Выбор ORM в информационных системах на основе моделирования пользовательской нагрузки и многокритериального анализа

П.А. Курников

АО «Когнитив», г. Москва, Россия

Аннотация. В работе рассматривается проблема выбора эффективного инструмента объектно-реляционного отображения (ORM) в условиях функционирования реальных информационных систем. Анализ существующих исследований показал, что большинство из них основаны на микротестах CRUD-операций и не учитывают комплексное поведение системы, что приводит к противоречивым и трудно интерпретируемым результатам. В статье предлагается подход к оценке производительности ORM, основанный на дискретно-событийном моделировании пользовательской активности и многокритериальном анализе. Разработана модель, описывающая поведение множества пользователей в виде потоков событий с вероятностным распределением, а также формализована система показателей производительности, включающая индексы времени выполнения, потребления памяти и загрузки процессора. Проведены вычислительные эксперименты для различных профилей нагрузки, имитирующих реальные сценарии работы информационных систем. Полученные результаты показывают, что производительность ORM зависит от характера нагрузки и сценариев эксплуатации информационных систем. Предложенный подход позволяет повысить обоснованность выбора ORM-инструментов и может быть использован при проектировании высоконагруженных информационных систем.

Ключевые слова: *информационная система, производительность, ORM, база данных, кэширование, многокритериальная оптимизация, дискретно-событийное моделирование.*

DOI: 10.14357/20790279260208 **EDN:** WNJVIZ

Введение

Современные информационные системы (ИС) широко используют технологии объектно-реляционного отображения (ORM) для взаимодействия с реляционными базами данных [1]. Данный подход позволяет сократить разрыв между объектной и реляционной моделями данных, упростить разработку и повысить уровень абстракции. Вместе с тем использование ORM сопровождается дополнительными накладными расходами, связанными с генерацией запросов, управлением состоянием объектов и механизмами кэширования, что влияет на производительность системы [2-3]. Существующие исследования в области оценки ORM-фреймворков в основном сосредоточены на сравнении отдельных операций (CRUD) в изолированной среде [4]. Такие подходы не учитывают поведение системы в условиях реальной эксплуатации, где нагрузка формируется множеством пользователей, выполняющих различные сценарии взаимодействия с системой. В результате получаемые оценки носят фрагментарный характер и часто приводят к противоречивым выводам.

Существенной особенностью функционирования ORM в ИС является их динамический и нелинейных характер. Так, например, в работе Е. Андриановой, С. Мельникова 2015 года показано, что информационные системы могут проявлять свойства сложных динамических сред, в которых возникают эффекты, аналогичные диссипативным процессам в физических системах. Авторы отмечают возможность формирования нестационарных режимов, колебательных процессов, блокирующих структур и других эффектов самоорганизации в потоках запросов к базам данных [5]. Данные выводы указывают на необходимость рассмотрения ИС как открытых нелинейных систем с изменяющимся внутренним состоянием. В работах [6-7]

показано, что кэш-подсистемы могут рассматриваться как нелинейные открытые системы, функционирующие в условиях постоянного взаимодействия с внешней средой. Дополнительно, в работе П. Курникова, Н. Крапухиной 2019 года выявлена статистическая связь между режимами функционирования системы и временем обработки пользовательских запросов, что указывает на зависимость производительности от текущего состояния системы. Пользовательская активность формирует поток запросов, который изменяет внутреннее состояние системы, вызывая процессы накопления и высвобождения кэша, а также формируя положительные и отрицательные обратные связи [6]. В работе [7] показано, что ключевой особенностью современных ИС является их динамическая природа: поток пользовательских запросов изменяется во времени, а внутренняя структура системы (в частности, кэш) эволюционирует под воздействием этой нагрузки. 1 показано, что кэширующие механизмы ИС демонстрируют нелинейную динамику, включая режимы самоорганизации, би-стабильности и предельных циклов. Следовательно, система представляет собой нелинейную открытую динамическую среду с обратными связями, в которой производительность определяется не только отдельными операциями, но и их совокупным взаимодействием.

Таким образом, поведение ORM и внутренних процессов кэширования определяется не отдельными запросами, а совокупностью взаимодействующих процессов во времени. Это делает классические подходы, основанные на бенчмаркинге и анализе отдельных операций, недостаточными. В этих условиях дискретно-событийное моделирование представляется наиболее адекватным инструментом, так как позволяет учитывать поток пользовательских событий, изменение состояния системы и многопользовательскую нагрузку в рамках единой модели ИС. Для адекватного анализа таких систем необходим подход, учитывающий:

– временную структуру нагрузки,
– многопользовательский режим работы,
– изменение внутреннего состояния системы,
– стохастический характер пользовательской активности.

В данной работе предлагается использовать дискретно-событийное моделирование, позволяющее представить функционирование ИС как последовательность событий, генерируемых пользователями и приводящих к изменениям состояния системы. Такой подход позволяет сформировать реалистичные профили нагрузки и исследовать поведение ORM-фреймворков в условиях, приближенных к реальной эксплуатации.

1. Обзор существующих исследований

Несмотря на широкий спектр ORM-фреймворков, представленных в различных технологических стеках, в данной работе внимание сосредоточено на инструментах экосистемы .NET (Entity Framework, NHibernate, Dapper, LINQ to SQL). Такой выбор обусловлен их широкой распространенностью в корпоративных информационных системах и практической значимостью при разработке высоконагруженных приложений. Кроме того, ограничение области исследования одной платформой позволяет обеспечить сопоставимость экспериментальных условий и корректность сравнительного анализа, исключая влияние различий на уровне среды выполнения, драйверов доступа к данным и архитектурных особенностей платформ. При этом следует отметить, что предлагаемые автором в данной работе модели, методы моделирования нагрузки и подходы к оценке эффективности не ограничиваются исключительно .NET-технологиями и могут быть обобщены на более широкий класс ORM-фреймворков независимо от используемой платформы.

