

InterSystems Caché и технологии NoSQL

oleg.olenin@intersystems.com

Современные высоконагруженные приложения изменили требования к СУБД - сегодня необходимы простые и эффективные инструменты создания специализированных решений с гарантированным временем реакции при обработке больших массивов данных. Вместе с тем, несмотря на появление таких относительно новых технологий как NoSQL, потенциал давно существующих подходов реализован еще не полностью.

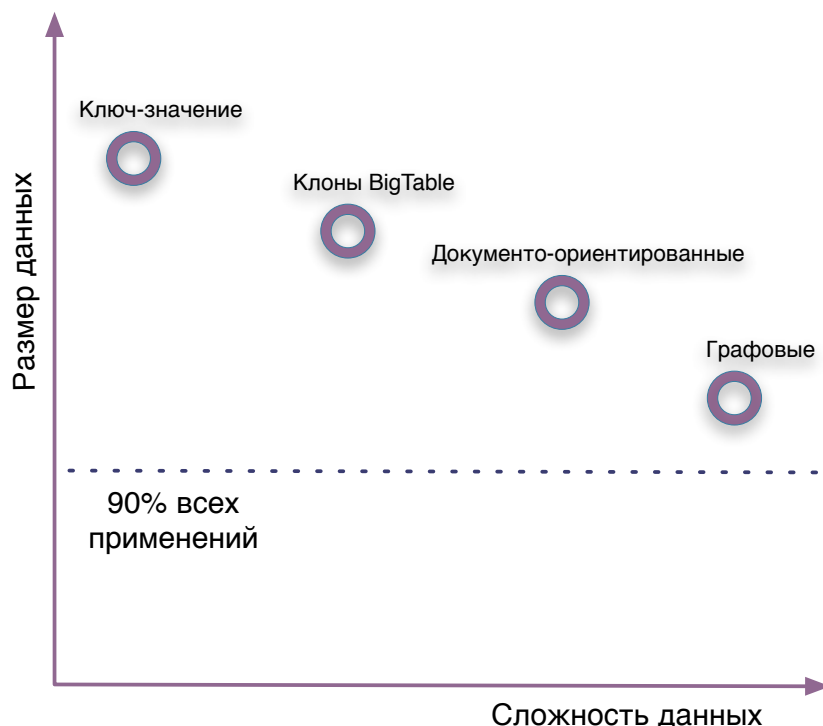
Интернет проекты с высокой нагрузкой и приложения класса ХТР (extreme transaction processing) изменили требования к технологиям СУБД. Приоритетными требованиями стали простота разработки, возможность специализации технологии хранимых данных для конкретного проекта, поддержка постоянного времени реакции системы при увеличении нагрузки, обеспечение низкой стоимости масштабирования и стоимости обработки больших объемов данных.

В качестве ответа на новые потребности возникло движение NoSQL - новый класс баз данных, который обещает разработчикам высокую скорость внесения изменений в приложения, низкие затраты на масштабирование и обработку/хранение больших объемов данных, высокую скорость работы на относительно недорогом железе - ценности, которые всегда были важны и для технологий компании InterSystems. Практически всегда NoSQL базы реализуют отличную от привычной парадигму работы приложений с базами данных - переход от концепции интегрирующей СУБД для нескольких приложений к концепции СУБД для одного приложения или одного проекта и больше - отдельной специфичной задачи в рамках проекта.

Родовыми чертами NoSQL стали использование нереляционных моделей данных, простые API или протоколы доступа (в сравнении с традиционными), способность к горизонтальному масштабированию по требованию для некоторого набора операций на многих серверах, распределенное хранение данных на многих серверах, эффективное использование распределенных индексов и памяти для запросов, свободное обращение с серьезными и незыблемыми для традиционных СУБД вещами - целостностью данных и транзакциями.

Сегодня существует более сотни NoSQL-решений, отличающихся подходами к масштабированию и распределенному хранению данных, поддерживаемые модели данных и схемы хранения (в том числе реализация хранения). Отдельным пунктом сравнения несомненно являются запросы к хранимым данным и их исполнение - в мире NoSQL стандартного языка запросов пока не существует, а прозрачное

понимание принципа работы необходимо для успешной реализации запросов разработчиком.



