Проектирование информационной обучающей системы: выбор СУБД и способа доступа к данным

О.К. Головнин
Факультет информатики
Самарский национальный исследовательский университет имени академика С.П. Королева
Самара, Россия
e-mail: golovnin@bk.ru

Д.Е. Егоров
Факультет информатики
Самарский национальный исследовательский университет имени академика С.П. Королева
Самара, Россия
e-mail: edaniile@mail.ru

Аннотация[1]

Рассмотрена проблема выбора системы управления базами данных (СУБД) при проектировании автоматизированной обучающей системы. Рассмотрены популярные алгоритмы и модели доступа к данным в реляционных и NoSQL СУБД на примере проектируемой автоматизированной обучающей системы. Проведен сравнительный анализ различных СУБД и их возможностей. Проанализированы такие параметры, как функционал диалектов SQL, время отклика, надежность, доступность СУБД, масштабируемость хранимых данных. Приведены результаты расчетов и используемые модели в нотациях UML и ER.

1. Введение

При проектировании информационных систем одна из главных проблем, с которыми сталкивается разработчик – проблема выбора способа хранения данных. В подавляющем большинстве случаев используются реляционные СУБД (РСУБД), обеспечивающие целостность данных, избавляя тем самым разработчика от необходимости проверки консистентности на уровне бизнес-слоя системы. Но в случаях, когда количество хранимой информации начинает исчисляться в десятках терабайт, возникает необходимость в использовании нескольких физических серверов, в чем гораздо сильнее бессхемные СУБД.

В данной статье проводится анализ сильных и слабых сторон РСУБД и NoSQL, а также обзорное сравнение самых популярных СУБД каждого вида, применительно к задаче автоматизации обучения.

Труды Шестой всероссийской научной конференции "Информационные технологии интеллектуальной поддержки принятия решений", 28-31 мая, Уфа-Ставрополь, Россия, 2018

2. Анализ возможностей NoSQL СУБД

2.1. Классификация NoSQL СУБД

NoSQL – общее описание ряда подходов, в которых требуется решить проблему масштабируемости и доступности данных за счет их атомарности и целостности [1].

По модели хранимых данных и подходов к реализации распределенности и репликации выделяется четыре типа NoSQL СУБД [2]:

- «Ключ-значение» (MemcacheDB, Redis, Riak и пр.);
- Документно-ориентированные (MongoDB, CouchDB, MarkLogic и пр.);
- Хранилище семейств колонок (Cassandra, Hypertable, SimpleDB и пр.);
- Графовые (Neo4j, Blazegraph, FlockDB).

Базы данных первого типа предназначены для хранения медиаданных и представляют набор хэш-ключей для хранимых объектов.

Документно-ориентированные СУБД хорошо подходят для построения иерархических структур данных, например, систем управления содержимым.

Хранилище семейств колонок использует модель хранения данных на базе семейства столбцов, что позволяет хранить хэш-ключи на нескольких уровнях вложенности.

Графовые СУБД позволяют хранить большое количество связей, являясь, тем самым, крайне удобным инструментом в написании социальных сетей. В силу своей структуры не требуют таких сложных вычислений, как, например, операция JOIN в SQL.

Основное отличие баз данных на основе графов от вышеперечисленных NoSQL СУБД заключается в поддержке ACID-транзакций.

2.2. Преимущества NoSQL СУБД

Очевидно, что в каждой из вышеперечисленных категорий NoSQL СУБД разработчикам требовалось

решить различные проблемы, что, как следствие, сводит к минимуму сравнение, например, таких систем, как FlockDB и Cassandra, поэтому данная статья ставит своей целью выявление критериев, в случае которых имеет смысл отказаться от использования реляционных баз данных в проекте, или подключить более одной СУБД разных типов.

В первую очередь, большинство NoSQL систем имеют встроенные алгоритмы репликации и шардирования, что в разы повышает их способность к горизонтальному масштабированию. Как следствие, возникает необходимость в использовании бессхемных СУБД, когда речь заходит о BigData.

Во-вторых, отсутствие жесткой схемы данных позволяет осуществлять более гибкий подход к добавлению новых полей в таблицы.

