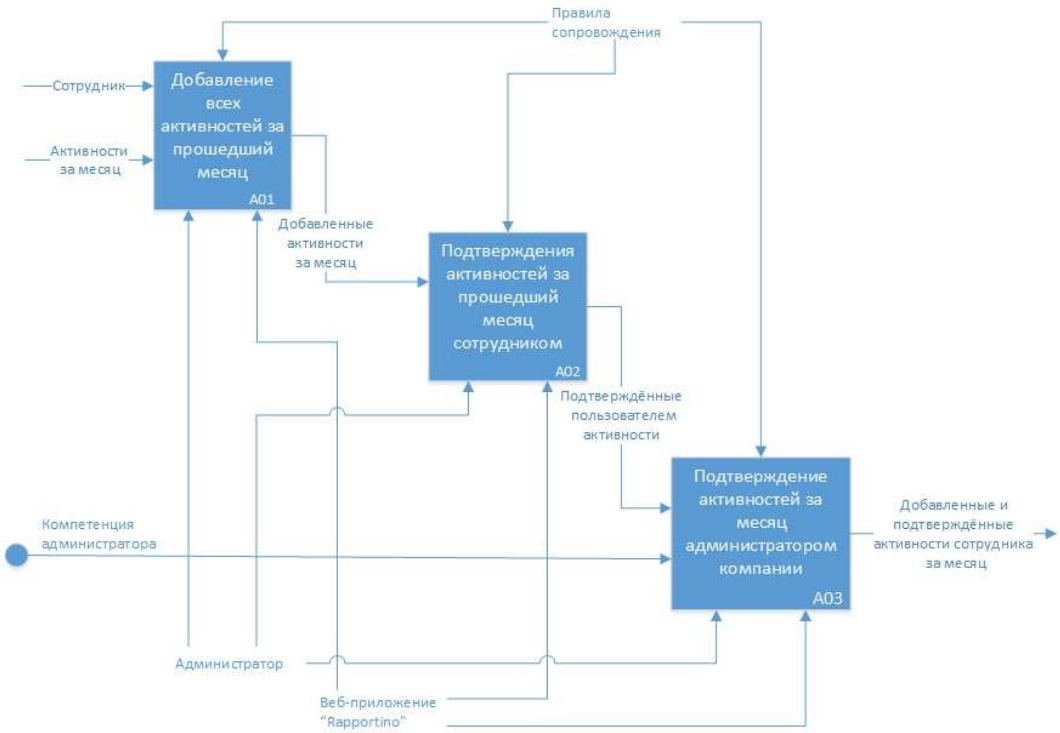

Рис.4 –Диаграмма декомпозиции для веб-приложения “Rapportino”

Литература

1. Долинина О.Н. Информационные технологии в управлении современной организацией: Монография / О.Н. Долинина. - Саратов: Сарат. гос. техн. ун-т, 2006.

2. Bolshakov A., Dolinina O. N., Perova L., Sytnik A. Combined System for Management of Formation of Competences of Technical University Students // ICEE/ICIT-2013: Proceedings of the Joint International Conference on Engineering Education and Research and International Conference Information Technology, Cape Town, 8 -12 December 2013. 2013. С. 308-315

3. Mark Corker. Top 10 Microsoft Project Problems [Электронный ресурс]. – Режим доступа http://www.slideshare.net/MarkCorker/top-10-ms-project-problems. – (Дата обращения: 23.01.2017).

И.Д. Семенов, Е.И. Чигарина

СРАВНИТЕЛЬНЫЙ АНАЛИЗ JPA ПРОВАЙДЕРОВ В СИСТЕМАХ БАЗ ДАННЫХ

(Самарский университет)

В данной работе приведен сравнительный анализ реализаций спецификации Java Persistence Api – набора инструментов, позволяющих решать задачи по задаче объектно-реляционного отображения (преобразования данных, используемых в современном объектно-ориентированном подходе, в информацию, хранящуюся в таблицах реляционных баз данных) [1].

Основными критериями сравнения были выбраны следующие параметры:

- запрос, формируемый к системе управления базой данных (СУБД);
- время выполнения запроса;
- способы выполнения запроса к СУБД (напрямую или с использованием кэширования).

Сравнительный анализ проведен на ПК со следующими характеристиками: ОС Windows 7 64-bit, Intel Core i3-3110M 2,40GHz, версия JDK – 1.8, сервер приложений – Apache Tomcat, в качестве СУБД выбрана PostgreSQL 9.1.3-901.

В качестве объектов исследования использовались наиболее популярные открытые реализации последней версии JPA спецификации: Hibernate (5.2.7.Final), OpenJpa (2.4.2), EclipseLink (2.5.2) [2].

Для проведения анализа была выбрана база данных выставочной деятельности самарского университета, логическая схема которой представлена на рисунке 1.

Для каждого из JPA провайдеров был разработан отдельный модуль, который разворачивался на сервере приложений Tomcat. Для получения результатов перед каждым выполнением запроса производился перезапуск сервера.