↑ По своей сути дизайн NoSQL решений направлен либо на борьбу с большим объемом данных либо с их повышенной сложностью. Идея взята из презентации автора Neo4J Emil Eifrem.¹

Общее, что объединяет NoSQL проекты это широкое использование компромиссов по отношению к взаимно противоречащим требованиям, отказ от которых был невозможен и являлся догмой для традиционных СУБД - например, уход от поддержки всех свойств ACID в пользу горизонтальной масштабируемости. Самым популярным способом объяснения причин, по которым компромиссы естественны, является CAP теорема. Вольно интерпретируя ее смысл можно сказать, что невозможно быть надежным, быстрым, распределенным и целостным одновременно - однако могут быть варианты.

Другой источник компромиссов – характер данных решаемой задачи. Именно тут важно требование к технологии, которая должна быть гибкой и максимально использовать особенности предметной области. Например, если можно распараллелить обработку данных и использовать принцип «shared nothing» (без разделения ресурсов), то нужно это эффективно использовать как для хранения, так и для исполнения запросов. В этом случае надо строить модель хранения и распределения данных, которая опирается на эту возможность однако в реляционных базах свободы выбора почти нет и приходится использовать то, что есть, например, нельзя поколонно хранить данные на разных серверах. При этом в отличие от традиционных СУБД NoSQL дают разработчику больше свободы в

¹ Презентация Emil Eifrem доступна <http://www.infoq.com/presentations/Neo4j-NOSQL-and-Graph-Databases>. Интересно, что он говорит об теоретической изоморфности данных - одна и та же информация может быть представлена в разных моделях - от графовых до ключ/значение. Практическую возможность такого подхода демонстрирует компания InterSystems применяя в Cache принцип Unified Data Model.

использовании естественных особенностей проектной задачи, которые могут быть использованы например для горизонтального масштабирования. Однако разработчик при этом несет больше ответственности за принимаемые решения по архитектуре персистентности данных. Отчасти NoSQL базы можно также сравнивать по наличию механизмов, стандартных для традиционных СУБД или классифицировать их по применяемым инженерным решениям, предлагаемым в качестве альтернативы для традиционных свойств СУБД.

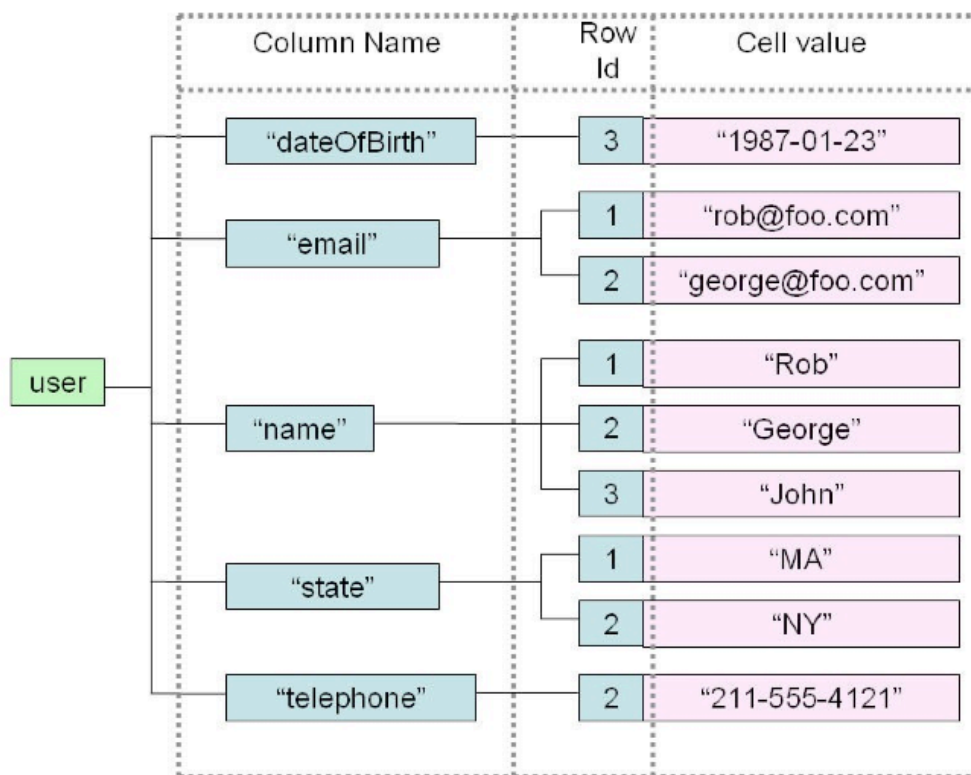
Несмотря на все многообразие NoSQL проектов, сейчас нет ни одного, который бы можно было бы смело назвать универсальной и всеобъемлющей NoSQL платформой - это противоречит и самому принципу специализации, который явно или неявно прослеживается в NoSQL. Поэтому в том случае, если у вас возникла идея воспользоваться NoSQL подходами следующем проекте, то скорее всего вам придется ответить себе на целый ряд вопросов и разрешить множество рисков, например: какую модель данных выбрать; насколько стабильна и зрела выбранная технология; насколько серьезными будут изменения в коде в случае попытки смены NoSQL решения на другую, более эффективную; будет ли язык запросов достаточно полным и технологичным для удовлетворения проектных требований. Отдельно стоит отметить, что многие NoSQL технологии были созданы специально в рамках конкретного проекта и в некоторой степени подобны флюсу - есть вероятность того, что отлично закрывая требования и задачи первоначального проекта они могут не очень подходить в вашем случае.