В-третьих, в случае, когда специфика проектируемой системы заточена под хранение большого объема медиаданных, связей между объектами и т.п., имеет смысл обратиться к таким NoSQL СУБД, которые располагают встроенными средствами для более удобного манипулирования конкретным типом данных.

Например, известная социальная сеть Twitter, изначально построенная на MySQL, испытывала проблемы с хранением связей «многие ко многим» и необходимостью обхода деревьев. Решением этих проблем стало использование графовой СУБД FlockDB поверх MySQL. В результате, при осуществлении 20000 записей и 100000 операций чтения в секунду, FlockDB предоставляет следующую производительность [3]:

- подсчет количества строк: 1 мс;
- временные запросы: 2 мс;
- запись: 1 мс для журнала, 16 мс для надежной записи;
- обход дерева: 100 граней/мс.

2.3. Недостатки NoSQL СУБД

В силу того, что, по CAP-теореме [4], в пользу достижения высокой доступности, большинство бессхемных СУБД «согласованы в конечном счете», что значит, что распределенная система будет консистентна через неопределенный конечный промежуток времени в случае отсутствия изменений в данных, ACID-транзакции не поддерживаются. Как следствие, такие СУБД противопоказаны к использованию в контексте денежных средств.

Во-вторых, NoSQL СУБД не проверяют целостность данных на уровне базы данных, что, в случае необходимости проверки консистентности, возлагает эту задачу на бизнес-слой приложения.

Кроме того, по данным сайта db-engines.com, учитывающего такую статистику, как общая заинтересованность в конкретной СУБД, релевантность в социальных сетях, количество упоминаний системы в предложениях о работе и пр., построена следующая статистика (рисунок 1) [5]:

