM (табл. 6).

Табл. 6 — Методом ЭЛЕКТРА установлено, что самой производительной альтернативной из тестируемых ORM платформ в данной проекции ИС является Entity Framework. На данном этапе исследования можно сказать, что эффективность ORM-компонент зависит от заданного профиля ИС. Примером являются тесты SearchSimple, GetStructObject для NHibernate,

Таблица 5

Распределение и количество выполненных тестов

Тип теста	Распределение	Количество выполненных тестов		
		Entity Framework	LINQ to SQL	NHibernate
GetStructObject	30	210	60	1590
SearchSimple	70	490	140	3710

Таблица 6

Результаты индексов производительности в рамках профиля тестирования

Вид ORM	Индекс ресурса времени (меньше — лучше)	Индекс ресурса памяти (меньше — лучше)
Entity Framework	7,77	SearchStructObject 2,91
LINQ to SQL	479,20	SearchStructObject 60,65
NHibernate	4,83	SearchStructObject 2,37

показывающей лучшую производительность (табл. 3). Однако в рамках данной проекции ИС (табл. 2) NHibernate оказалась наименее производительной из тестируемых ORM. При другом распределении этих тестов в профиле тестирования результаты были бы другими. В частности, увеличение количества тестов SearchSimple, GetStructObject в профиле ведет к улучшению показателей производительности у NHibernate (табл. 5, 6).

Заключение

Реализованный программный продукт позволяет внести экспертом ИС ряд тестов, имитирующих ИС, и настроить профиль тестируемой системы. Результаты запуска набора базовых тестов представлены в виде временных рядов. Для каждого ORM-решения структура динамики временных рядов различна. В данной работе введены оценочные индексы ORM. Рассмотрены методы оценки значений критериев каждой ORM альтернативы. Для найденных оценок применен метод векторной оптимизации ЭЛЕКТРА, позволяющий сузить множество альтернатив до числа, приемлемого для ЛПР. Создана среда тестирования, позволяющая собирать данные о работе ORM в проекции ИС. Архитектура реализованного приложения на основе MVC (Model-view-controller) модели позволяет подключать новые ORM модули для тестирования без изменения архитектуры, а также сравнивать результаты тестирования.

Был проведён эксперимент для распределения тестов, соответствующего системе электронного документооборота. В результате оценки 3 различных ORM самой производительной была выбрана Entity Framework.

Литература

1. Платформа ADO.NET Entity Framework // MSDN Library. Официальный ресурс продуктов Microsoft. http://msdn.microsoft.com/ru-ru/library/bb399572(v=vs. 110).aspx
2. Обзор архитектуры Entity Framework // Web ресурс Professor Web. http://professorweb.ru/my/ADO_NET/base/level3/3_1.php
3. Платформа Linq to Sql // MSDN Library. Официальный ресурс продуктов Microsoft. http://msdn.microsoft.com/ru-ru/library/vstudio/ bb386976(v=vs.100).aspx
4. Новые возможности в NHibernate 3.2 // Web блог weblogs.asp.net.
5. NHibernate 3.2 // Блоги сообщества ASP.NET. http://weblogs.asp.net/ricardoperes/ archive/2011/08/08/nhibernate-3-2-released.aspx
6. Создание настольного приложения с применением NHibernate. Орен Эйни (Oren Eini) // Журнал MSDN Magazine. http://msdn.microsoft.com/ru-ru/magazine/ee819139.aspx
7. *Мельников С. В.* Образование диссипативных структур в ORM-компонентах высоконагруженных порталов // Современные проблемы науки и образования. 2013. № 6 (приложение «Технические науки»). С. 5.
8. *Ларичев О. И.* Теория и методы принятия решений, а также Хроника событий в Волшебных Странах: Учебник. М.: Логос, 2000. 296 с.
9. *Бек К.* Экстремальное программирование: разработка через тестирование. Библиотека программиста. СПб.: Питер, 2003. 224 с.
10. *Фримен А.*, *Сандерсон С.* ASP.NET MVC 3 FRAMEWORK с примерами на C# для профессионалов / пер. с англ. Ю. И. Корниенко, А. А. Моргунова. М., 2012.

Крапухина Нина Владимировна. Профессор НИТУ МИСиС. К. т. н. Окончила 1971 г. МАИ. Количество печатных работ: 110. Область научных интересов: системный анализ, принятие решений, математическое моделирование, методы искусственного интеллекта. E-mail: krapuhina@misis.ru

Курников Павел Александрович. Студент НИТУ МИСиС. Количество печатных работ: 1. Область научных интересов: Системный анализ, документооборот, системное программирование. E-mail: two-names@yandex.ru

Тарханов Иван Александрович. С. н. с. ИСА РАН. К. т. н.. Окончил в 2005 г. МФТИ(ГУ). Количество печатных работ: 14. Область научных интересов: системы управления базами данных, электронный документооборот, архивы, информационная безопасность, системный анализ. E-mail: ivant@cs.isa.ru