1.1. Общие подходы к анализу ORM

В работе K. Hule, R. Ranawat 2023 года «Analysis of Different ORM Tools for Data Access Object Tier Generation: A Brief Study» [4] авторы провели фундаментальный обзор большого количества работ по ORM, написанные в период с 1999 года по 2023. Данный обзор охватывает 30 статей, 8 ORM инструментов и 4 языка программирования. В работе рассмотрены как .NET-семейство (EF, NHibernate, Dapper), так и Java/JPA/Hibernate. Исследования в основном сосредоточены на сравнительном анализе производительности CRUD-операций, сложности генерации SQL-запросов, разработке оберток или библиотек для оптимизации доступа к данным, а также на проблеме «импедансного несоответствия» между объектными и реляционными моделями. Данный обзор использовался в настоящей работе в качестве базового источника для формирования корпуса релевантных исследований, а также для идентификации ключевых направлений анализа ORM. На основе анализа [4] можно выделить основные исследовательские подходы:

– сравнительный анализ производительности CRUD-операций;
– оценка качества генерации SQL-запросов;

– разработка вспомогательных библиотек и оптимизационных обёрток;
– изучение проблемы импедансного несоответствия.

При этом обзор показывает, что подавляющее большинство работ рассматривает ORM-инструменты в изоляции, без учета поведения системы в целом и без моделирования реальной нагрузки. Это послужило отправной точкой для постановки задачи настоящего исследования.

1.2. Экспериментальные исследования производительности ORM

В работе [8] проведен экспериментальный бенчмарк трех ORM-фреймворков (.NET): Entity Framework Core, NHibernate и Dapper. Использована единая тестовая архитектура (консольное приложение + DotNetBenchmark), фиксированная база данных и одинаковые CRUD-операции различной сложности (с учетом связей one-to-one и one-to-many). Для оценки результатов использовались метрики времени выполнения запросов (execution time) и потребление памяти (memory usage) при разном объеме данных. К ограничениям данной работы можно отнести тестирование в локальной среде без распределнных, реальных нагрузок, ограниченный набор сценариев (CRUD), а также единственная конфигурация БД и аппаратной среды. Однако автором работы показано, что эффективность ORM зависит от типа выполняемых операций. Так например, NHibernate показал лучшие результаты для Insert и сложных выборок, EF Core эффективен для операций с простыми связями, Dapper эффективен в ряде Update/Delete и простых выборках. Следовательно, выбор ORM должен определяться профилем нагрузки приложения. В работе [9] проводится сравнительный анализ двух ORM-фреймворков платформы .NET – Entity Framework Core и NHibernate с фокусом на операциях DML (Create, Update, Delete). Методология основана на экспериментальном подходе: реализовано настольное приложение, выполняющее набор тестовых сценариев для различных типов связей (1:1, 1:N, M:N) и объемов данных (500 и 5000 записей), как в одиночных, так и массовых операциях. В качестве метрик используются среднее время выполнения операций и показатели вариативности (стандартное отклонение, коэффициент вариации), что позволяет оценить не только производительность, но и стабильность. Результаты показывают, что NHibernate в большинстве сценариев демонстрирует более высокую производительность, особенно в одиночных операциях, тогда как в массовых тестах различия между фреймворками значительно сокращаются; уровень стабильности решений оказывается сопоставимым. При этом исследование имеет ряд ограничений: тестирование проводится в изолированной среде с фиксированными сценариями CRUD/DML, используется упрощенная модель базы данных и единичные прогоны тестов, отсутствует моделирование реальной нагрузки, многопользовательского доступа и системных эффектов (например, кэширования или конкуренции за ресурсы). В результате работа не рассматривает поведение ORM как части целостной информационной системы, что ограничивает применимость полученных выводов к реальным эксплуатационным условиям. Работа [10] представляет собой сравнительный бенчмарк ORM-инструментов в среде .NET 6 (Dapper, EF Core, NHibernate), где методология сводится к выполнению изолированных CRUD, search и sort операций на фиксированном наборе данных (Chinook DB) с измерением трех метрик: времени выполнения, потребления RAM и загрузки CPU. Несмотря на аккуратную инструментализацию (BenchmarkDotNet, многократные прогоны, попытки нормализации CPU), экспериментальная постановка носит микроуровневый характер: операции выполняются в изоляции, без моделирования пользовательской нагрузки, конкурентного доступа, сетевых задержек и транзакционных сценариев. В результате отсутствует системное представление нагрузки как целостной модели – не учитываются профили пользователей, смешанные типы операций, очереди запросов и деградация под нагрузкой. Это ограничивает применимость результатов, так как полученные метрики отражают синтетические кейсы, а не поведение ORM в реальных высоконагруженных системах. Дополнительно, использование одного датасета и одной конфигурации среды без вариативности параметров усиливает проблему обобщаемости и делает выводы скорее частными, чем универсальными. В работе [11] авторами предложено сравнительное исследование ORM-подходов (Entity Framework, LINQ to SQL и Transact Query) на основе выполнения 11 типов запросов. Методология сводится к локальному эксперименту с замерами времени выполнения (CPU, мс) и потребления памяти (KB) через брейкпоинты, а также к проверке устойчивости к SQL-инъекциям на упрощенной форме логина. В качестве метрик используются исключительно низкоуровневые показатели (CPU и память) и бинарная устойчивость к атакам. Однако исследование носит несистемный характер: нагрузка моделируется на уровне отдельных запросов, без учета