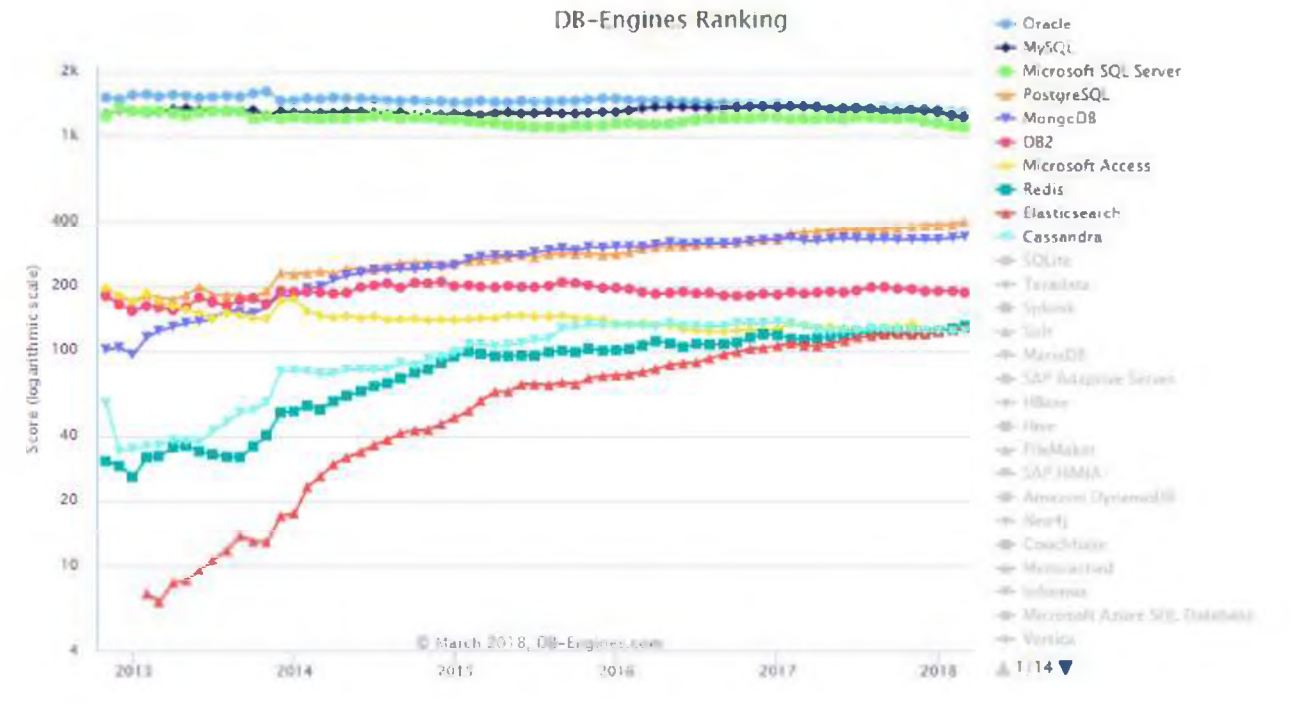

Рис. 1. Трендовая диаграмма СУБД

Как видно из диаграммы, из десяти наиболее популярных СУБД, шесть – реляционные. Кроме того, три системы, на порядок более популярные всех остальных, также относятся к классу реляционных СУБД, что значит, что на данных момент, системы, использующие в качестве хранилища данных РСУБД, могут рассчитывать на более доступную техническую поддержку и больший опыт разработчиков.

3. Аналитический обзор популярных реляционных СУБД

3.1. Доступность

1. Oracle – коммерческая лицензия.
2. MySQL – имеет двойное лицензирование. Распространяется по лицензии GPL, что заставляет программы, написанные с использованием MySQL также распространяться по лицензии GPL. В случае, если разработчик не желает открывать исходные коды программ, предусмотрена коммерческая лицензия.
3. Microsoft SQL Server – пользовательское соглашение Microsoft EULA.
4. PostgreSQL – свободная, существуют варианты коммерческой поддержки продукта, а также расширение для обеспечения совместимости с Oracle Database.

3.2. Функциональность

1. Oracle [6]:
   - многоверсионность данных для управления параллельными транзакциями;
   - пакеты;
   - поддержка последовательностей;
   - аналитические функции в SQL;
   - потоки;
   - объектные типы;
   - автоматический мониторинг и диагностика баз для выявления проблем производительности и возможность автоматической корректировки.
2. MySQL [7]:
   - поддержка полусинхронного механизма репликации;
   - дополнительный набор функций для обработки XML;
   - API для плагинов, позволяющий загружать сторонние модули без перезапуска сервера;
   - соединения клиент-сервер, защищенные через SSL.
3. Microsoft SQL Server [8]:
   - встроенный механизм репликации;
   - поддержка JSON, XML
   - поддержка всех существующих драйверов и фреймворков;
   - улучшенная производительность за счёт заранее скомпилированных модулей Transact-SQL.
4. PostgreSQL [9]:
   - многоверсионность данных для управления параллельными транзакциями;
   - возможность написания функций на таких языках, как C, C++, Java, а также ряде скриптовых языков;
   - расширенный набор встроенных типов данных, в т.ч. геометрические примитивы, сетевые данные, JSON и пр.;
   - возможность создания пользовательских типов и программировать для них механизмы индексирования.

3.3. Время выполнения запроса

С целью выявить СУБД с наибольшим быстродействием был проведен сравнительный тест MySQL и PostgreSQL, как двух свободно распространяемых СУБД.

Тестирование выполнялось с помощью консольного приложения на языке C#, имитирующего запросы, наиболее часто встречающиеся в обещающих системах. Алгоритм тестирования заключался в следующем:

1. Установка соединения с сервером баз данных;
2. Установка первой отметки времени;
3. Выполнение запроса на запись или чтение в цикле 32768 раз;
4. Установка второй отметки времени;
5. Разность между второй и первой отметками и есть искомый результат.

Результаты теста приведены в таблице 1:

Таблица 1. Скорость выполнения запросов в СУБД MySQL и PostgreSQL

	Чтение, с	Запись, с
MySQL	59,313	2055,502
PostgreSQL	5,681	17,108

Для наглядности полученные данные приведены на диаграммах (рисунки 2, 3).