В таблице 1 представлены средние временные характеристики добавления разного количества объектов типа Exhibit посредством операции *persist()*.

Таблица 1 – Среднее время добавления объектов типа Exhibit

Провайдер	Количество записей		
	1	1000	100000
EclipseLink	2 мс	27 мс	347 мс
OpenJpa	20 мс	178 мс	1152 мс
Hibernate	12 мс	116 мс	851 мс

Таблица 2 отражает средние результаты выполнения запросов по выборке всех данных указанного типа (количество записей 17500), в таблице 3 приведено среднее время для выборки одной записи с заданным идентификатором.

Из приведенных выше данных видно, что запросы, формируемые к СУБД, имеют практически индентичную структуру, и так как обращение происходит к СУБД напрямую, и время, затраченное на выполнение запроса на стороне СУБД будет одинаковым, то фактором, влияющим на различие во времени, является то, как каждый из JPA провайдеров обрабатывает сущностные классы, используемые для объектно-реляционного отображения.

Важным этапом взаимодействия между уровнем бизнес-логики и уровнем данных в веб-приложении с целью снижения нагрузки на БД является возможность использования системы кэширования. В JPA спецификации выделяют два уровня кэширования (кэш первого уровня, используемый в рамках одной сессии, и кэш второго уровня, применяемый при работе с различными сессиями).

В рамках данного приложения рассмотрено, как использование системы кэширования в JPA провайдерах влияет на время выполнения запросов, рас-

сматриваемых ранее. Для примера было проведено повторное выполнение запросов, результаты которых представлены в таблицах 2 и 3. В данном случае запрос выполнялся без перезапуска сервера (перезапуск осуществлялся только при смене JPA провайдера). В таблице 4 приведены полученные результаты, исходя из которых прослеживается существенная разница между временем, полученным при первом выполнении запроса и последующими (в таблице приведены только первые три результата ввиду того, что последующие выполнения запроса имеют несущественные отличия от времени, полученным на третьей итерации).

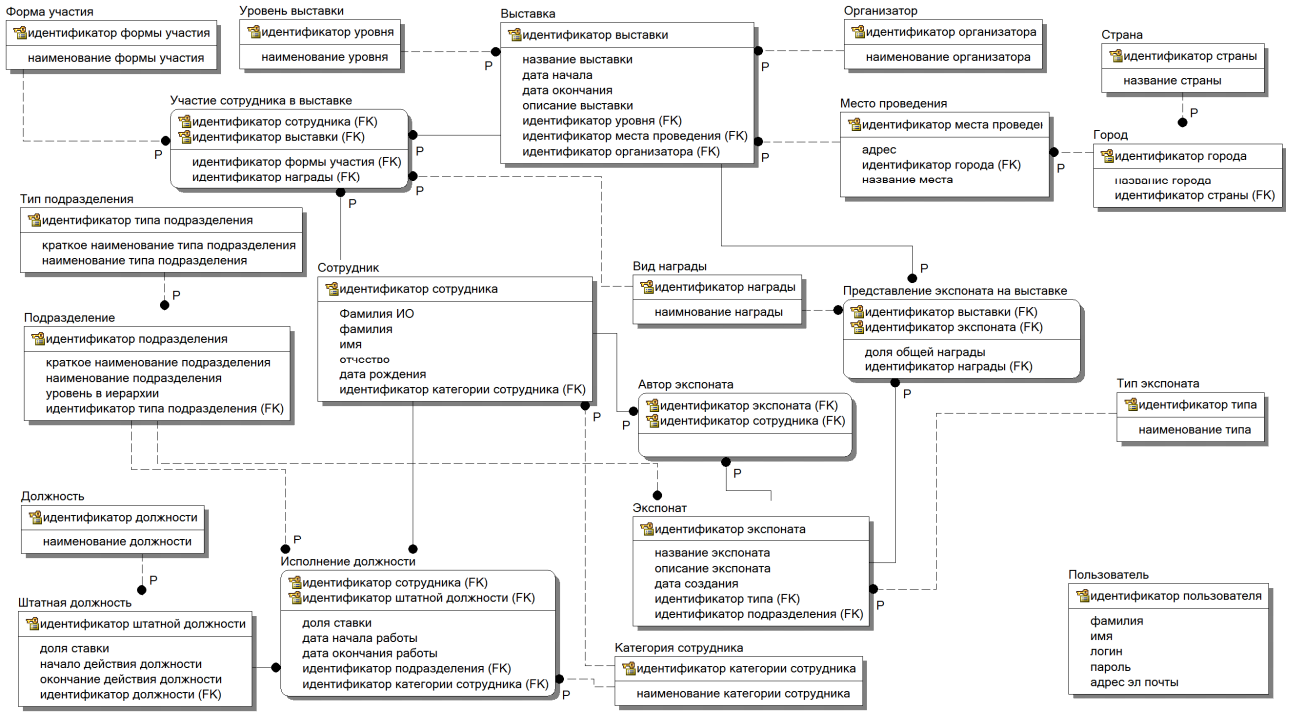

Рисунок 1 – Логическая модель базы данных, используемая для проведения анализа

Таблица 2 – Среднее время выполнения запроса по выборке данных

Провайдер	Время выполнения запроса	Сформированный запрос
Hibernate	3669 мс	select exhibito_.ID as exhibitIDo_, exhibito_.nameo_ as nameo_, exhibito_.creationDate as creationDateo_, exhibito_.description as descriptiono_ from exhibit exhibito_
EclipseLink	3469 мс	SELECT Id, name, creationDate, description FROM exhibit
OpenJpa	5887 мс	SELECT to.Id, to.name, to.creationDate, to.description FROM exhibit to