В ситуации первого проекта с использованием NoSQL подхода мудрым решением стал бы гибридный подход к построению подсистемы управления хранимыми данными. Для гибридного подхода можно предложить два возможных дизайна - одновременное использование в проекте как NoSQL технологии, так и обычной СУБД или использование технологии, которая в необходимой степени поддерживает концепции обоих миров. И в этом случае InterSystems Caché дает уникальную возможность предоставить такую гибридную технологическую платформу - зрелую, проверенную, поддерживаемую.

Первое, очевидное, фонетическое, сходство, которое сразу привлекает внимание в сравнении NoSQL и InterSystems Caché это нереляционность. В основе Caché лежит реализация более простой, чем реляционная, модели, называемой по имени своих атомарных элементов глобалами (или если быть точным полное название Global Persistent Variables или просто globals). Глобалы не имеют схемы, допускают динамическое добавление столбцов, используют разреженное хранение значений столбцов. На уровне глобалов можно использовать при желании блокировки, транзакции, делать распределенное хранение и партиционирование. Жертвуя некоторой неточностью в определении можно думать о глобалах как о структуре, схожей с ассоциативным массивом в PHP или HashMap в Java.

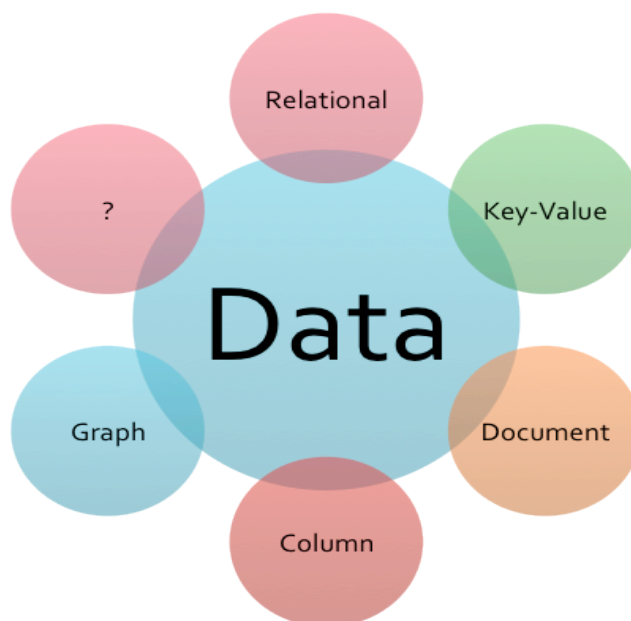
Глобалы как простая и гибкая модель данных предоставляют превосходный базис для построения нереляционных моделей, которые используются в NoSQL: ключ-значение, расширяемые записи, column-based, графы. Подробные примеры реализации популярных в NoSQL моделей приводятся в статье A Universal NoSQL Engine, Using a Tried and Tested Technology (<http://www.mgateway.com/docs/universalNoSQL.pdf>²) - авторы предлагают решения для четырех типов моделей данных.