пользовательских сценариев, конкуренции, сетевых задержек и реальной многопользовательской нагрузки. Отсутствует целостная модель системы и профилирование поведения под нагрузкой (workload), что существенно ограничивает применимость результатов. Кроме того, малая выборка прогонов (2–3 раза), игнорирование «холодного старта» и упрощенная модель безопасности снижают достоверность выводов и не позволяют экстраполировать их на реальные production-системы. Работа D. Colley, C. Stanier 2018 года носит преимущественно качественно-демонстрационный характер: методология основана на разборе отдельных кейсов и сравнении ORM-сгенерированных и ручных SQL-запросов в рамках одного приложения (Contoso), без систематического экспериментального дизайна. В работе фактически анализируется один ORM – Entity Framework (на примере приложения Contoso University). Сравнение проводится не между разными ORM, а между ORM-подходом и «ручными» SQL-запросами (non-ORM). В качестве метрик используются точечные показатели (время выполнения, logical/physical reads, время компиляции, размер execution plan), однако отсутствует масштабируемый бенчмарк и статистическая обработка результатов. Ключевое ограничение –полное отсутствие системного моделирования нагрузки: анализ проводится на уровне отдельных запросов и сценариев без учета пользовательской активности, конкурентного доступа, сетевых задержек и реального workload, что не позволяет рассматривать систему как целостную динамическую модель. Также выборка экспериментов минимальна и не репрезентативна, а выводы носят иллюстративный характер, что существенно ограничивает их переносимость на production-системы [12].

1.3. Ограничения существующих подходов

Результаты представленных работ не согласованы между собой в глобальном смысле. Даже при использовании одних и тех же критериев (время выполнения запросов, потребление памяти, CPU) итоговые выводы о наиболее производительном ORM различались: в одних экспериментах быстрее оказывался Entity Framework, в других – NHibernate, LINQToSQL, Dapper или даже SubSonic. Так, в работе [8] предпочтение отдается NHibernate и частично Dapper в зависимости от типа операций; в [9] NHibernate демонстрирует преимущество над Entity Framework; в [11] делается вывод о высокой эффективности Dapper и EF Core; в работах [11-12] результаты носят частный характер и не позволяют выделить устойчивого лидера. Для одного и того же множества ORM-решений отсутствует согласованное ранжирование альтернатив. Более того, ни в одной из работ не рассматривается устойчивость полученных решений при изменении профиля нагрузки. Это приводит к ситуации, когда выводы различных авторов оказываются несопоставимыми и зависят от частных экспериментальных условий. Такая противоречивость объясняется тем, что авторы использовали собственные начальные условия, различные базы данных и различный набор CRUD-тестов, который способен частично покрыть лишь один сценарий функционирования информационной системы. В итоге исследования ограничивались микротестами и не рассматривали нагрузку как целостную системную модель. В них отсутствовал подход к робастности выбора ORM-инструментов – будь то многокритериальная оптимизация или ранжирование альтернатив с учетом разных факторов эксплуатации. Таким образом, существующие работы фокусируются на частных экспериментах и не дают целостного представления о поведении ORM-систем в реальных условиях. Данное противоречие послужило мотивацией для разработки подхода, основанного на моделировании пользовательской нагрузки как целостной системы и применении методов многокритериального анализа, что позволяет исследовать устойчивость выбора ORM при варьировании профиля ИС.

2. Методология исследования

Несмотря на широкий спектр ORM-фреймворков, представленных в различных технологических стеках, в данной работе внимание сосредоточено на инструментах экосистемы .NET (Entity Framework, NHibernate, Dapper, LINQ to SQL). Такой выбор обусловлен их широкой распространенностью в корпоративных информационных системах и практической значимостью при разработке высоконагруженных приложений. Кроме того, ограничение области исследования одной платформой позволяет обеспечить сопоставимость экспериментальных условий и корректность сравнительного анализа, исключая влияние различий на уровне среды выполнения, драйверов доступа к данным и архитектурных особенностей платформ. При этом следует отметить, что предлагаемые автором в данной работе модели, методы моделирования нагрузки и подходы к оценке эффективности не ограничиваются исключительно .NET-технологиями и могут быть обобщены на более широкий класс ORM-фреймворков независимо от используемой платформы.