Таблица 3 – Среднее время выборки отдельной записи

Провайдер	Время выполнения запроса	Сформированный запрос
Hibernate	157 мс	select exhibito_.ID as exhibitIDo_, exhibito_.nameo_ as nameo_, exhibito_.creationDate as creationDateo_, exhibito_.description as descriptiono_ from exhibit exhibito_ where exhibito_.id=?
EclipseLink	521 мс	SELECT Id, name, creationDate, description FROM exhibit WHERE (Id=?)
OpenJpa	1052 мс	SELECT to.Id, to.name, to.creationDate, to.description FROM exhibit to WHERE to.id=? [params=?]

Таблица 4 – Время выполнения запросов с использованием кэширования:

Запрос/номер повторного выполнения	EclipseLink	Hibernate	OpenJpa
Выборка всех записей (1)	3389 мс	3568 мс	5779 мс
Выборка всех записей (2)	506 мс	272 мс	521 мс
Выборка всех записей (3)	440 мс	218 мс	263 мс
Выборка записи по идентификатору (1)	521 мс	156 мс	1050 мс
Выборка записи по идентификатору (2)	1 мс	6 мс	3 мс
Выборка записи по идентификатору (3)	1 мс	3 мс	3 мс

Исходя из полученных данных, можно сделать вывод, что из рассмотренных JPA провайдеров лучшие результаты показал Hibernate. Основным преимуществом использования технологии объектно-реляционного отображения является возможность существенно сократить качество кода, требуемого для организации работы между Java-объектами и таблицами реляционных баз данных. Использование JPA спецификации включает в себя такие преимущества как построение графа объектов, загрузка объектов по требованию, система кэширования и т.д.

Далее планируется провести анализ по временным характеристикам при загрузке разного количества объектов из графа объектов, провести более детальное сравнение различных уровней кэширования, используемых в данных провайдерах, а также провести анализ по выбранным критериям с такими СУБД, как MySQL, HSQLDB, Oracle.

Литература

1. Introduction to Java Persistence Api [Электронный ресурс]. – https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html

2. EclipseLink Developer Guide [Электронный ресурс]. – https://www.eclipse.org/eclipselink/releases/2.6.php.

А.А. Столбова, С.В. Востокин, С.Н. Попов

ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТОВ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЯ НА КЛАСТЕРНЫХ СИСТЕМАХ

(Самарский национальный исследовательский университет имени академика С.П. Королёва)

В настоящее время для анализа сигналов широко используется вейвлет-преобразование [1]:

$$W_{\psi}(a_i, b_j) = \frac{1}{\sqrt{a_i}} \int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-b_j}{a_i}\right)dt,$$

где f(t) – случайный процесс, ψ(t) – выбранный анализирующий вейвлет, a≠0 – параметр масштаба, b≥0 – параметр сдвига.

Одной из прикладных задач вейвлет-преобразования является анализ акустических сигналов при тестировании деталей на прочность. Часто такие сигналы имеют большой объем и, следовательно, решение данной задачи при помощи непрерывного вейвлет-преобразования занимает много времени.

Целью данной работы является демонстрация возможностей применения высокопроизводительных вычислений на кластерных системах для уменьшения времени проведения вычислительных экспериментов вейвлет-анализа сигналов.

Для проведения вычислительных экспериментов был выбран базовый вейвлет Гаусса 8 порядка [2, 3]:

$$\psi(\tau) = \left(\tau^8 - 28\tau^6 + 210\tau^4 - 420\tau^2 + 105\right)\exp\left(-\frac{\tau^2}{2}\right),$$

где $\tau = \frac{t-b}{a}$.

В качестве анализируемого сигнала использовался акустический сигнал с интервалом дискретизации $\Delta t = 5 \cdot 10^{-7}$ и числом отсчетов N=1500000.

Для распараллеливания использовалась технология OpenMP [4], предназначенная для вычислительных систем с общей памятью. Для того чтобы получить параллельную версию программы, необходимо определить ресурс её параллелизма, то есть найти в ней участки, которые могут выполняться независимо. Структура численного метода, реализующего непрерывное вейвлет-преобразование такова, что он имеет большой ресурс параллелизма. Каждый отсчет функции W_{Ψ} может вычисляется независимо и одновременно с другими отсчетами. Данное свойство позволило легко преобразовать исходный последовательный алгоритм в параллельный, выделив в нем параллельно выполняющиеся циклы с использованием директив *parallel* и *for* OpenMP.