² Русский перевод статьи можно найти по адресу <http://bloggerator.ru/page/nosql-vvedenie-v-teoriju-bd>



↑ Реализация построчного хранения (из статьи *A Universal NoSQL Engine, Using a Tried and Tested Technology*)

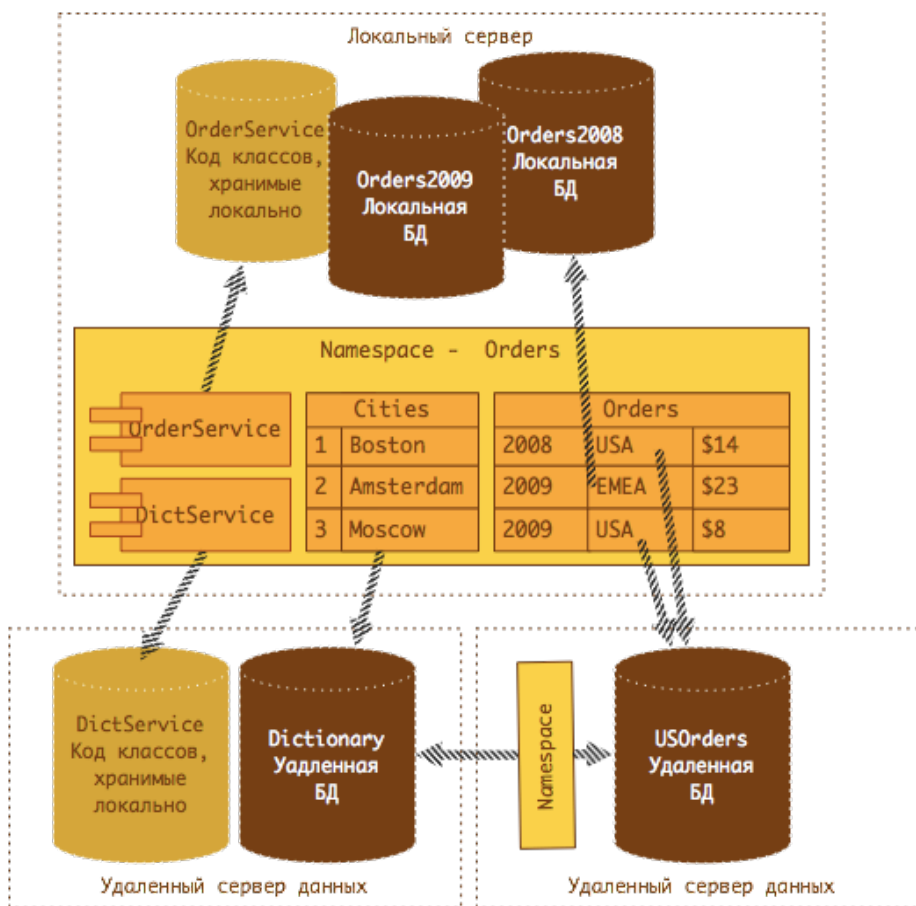
На уровне глобалов не существует привычного для реляционных баз данных декларативного языка запросов. Запросы определяются алгоритмическим способом - выполнение запроса сводится к исполнению кода, написанного на языке Caché Object Script, который предоставляет достаточный набор простых, эффективных операций для работы с данными, хранимыми в глобалах. Уникальность Caché Object Script как языка программирования в том, что это пожалуй единственный язык, в синтаксисе которого явно введена конструкция для указания того, где хранится переменная - в памяти или грубо говоря на диске. Представьте, что в таких традиционных платформах как Java или .NET была бы такая возможность - во многом проблемы с преодолением среды между программой и базой данных просто бы не было. Отсутствие такой конструкции для универсальных языков программирования после работы с Caché кажется странным - ведь естественно предположить, что код работает не только с переменными в памяти, но и с хранимыми переменными. При этом вам не надо заранее определять структуры в базе данных - вы просто работаете с ними так же как с переменными в языках с нестрогой типизацией.