Табл. 1

Сравнительный анализ исследований производительности ORM-фреймворков .NET

Работа	ORM	Методология	Метрики	Ограничения
Performance comparison of crud methods using net object relational mappers: A case study	{Entity Framework, Dapper, NHibernate}	Проведено детальное исследование влияния определенного ORM на производительность приложений при выполнении запросов к БД. Для анализа использован специально разработанный механизм тестирования.	Метрики времени выполнения запросов (execution time) и потребление памяти (memory usage) при разном объеме данных	Проверялись только 3 ORM из .NET-семейства.
Comparative analysis of selected object-relational mapping systems for the. NET platform	{NHibernate, Entity Framework}	Сравнение производительности Entity Framework Core и NHibernate на платформе .NET для операций CRUD. NHibernate оказался намного быстрее в единичных тестах и незначительно быстрее в массовых.	Среднее время выполнения операций и показатели вариативности (стандартное отклонение, коэффициент вариации)	Фокус только на .NET. Тестирование проводилось на простой схеме БД. Не рассматривались более сложные запросы.
Performance analysis of object-relational mapping (orm) tools in. net 6 environment	{Dapper, Entity Framework, NHibernate}	Изучаются инструменты ORM – Entity Framework и NHibernate и проводится сравнение их языков запросов и результатов с SQLClient.	Время выполнения запроса, потребление RAM и загрузки CPU	Работа ограничена только использованием .NET Framework.
Systematic Performance and Security Evaluation of .NET Models for Accessing Database	{Entity Framework, LINQ to SQL, Transact Query}	Локальный эксперимент с замерами времени выполнения (CPU, мс) и потребления памяти (KB) над группой ORM: Entity Framework, LINQ to SQL и Transact Query	Время выполнения (CPU, мс) и потребления памяти (KB) как низкоуровневые показатели и бинарная устойчивость к атакам	Несистемный подход: тесты на уровне отдельных запросов без пользовательской нагрузки и целостной модели. Малая выборка (2–3 прогона).
The Impact of Object-Relational Mapping Frameworks on Relational Query Performance	{Entity Framework}	Сравнение ORM и ручного SQL на примере (Entity Framework), анализ execution plan и метрик (время, reads, компиляция) на отдельных запросах	Время выполнения, logical/physical reads, время компиляции, размер execution plan	Нет системного бенчмарка: анализ отдельных запросов без user workload, конкуренции и целостной модели; один ORM, малая выборка, результаты не обобщаемы

2.1 Описание дискретно-событийной модели

Для моделирования поведения многопользовательской информационной системы, взаимодействующей с базой данных через ORM, используется дискретно-событийный подход, в рамках которого система представляется как совокупность потоков пользовательских запросов и механизма их обработки. Определим множество пользователей:

$$U = \{u_1, u_2, ..., u_M\}, \quad (1)$$

Каждому пользователю $u_M \in U$ сопоставляется подмножество тестов (запросов к базе данных):

$$F_m \subseteq F, \quad (2)$$

где $F = \{f_1, ..., f_L\}$ – множество всех типов тестов (CRUD-операций и их комбинаций).

Под тестом понимается запрос к базе данных на извлечение или модификацию нескольких объектов. Причем тест должен быть реализован на стороне сущностной модели и выполняться на стороне сервера через ORM преобразование [14–15]. Выбор теста пользователем моделируется как дискретная случайна величина с распределением вероятностей:

$$P = \{p_1, ..., p_L\}, p_l = \mathrm{P}(f = f_l), \sum_{l=1}^{L} p_l = 1, \quad (3)$$

Модель потока событий**.** Процесс генерации запросов пользователем задаётся последовательностью моментов времени:

$$0 = t_0 < t_1 < ... < t_K = T, \quad (4)$$

где T – время моделирования.

Интервалы между событиями определяются как $\tau_k = t_k - t_{k-1}$. Для каждого пользователя поток событий описывается функцией распределения интервалов: $\tau_k \sim \omega_m(\tau)$. В общем случае поток является нестационарным. Однако для анализа производительности рассматривается участок с максимальной интенсивностью нагрузки, что по-

зволяет аппроксимировать поток простейшим пуассоновским процессом с параметром $\lambda_m = const$. Поступающие запросы формируют совокупный поток, являющийся суперпозицией пользовательских потоков, рис. 1.

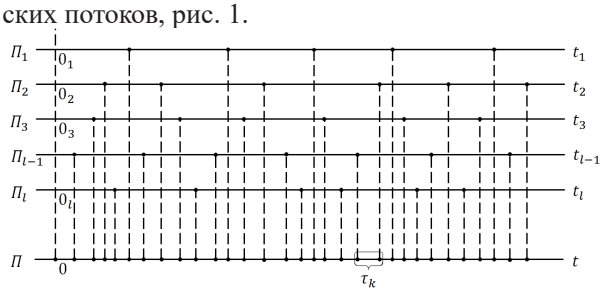

Рис. 1. Суперпозиция потоков

Модель обработки запросов. Система обработки запросов представляется как многоканальная система массового обслуживания (СМО) с ограниченной очередью:

$$S = (M,H), \tag{5}$$

где M – число параллельных каналов обслуживания, H – максимальная длина очереди.

Параметры системы обслуживания S определяют условия обработки входного потока запросов и напрямую влияют на наблюдаемые характеристики выполнения тестов. В частности, время выполнения каждого запроса Δt_l^s включает как собственно время выполнения операции ORM, так и задержки, обусловленные ожиданием в очереди и конкуренцией за ресурсы:

$$\Delta t_l^s = T_l^{exec} + T_l^{exec}(\lambda, M, H), \tag{6}$$

Показатели потребления ресурсов системы (оперативной памяти и CPU) являются функциями интенсивности входного потока и параметров системы обслуживания:

$$B_l^s = g(\lambda, M, H),\ C_l^s = h(\lambda, M, H). \tag{7}$$

Таким образом, система массового обслуживания выступает связующим звеном между моделью пользовательской нагрузки и наблюдаемыми показателями производительности, рис. 2.