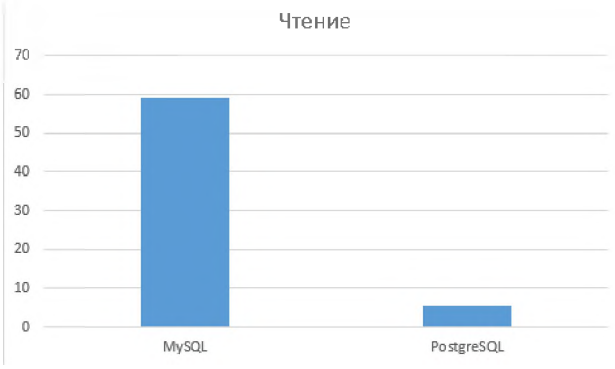

Рис. 2. Сравнение временных затрат СУБД на чтение данных

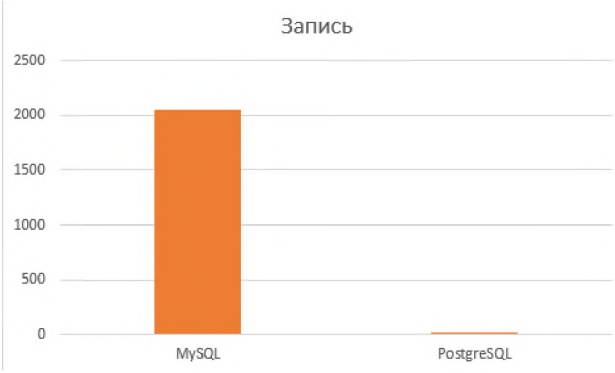

Рис. 3. Сравнение временных затрат СУБД на запись данных

Как видно по результатам расчетов, PostgreSQL выигрывает более, чем в 10 раз, по временным затратам на чтение данных и более, чем в 120 раз на запись.

4. Проектирование автоматизированной обучающей системы

В [10] проводится анализ требований к проектируемой системе. Было принято решение геймифицировать обучающую систему. Планируется, что игровой процесс будет проходить на электронной карте, за территории которой будет происходить соревновательный процесс, заключающийся в наиболее точном и быстром ответе на задачи системы.

Более подробно с функционалом системы можно ознакомиться на приведенной диаграмме вариантов использования (рисунок 4).

На основании диаграммы вариантов использования была разработана модель базы данных в нотации ER (рисунок 5).

На вышеприведенной модели ясно прослеживается отношение всех сущностей системы. Видно, что данные достаточно однозначно структурированы, и в ходе эксплуатации системы вероятность необходимости изменений полей таблиц крайней низка.

В подавляющем большинстве данные либо текстовые, либо числовые, что сводит к минимуму необходимость прибегать к специальным средствам для хранения медиаданных и быстрого доступа к ним.