↑ Концепция *Unified Data Model* основана на принципе “данные одни - моделей представления много”

Вслед за InterSystems, которая уже реализовала на основе глобалов объектно-ориентированный и реляционный (SQL) доступы вы способны реализовать свою собственную, уникальную модель данных и так же, как уже готовые к использованию в Caché модели, использовать принцип *Unified Data Model* - подход к управлению персистентностью, который предполагает работу с одними и теми же данными в разных моделях в зависимости от удобства их использования в контексте конкретной задачи. К примеру, для быстрой вставки и чтения возможно использование *key-value* модели, а для запросов используются возможности реляционной модели. При построении своего языка запросов для NoSQL-решения можно использовать Caché Object Script, который предоставляет набор простых операций для работы с данными, хранимыми в глобалах.

Отдельный, но не столь очевидный как нереляционность аспект сравнения Caché и NoSQL это распределенность и масштабирование. Если сравнивать Caché в такой важной для NoSQL категории, как обеспечение горизонтального масштабирования и распределенного хранения с использованием таких механизмов как шардинг и партиционирование, то с одной стороны Caché не имеет готовых *out of the box* вариантов с такими названиями. С другой точки зрения это не совсем так, потому что в Caché либо это делается немного иначе, либо опять же, так же как и в случае с глобалами, предоставляется надежные базовые технологии для эффективного обеспечения таких возможностей. С использованием ECP, концепции областей (*namespace*), *Subscript Level Mapping* возможна реализация эффективной распределенной обработки данных.



↑ *Партиционирование и распределенное хранение с помощью ECP и SLM*

Понимая, что NoSQL привлекает сейчас внимание многих разработчиков, InterSystems выпустила бесплатную СУБД, которая получила название InterSystems Globals (<http://globalsdb.org>). Цель выпуска Globals - познакомить разработчиков с технологией, которая является сердцем Caché и расширить круг разработчиков и архитекторов, которые знают как применять ее.

Globals как многие другие NoSQL проекты предполагает свободное использование для разработки и распространения. Нереляционные модели могут быть реализованы в Globals и в качестве примера такой реализации создан проект с открытым исходным кодом Globals Document Store (GDS) API. Globals может с успехом использоваться в проектах, где требуется высокая скорость и производительность, порядок скорости работы Globals - десятки и сотни тысяч записей в секунду).

InterSystems Globals предоставляет простые API для работы из Java и Node.js (механизм доступа из .Net находится в процессе разработки). В отличие от описанной выше истории с Caché Object Script в случае Globals используется другой подход - операции по работе с глобалами доступны из внешнего по отношению к СУБД языка программирования. При этом процесс работающего с Globals приложения (например JVM) становится фактически одним из процессов СУБД.

В случае с Java технология позволяет быстро реализовывать свои собственные структуры хранимых данных, которые при этом естественны для языка. Например можно быстро реализовать аналог HashMap, данные которого будут храниться в

СУБД. При таком подходе также как и в случае Caché Objects Script различия между переменной в памяти и на диске начинают исчезать.