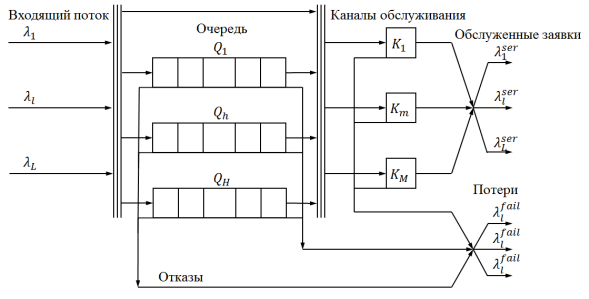

Рис. 2. Модель обслуживания заявок

Сбор и представление данных. Каждое выполненное событие (тест) характеризуется тройкой наблюдаемых величин:

$$(\Delta t_l^s, B_l^s, C_l^s), \tag{8}$$

где
- Δt_l^s – время выполнения теста s типа l,
- B_l^s – потребление оперативной памяти,
- C_l^s – загрузка CPU.

На основе этих данных формируются неэквидистантные временные ряды:

$$\left\{\Delta t_l^s\right\}, \left\{B_l^s\right\}, \left\{C_l^s\right\}, \tag{9}$$

отражающие динамику функционирования системы.

Интегральные показатели производительности**.** Для количественной оценки эффективности ORM автором введены агрегированные индексы производительности:

$$k_t = G_t\left(\left\{\Delta t_l^s\right\}\right), k_b = G_b\left(\left\{B_l^s\right\}\right), k_c = G_c\left(\left\{C_l^s\right\}\right), \tag{10}$$

где G_t, G_t, G_t– функции агрегации временных рядов.

Многокритериальный выбор ORM**.** Сравнение альтернативных ORM-фреймворков осуществляется в пространстве критериев (k_t, k_b, k_c) с использованием метода многокритериального анализа решений ELECTRE (ELimination Et Choix Traduisant la REalite) для сужения исходного множества альтернатив, пока количество элементов во множестве не достигнет требуемого значения для лица, принимающего решение (ЛПР) [16].

3. Эксперименты и полученные результаты

В рамках апробации предложенного подхода проведено моделирование поведения информационной системы с использованием группы ORM-фреймворков экосистемы .NET: Entity Framework, LINQ to SQL, NHibernate, Dapper, SubSonic, Insight Database. Выбор данной группы обусловлен их распространенностью в корпоративных ИС и сопоставимостью архитектурных решений. Нагрузка пользовательской активности формируется как суперпозиция потоков запросов различного типа. Рассматривались сценарии, имитирующие реальные профили эксплуатации (документооборот, поисковые нагрузки, смешанные CRUD-сценарии). Для апробации реализованного программно-математического комплекса был создан профиль тестирования А, приведенный в табл. 8, соответствующий системам документооборота. Было выполнено 5 тестовых прогонов для трех ORM. Время каждого цикла составило 300 минут. Заданное распределение операций отражает характерную нагрузку для

Табл. 2

Распределение и количество выполненных тестов для профиля документооборота

Тип теста	Распределение	Количество выполненных тестов					
		Entity Framework	LTS	NHibernate	Dapper	SubSonic	Insight Database
Get Struct Object	30	10800	14400	10800	14400	14400	14400
Put Struct Object	40	14400	19200	14400	19200	19200	19200
Search Struct Object	25	9000	12000	9000	12000	12000	12000
Search Simple	5	1800	2400	1800	1800	2400	2400

Табл. 3

Результаты индексов производительности отдельных тестов для профиля документооборота

Вид теста	Индексы производительности																	
	Entity Framework			LINQ to SQL			NHibernate			Dapper			SubSonic			Insight Database		
	k_t	k_b	k_c	k_t	k_b	k_c	k_t	k_b	k_c	k_t	k_b	k_c	k_t	k_b	k_c	k_t	k_b	k_c
GetStructObject	1,72	2,88	4,29	0,71	2,42	11,24	3,44	2,36	27,49	4,54	2,81	13,15	5,54	3,33	12,53	4,76	2,84	5,1
PutStructObject	187,24	4,97	24,08	212,76	2,37	47,84	219,33	2,38	31,25	198,34	5,04	14,95	241,97	5,75	53,34	207,86	4,65	28,6
SearchStructObject	121,11	30,42	18,54	115,95	32,64	34,58	252,65	58,46	54,52	120,45	29,8	26,08	146,95	35,2	16,51	126,23	31,9	22,01
SearchSimple	0,41	2,33	3,72	36,92	30,31	7,23	0,04	0,02	2,43	1,79	2,25	1,16	0,98	2,75	8,06	1,87	2,28	4,42

Табл. 4

Результаты индексов производительности в рамках профиля тестирования документооборота

Вид ORM	Индекс ресурса времени (меньше – лучше)	Индекс ресурса памяти (меньше – лучше)	Индекс ресурса CPU (меньше – лучше)
Entity Framework	310,4883	30,42	50,63
LINQ to SQL	366,3288	32,64	100,89
NHibernate	475,455	58,46	115,69
Dapper	325,12	29,8	55,34
SubSonic	395,44	35,20	90,45
Insight	340,85	31,90	60,12

Табл. 5

Распределение и количество выполненных тестов

Тип теста	Распределение	Количество выполненных тестов					
		Entity Framework	LINQ to SQL	NHibernate	Dapper	SubSonic	Insight DB
GetStruct Object	30	14400	14400	108800	14400	14400	14400
Search Simple	70	24800	18600	24800	29600	31400	29500

Табл. 6

Результаты индексов производительности в рамках профиля тестирования B

Вид ORM	Индекс ресурса времени (меньше – лучше)	Индекс ресурса памяти (меньше – лучше)	Индекс ресурса CPU (меньше – лучше)
Entity Framework	7,77	2,91	16,75
LINQ to SQL	479,20	60,65	37,24
NHibernate	4,83	2,37	24,41
Dapper	3,92	1,85	14,60
SubSonic	8,45	3,40	20,70
Insight DB	5,10	2,10	18,25

систем документооборота, где преобладают операции записи и выборки структурированных данных, а также поисковые сценарии различной сложности. Интенсивность потока запросов формируется как суперпозиция пользовательских потоков, что соответствует реальному многопользовательскому режиму эксплуатации.