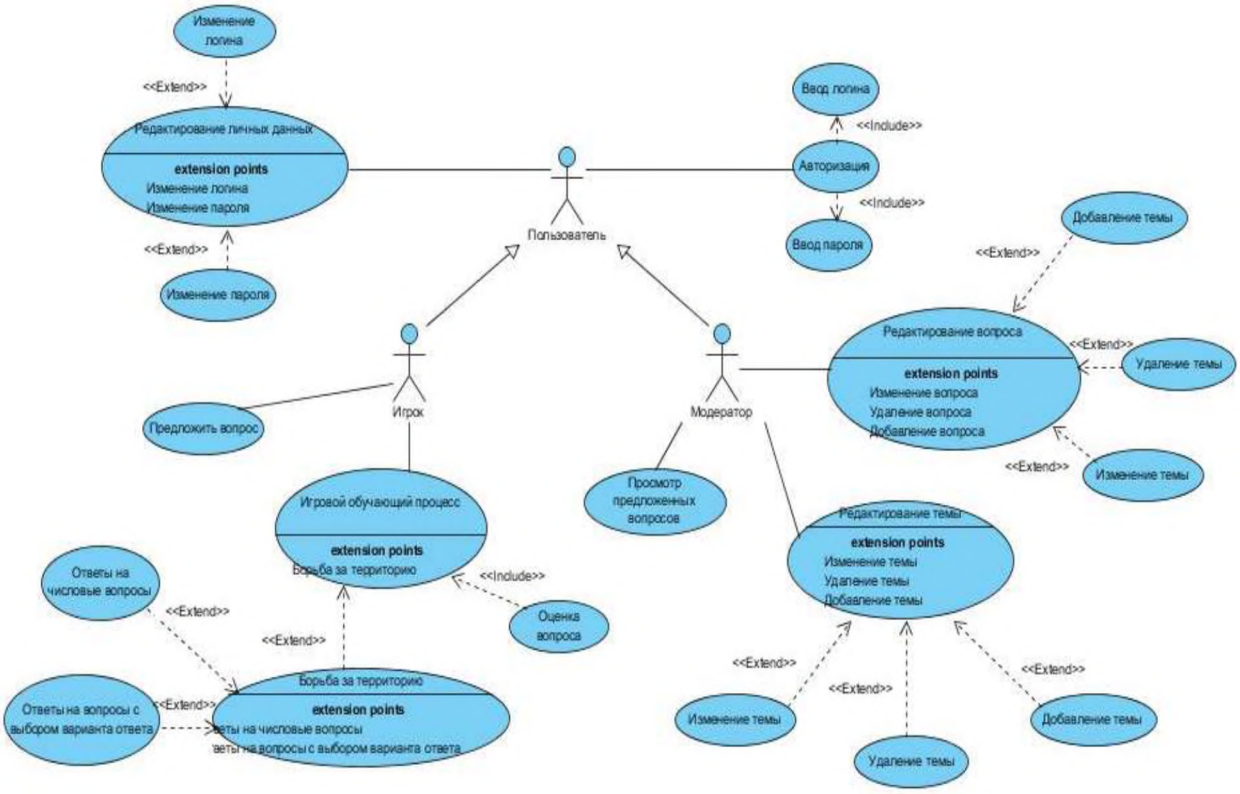

Рис. 4. Диаграмма вариантов использования разрабатываемой системы

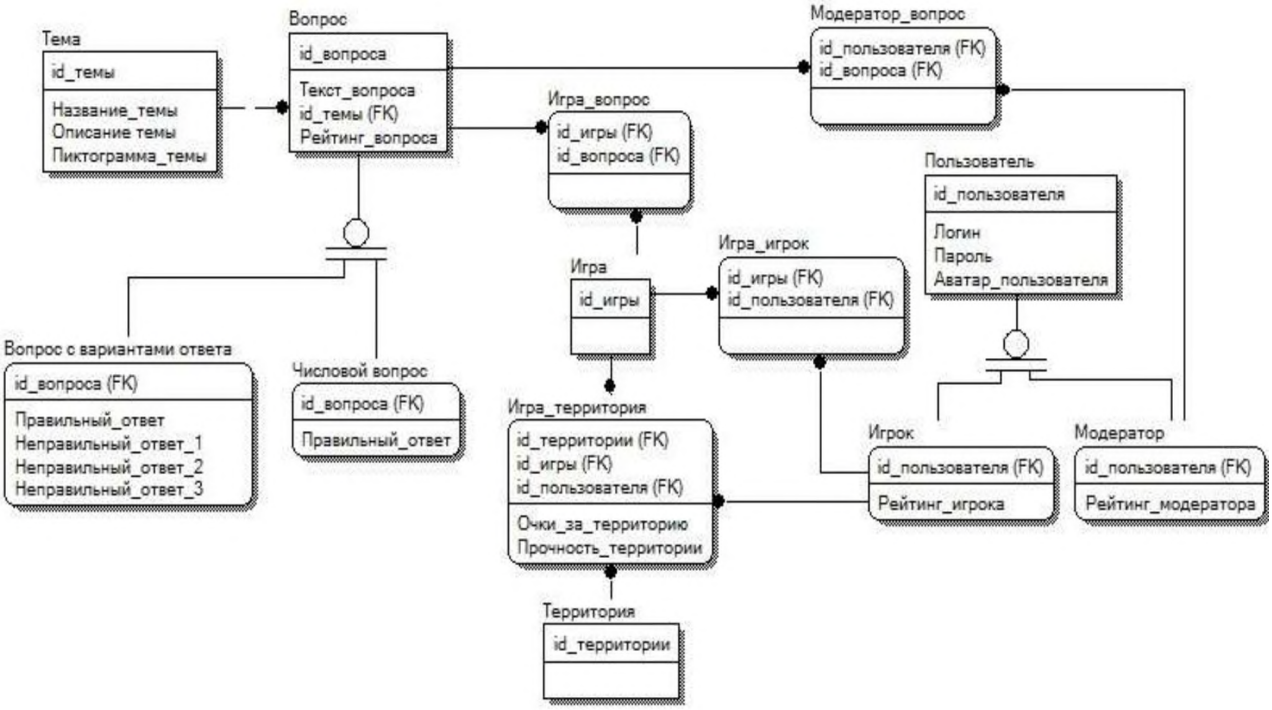

Рис. 5. ER-модель системы

Предполагается, что длина каждой из таблиц не будет превышать 10000 записей, следовательно, использование инструментов обработки больших данных выглядит нерациональным.