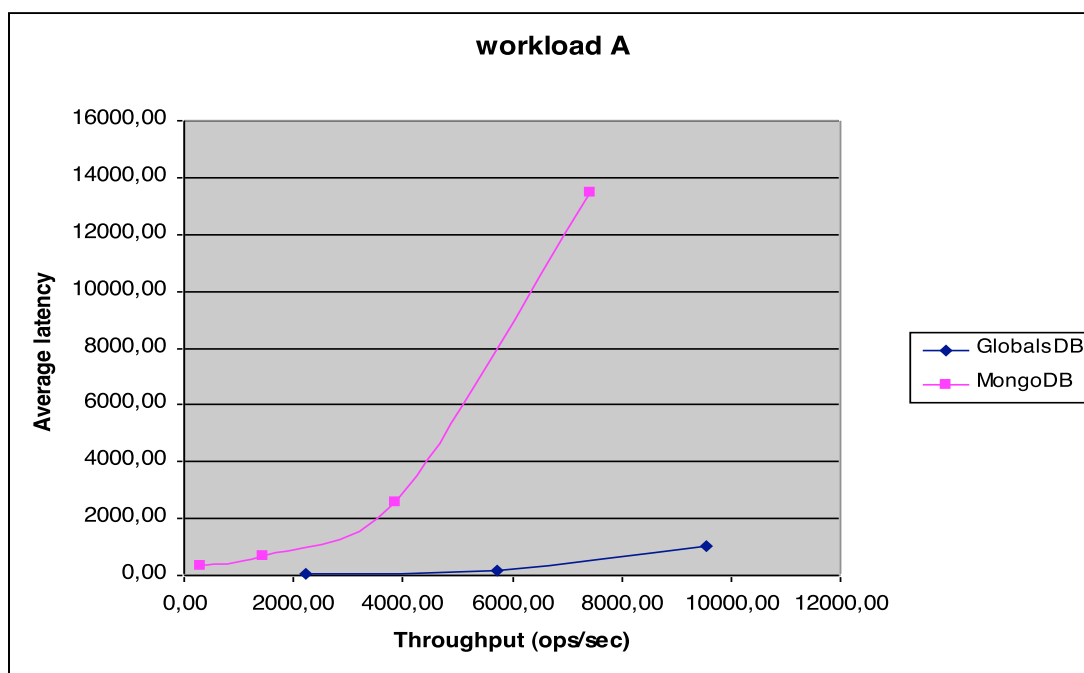
Для Node.js доступ к Globals сразу предоставляет возможность работать с естественными для javascript типами данных - массивы и объекты javascript можно без дополнительных накладных расходов на разработку сразу сохранять, читать и изменять, что на порядок упрощает проблему персистентности данных при работе в javascript. В добавок к этому Globals в связке с Node.js дает высокую скорость работы - для сравнения Globals по тестам работает быстрее³, чем Redis (один из самых широкоиспользуемых проектов NoSQL в том числе известный своей скоростью).

```
myData.open({path: pathToGlobalsMGR, namespace: namespace});  
  
a[n ++] = {global: "Contact", subscripts: [1], data: "Michael Pantaleo"};  
a[n ++] = {global: "Contact", subscripts: [1, "address", 1], data: "San Diego, CA"};  
  
var result = myData.update(a, "array");
```

↑ Пример кода для Node.js при работе с InterSystems Globals. Сохранение массива javascript в глобал

InterSystems Globals позиционируется как NoSQL база данных, но при этом отличается от основного потока NoSQL в нескольких аспектах: в InterSystems Globals нет ограничений на конкретную модель данных, в отличие от многих других решений вы можете использовать блокировки и транзакции, Globals обеспечивает одновременно и эффективную работу с данными в памяти и целостность данных на диске. В настоящий момент в Globals нет возможности по распределенной работе с хранимыми данными. Globals используют стабильную базовую технологию, которая будет гарантированно развиваться и поддерживаться.

³ Тестирование произвoдила команда студентов МИФИ в сентябре 2011. Выполнялась одна из группы тестов YCSB (Yahoo! Cloud Service Benchmark) - GlobalsDB вместе с MongoDB, Cassandra 7.0 и Hbase. Linux x64, 2x Xeon E5504, 2ГЦРАМ 8 ГБ, SATA II 500 ГБ



↑ Итоги независимого тестирования Globals vs. Mongo. При росте операций видно, как возникают задержки в обслуживании у Mongo. При этом время отклика для Globals растет слабо

Globals, в отличие от Caché, предоставляет ядро по работе с глобалами, без объектного и реляционного доступов. Но при развитии проекта всегда есть возможность перейти на Caché без изменения кода приложения - Globals API является подмножеством технологии Caché Extreme.

Подводя итог можно сказать что сейчас можно думать о Caché как о NoSQL базе данных и больше - универсальной, стабильной платформе для NoSQL проектов с поддержкой объектной и реляционной моделей не уступающей по своим эксплуатационным качествам традиционным СУБД. И несмотря на то, что NoSQL - термин, ставший популярным достаточно недавно вполне соответствует ценностям компании, которые остаются прежними - так же, как и на протяжении 30 лет, технология, которая лежит в основе Caché позволяет вам быстро создавать решения, которые полностью специфику именно вашего проекта. Зачем вам технология, полученная в рамках другого проекта, если вы можете использовать свою?

Санкт-Петербург
16.12.2011