Профиль A (документооборот)**.** Для первого эксперимента сформирован профиль нагрузки, табл. 2, моделирующий работу системы электронного документооборота небольшой организации с числом одновременно активных пользователей $M \in [50;100]$. Выбор данного профиля обусловлен типовыми функциональными и информационными потребностями пользователей в подобных системах, к которым относятся: работа со структурированными объектами (карточки документов, метаданные), создание и редактирование документов, поиск по атрибутам и содержимому, выполнение частых простых выборок. В соответствии с этим профиль включает следующие классы операций:

– извлечения структурированных объектов (GetStructObject);
– модификации (PutStructObject);
– сложного поиска (SearchStructObject);
– простого поиска (SearchSimple).

Временные ряды были проанализированы описанным выше методом, и полученные результаты приведены в табл. 3, 4. Значение индекса k_b выражается в байтах, но в таблицах результатов значения приведены в мегабайтах.

Профиль B (измененное распределение нагрузки)**.** Во втором эксперименте (табл. 5–6) изменено распределение операций в сторону увеличения доли простых выборок и протестирована на тех же ORM-системах.

С помощью предложенного подхода по выбору эффективной ORM-стратегии, базируемого на многокритериальном анализе решений ELECTRE, получено, что самой производительной альтернативной из тестируемых ORM-платформ в данной проекции ИС является Entity Framework. Под проекцией ИС подразумевается сценарий работы выполнения запросов пользователей к БД через механизмы кэширования ORM во времени. При изменении функционала воспроизводимой ИС и сценария использования данного функционала пользователем, представленного в табл. 5, производительными альтернативами стали ORM-решения Dapper и NHibernate, табл. 6.

По результатам исследования можно сказать, что эффективность ORM-компонент зависит от профиля ИС. Примером являются тесты SearchSimple, GetStructObject для NHibernate, показывающей лучшую производительность, табл. 3. Однако в рамках проекции ИС, представленной в 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 модели ведет к улучшению показателей производительности у компонента NHibernate, табл. 5, 6.

Дополнительные профили (C–E)**.** Дополнительно были сформированы 3 конфигурации ИС: профиль C, профиль D, профиль E. Данный профили охватывают функционал как CRUD-микротестов, так и N:M-1:M составные сценарии работы с данными, табл. 7-9.

С помощью метода ELECTRE дополнительно были проанализированы профили C-E. Результаты многокритериального анализа решений для нагрузочных профилей A-E представлены в табл. 10.

Табл. 7

Распределение и количество выполненных тестов (Профиль C)

Тип теста	Количество выполненных тестов					
	Entity Framework	LINQ to SQL	NHibernate	Dapper	SubSonic	Insight Database
SearchSimple	3600	4800	3600	4800	4800	4800
AddDocument	13200	14200	12200	21200	14600	21100
RemoveDocument	12800	14400	12100	20800	14600	18300

Табл. 8

Распределение и количество выполненных тестов (Профиль D)

Тип теста	Количество выполненных тестов					
	Entity Framework	LINQ to SQL	NHibernate	Dapper	SubSonic	Insight Database
GetFile	8200	8400	7400	10400	8200	10800
PutFile	8600	8000	8100	9400	8200	10700

Табл. 9

Распределение и количество выполненных тестов (Профиль E)

Тип теста	Количество выполненных тестов					
	Entity Framework	LINQ to SQL	NHibernate	Dapper	SubSonic	Insight Database
GetStructObject	9800	12400	10000	11400	10600	12200
SearchStructObject	8800	11000	8800	12100	11600	10400
SearchSimple	12400	15200	12100	16400	14100	16400
AddDocument	6800	6800	7500	4800	6300	5100

Табл. 10

Результат многокритериального анализа решений для нагрузочных профилей A-E

Профиль нагрузки	Многокритериальная оптимизация
A	{ EF | Insight | LTS | Dapper | SubSonic | NHibernate }
B	{ Dapper | NHibernate | Insight | EF | SubSonic |LTS }
C	{ Dapper | Insight | LTS | SubSonic |EF | NHibernate }
D	{ Dapper | Insight | LTS | SubSonic |EF | NHibernate }
E	{ NHibernate | EF | LTS | SubSonic | Dapper | Insight }

При изменении функционала воспроизводимой ИС (профиль нагрузки) и сценария использования данного функционала пользователем, меняется и вектор решений, что говорит о недостатке в подходах, представленных в работах [8-13]. Полученные результаты подтверждают вывод о несогласованности существующих подходов. В частности, в рамках профиля A (табл. 4) наилучшие показатели демонстрирует Entity Framework, тогда как NHibernate оказывается наименее эффективным решением. Это противоречит результатам работ [8–10], где NHibernate в ряде сценариев рассматривается как предпочтительная альтернатива. В то же время при изменении профиля нагрузки (профиль B, табл. 6) лидирующие позиции занимают Dapper и NHibernate, что уже частично согласуется с выводами в работах [8] и [10]. Таким образом, одно и то же ORM-решение может занимать как лидирующие, так и проигрышные позиции в зависимости от структуры нагрузки. Дополнительный анализ профилей C–E (табл. 10) показывает, что ранжирование альтернатив не является транзитивным и устойчивым: изменение функционального состава нагрузки приводит к изменению порядка предпочтительности ORM. Это объясняет противоречивость результатов, представленных в работах [8–13], и демонстрирует, что выводы, полученные в рамках изолированных микротестов, не могут быть обобщены на реальные многопользовательские системы.