Таким образом, очевидно, что использование NoSQL СУБД в проектируемой автоматизированной обучающей системе не только не является необходимым, но и может негативно сказаться на этапах разработки и эксплуатации проекта.

Выбор реляционной СУБД обоснован, в первую очередь, доступностью системы: из рассмотренных выше это MySQL и PostgreSQL.

Поскольку игровой процесс происходит на электронной карте, необходимо, чтобы СУБД поддерживала геометрические данные. Несмотря на то, что в MySQL существуют расширения типов данных для поддержки географической информации OpenGIS, данные сначала должны быть сконвертированы в геометрический формат простыми командами перед тем, как будут вставлены в таблицу [11].

Таким образом, в качестве СУБД проектируемой автоматизированной обучающей системы на основе электронной карты выбрана PostgreSQL.

5. Заключение

- В проектах, где количество хранимой информации не исчисляется десятками терабайт, использование NoSQL СУБД не является необходимостью.
- В случае больших объемов обрабатываемой информации, бессхемная СУБД, размещенная между РСУБД и бизнес-слоем приложения может позволить добиться высокой производительности.
- В силу того, что реляционные СУБД на данный момент более популярны, разработчик может рассчитывать на более квалифицированную поддержку, используя РСУБД.
- Учитывая отрыв, с которым самые известные РСУБД опережают в популярности самые известные NoSQL СУБД, можно прогнозировать, что актуальность реляционных баз данных не будет уменьшаться ближайшие несколько лет.
- Коммерческие СУБД, как правило, располагают функционалом, необходимости в котором рядовой разработчик не испытывает.
- PostgreSQL выигрывает у MySQL более, чем в 10 раз, по временным затратам на чтение данных и более, чем в 120 раз, по временным затратам на запись.
- Из наиболее популярных некоммерческих СУБД наибольшим функционалом в области обработки и хранения геоданных обладает PostgreSQL.

Список используемых источников

1. Gaurav Vaish. Getting Started with NoSQL. – Packt Publishing, 2013. – 142 p.

2. Dan McCreary, Ann Kelly. Making Sense of NoSQL: A guide for managers and the rest of us. – Manning Publications, 2013. – 312 p.
3. 140 миллионов твитов в день: как работает Twitter изнутри? [Электронный ресурс] // «Хакер». – Безопасность, разработка, DevOps. – URL: https://xakep.ru/2011/06/01/55835.
4. Towards Robust Distributed Systems [Электронный ресурс] / UC Berkley – URL: https://people.eecs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf.
5. Method of calculating the scores of the DB-Engines Ranking [Электронный ресурс] // DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems. – URL: https://db-engines.com/en/ranking_definition.
6. What's New with Database Cloud [Электронный ресурс] // Oracle | Integrated Cloud Applications and Platform Services. – URL: https://www.oracle.com/database/index.html.
7. MySQL 5.7 Release Notes [Электронный ресурс] // MySQL: Developer Zone. – URL: https://dev.mysql.com/doc/relnotes/mysql/5.7/en/.
8. Новые возможности SQL Server 2017 [Электронный ресурс] // Microsoft – официальная страница. – URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-2017-features.
9. User Documentation [Электронный ресурс] // PostgreSQL: The world's most advanced open source database. – URL: https://wiki.postgresql.org/wiki/Main_Page.
10. Головнин О.К., Егоров Д.Е. Формирование требований к обучающей системе: некоторые аспекты геймификации // ИТ & Транспорт : сб. науч. статей. – Самара: Интелтранс, 2017. – Т. 8. – С. 32-39.
11. Чем PostgreSQL лучше других SQL баз данных с открытым исходным кодом. Часть 1. [Электронный ресурс] // Хабрахабр. – URL: habrahabr.ru/post/282764/.