Заключение

Предложенная в работе модель связывает профиль пользовательской нагрузки с наблюдаемыми показателями производительности системы, что позволяет корректно сравнивать ORM-фреймворки в условиях, приближенных к реальной

эксплуатации. В отличие от существующих подходов, основанных на изолированных микротестах, предложенный метод учитывает многопользовательский режим работы, стохастическую природу пользовательской активности и динамику изменения внутреннего состояния системы. Результаты проведенного моделирования наглядно демонстрируют, что эффективность ORM-компонент не является инвариантной характеристикой и напрямую зависит от функционального профиля информационной системы и сценариев взаимодействия пользователей с данными. В частности, одно и то же ORM-решение может демонстрировать как наилучшие, так и наихудшие показатели в зависимости от структуры нагрузки, что подтверждено на примере профилей A–E. Это объясняет противоречивость выводов, представленных в существующих исследованиях, и показывает ограниченность подходов, основанных на анализе отдельных CRUD-операций. Таким образом, задача выбора ORM-инструмента не может быть сведена к статическому сравнению альтернатив. Она должна рассматриваться как задача многокритериальной оптимизации в пространстве параметров нагрузки, учитывающей временную структуру запросов, интенсивность пользовательских потоков и композицию операций. Применение метода ELECTRE позволяет формализовать данный выбор и учитывать совокупность критериев в условиях неопределенности.

Практическая значимость работы заключается в возможности использования предложенного подхода при проектировании и оптимизации высоконагруженных информационных систем, в том числе систем документооборота и корпоративных приложений с числом одновременно работающих пользователей порядка 100 и более. Разработанная модель может быть адаптирована под различные предметные области за счет изменения профилей нагрузки и состава операций. В качестве направлений дальнейших исследований можно выделить расширение модели за счет учета распределенных архитектур, сетевых задержек, а также более детального моделирования механизмов кэширования и транзакционного взаимодействия. Кроме того, перспективным является интеграция предложенного подхода с инструментами автоматизированного подбора архитектурных решений в процессе проектирования информационных систем.

Литература

1. Bhatti S.N., Abro Z.H., Rufabro F. Performance evaluation of java based object relational mapping tool // Mehran University Research Journal of Engineering and Technology. 2013. Vol. 32. No 2. P. 159-166.
2. Gruca A., Podsiadło P. Performance analysis of. net based object–relational mapping frameworks // International Conference: Beyond Databases, Architectures and Structures. – Cham : Springer International Publishing. 2014. P. 40-49.
3. Крапухина Н.В., Курников П.А., Тарханов И.А. Многокритериальный метод оценки производительности ORM-решений в различных информационных системах // Труды ИСА РАН. 2015. Т. 65. № 2. С. 105-109.
4. Hule K., Ranawat R. Analysis of different ORM tools for data access object tier generation: a brief study // International Journal of Membrane Science and Technology. 2023. Vol. 10. No 1. P. 1277-1291.
5. Андрианова Е.Г., Мельников С.В., Раев В.К. Диссипация и энтропия в физических и информационных системах // Фундаментальные исследования. 2015. №. 8-2. С. 233-238.
6. Курников П.А., Крапухина Н.В. Реконструкция фазового пространства динамической системы высоконагруженного кэширующего механизма информационных систем // Информационные технологии и вычислительные системы. 2019. №. 1. С. 49-65.
7. Kurnikov P., Krapukhina N. Case Study of Self-Organization Processes in Information System Caching Components // International Journal of Advanced Computer Science and Applications. 2021. Vol. 12. No 6.
8. Zmaranda D. et al. Performance comparison of crud methods using net object relational mappers: A case study // International Journal of Advanced Computer Science and Applications. 2020. Vol. 11. No 1.
9. Drzazga K., Bobel M., Skublewska-Paszkowska M. Comparative analysis of selected object-relational mapping systems for the. NET platform // Journal of Computer Sciences Institute. – 2020. Vol. 16. P. 285-292.
10. Güvercin A.E., Avenoglu B. Performance analysis of object-relational mapping (orm) tools in. net 6 environment // Bilişim Teknolojileri Dergisi. 2022. Vol. 15. No 4. P. 453-465.
11. Ullah A. et al. Systematic performance, and Security evaluation of. NET models for accessing database // VFAST Transactions on Software Engineering. 2021. Vol. 9. No 4. P. 18-24.
12. Colley D., Stanier C., Asaduzzaman M. The impact of object-relational mapping frameworks on relational query performance // 2018 International Conference on Computing, Electronics &

Communications Engineering (iCCECE). – IEEE. 2018. P. 47-52.
13. Wiatrowski T. Comparative analysis of ORM systems for the. NET platform // Journal of Computer Sciences Institute. 2024. Vol. 31. P. 97-102.
14. Adya A., Bernstein P.A., Melnik S. Generation of query and update views for object relational mapping. U.S. Patent No. 7647298. 2010.
15. Albahari B., Simmons D. G. State transition logic for a persistent object graph. U.S. Patent No. 7526501. 2009.
16. Roy B. Classement et choix en présence de points de vue multiples // Revue française d'informatique et de recherche opérationnelle. 1968. Vol. 2. No. 8. P. 57-75.

Курников Павел Александрович. АО «Когнитив», г. Москва, Россия. Главный инженер исследователь. Область научных интересов: компьютерное зрение, системный анализ, математическое моделирование, системы управления базами данных. E-mail: p.kurnikov@cognitivepilot.com.

ORM selection in information systems based on user workload modeling and multicriteria analysis

P.A. Kurnikov
Cognitive JSC, Moscow, Russia

Abstract. The article addresses the problem of selecting efficient object-relational mapping (ORM) frameworks for real-world information systems. An analysis of existing studies shows that most approaches rely on CRUD microbenchmarks and do not account for the complex behavior of information systems under realistic workloads, resulting in contradictory and difficult-to-interpret conclusions. The proposed approach to ORM performance evaluation is based on discrete-event modeling of user activity and multicriteria analysis. A model describing user behavior as probabilistic event flows is developed, and a system of performance metrics including execution time, memory consumption, and CPU utilization is formalized. Computational experiments were conducted for different workload profiles simulating real-world operation scenarios of information systems. The results demonstrate that ORM performance depends on workload characteristics and usage scenarios. The proposed approach improves the validity of ORM framework selection and can be applied in the design of high-load information systems.

Keywords: *information systems, performance evaluation, ORM frameworks, database systems, workload modeling, multicriteria optimization, discrete-event simulation.*
DOI: 10.14357/20790279260208 **EDN:** WNJVIZ

References

1. Bhatti S. N., Abro Z. H., Rufabro F. Performance evaluation of java based object relational mapping tool. Mehran University Research Journal of Engineering and Technology. 2013;32(2):159-166.
2. Gruca A., Podsiadło P. Performance analysis of. net based object–relational mapping frameworks. International Conference: Beyond Databases, Architectures and Structures. Cham : Springer International Publishing. 2014. p. 40-49. https://doi.org/10.1007/978-3-319-06932-6_5
3. Kurnikov P. A., Krapuhina N. V., Tarkhanov I. A. A multicriteria method for evaluating the performance of ORM solutions in various information systems. Trudy ISA RAN. 2015;65(2):105-109. (In Russ.).
4. Hule K., Ranawat R. Analysis of different ORM tools for data access object tier generation: a brief study. International Journal of Membrane Science and Technology. 2023;10(1):1277-1291. https://doi.org/10.15379/ijmst.v10i1.2842
5. Andrianova E., Mel'nikov S., Raev V. Dissipation and entropy in the physical and information systems. Fundamental research. 2015;8(2):233-238. (In Russ.).
6. Kurnikov P. A., Krapuhina N. V. Phase space reconstruction of high-loaded caching mechanism dynamics in information systems. Informatsionnye Tekhnologii i Vychslitel'nye Sistemy. 2019;(1):49-65. (In Russ.). https://doi.org/10.14357/20718632190105
7. Kurnikov P., Krapukhina N. Case Study of Self-Organization Processes in Information System Caching Components. International Journal of Advanced Computer Science and Applications. 2021;12(6). https://doi.org/10.14569/IJACSA.2021.0120680
8. Zmaranda D. et al. Performance comparison of crud methods using net object relational mappers: A case study. International Journal of Advanced

Computer Science and Applications. 2020;11(1). https://doi.org/10.14569/IJACSA.2020.0110107
9. Drzazga K., Bobel M., Skublewska-Paszkowska M. Comparative analysis of selected object-relational mapping systems for the. NET platform. Journal of Computer Sciences Institute. 2020;(16):285-292. https://doi.org/10.35784/jcsi.2024
10. Güvercin A. E., Avenoglu B. Performance analysis of object-relational mapping (orm) tools in. net 6 environment. Bilişim Teknolojileri Dergisi. 2022;15(4):453-465. https://doi.org/10.17671/gazibtd.1059516
11. Ullah A. et al. Systematic performance, and Security evaluation of. NET models for accessing database. VFAST Transactions on Software Engineering. 2021;9(4):18-24. https://doi.org/10.21015/vtse.v9i4.752
12. Colley D., Stanier C., Asaduzzaman M. The impact of object-relational mapping frameworks on relational query performance. 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE). 2018. p. 47-52. https://doi.org/10.1109/iCCECOME.2018.8659222.
13. Wiatrowski T. Comparative analysis of ORM systems for the. NET platform. Journal of Computer Sciences Institute. 2024;(31):97-102. https://doi.org/10.35784/jcsi.6012
14. Patent USA № 7647298/01.01.2010. Adya A., Bernstein P. A., Melnik S. Generation of query and update views for object relational mapping. Available from: https://patents.google.com/patent/US7647298B2/en [Accessed: 15 March 2026]
15. Patent USA № 7526501/28.04.2009. Albahari B., Simmons D. G. State transition logic for a persistent object graph. Available from: https://patents.google.com/patent/US7526501B2/en [Accessed: 20 March 2026]
16. Roy B. Classement et choix en présence de points de vue multiples. Revue française d'informatique et de recherche opérationnelle. 1968;2(8):57-75. https://doi.org/10.1051/ro/196802V100571

Kurnikov Pavel A. Chief Research Engineer, Cognitive JSC, 6/2 Berezovaya str., Tomsk, 634063, Russia. Research interests: computer vision, systems analysis, mathematical modeling, database management systems. E-mail: p.kurnikov@cognitivepilot.com