Особенности управления транзакциями в Spring Data Solr

М.М. Гумеров
WaveAccess
Санкт-Петербург, Россия
e-mail: mgumerov@gmail.com

В.А. Подоприго
Технология решений
Витебск, Республика Беларусь
e-mail: viktor.podoprigo@tehnologia.by

Аннотация[1]

Apache Solr – по некоторым данным, наиболее популярная платформа полнотекстового поиска, предоставляющая широкий спектр возможностей. Сервер Solr обслуживает запросы по протоколу JSON или обычному HTTP. У приложений, исполняемых в виртуальной машине Java, есть возможность работать с Solr через интерфейс, адаптированный к проекту Spring Data - интерфейс репозитария. Эту адаптацию реализует Spring Data Solr. Предполагается, что репозитарии Solr участвуют в транзакциях JTA, обеспечивая утверждение или отмену внесенных в рамках транзакции изменений при завершении или отмене транзакции. Однако при практическом использовании можно обнаружить, что это поведение реализовано не в полном объеме.

1. Введение

Apache Solr может применяться для хранения и осуществления полнотекстового поиска в текстовых документах, документах XML, а также некоторых других видов. Пользоваться этими возможностями можно в различных целях: так, операторы могут задавать поисковые запросы серверу solr с использованием веб-браузера, или же клиентами сервера Solr могут быть программы для ЭВМ.

Spring Data – часть широко применяемого в различных проектах на языке программирования Java (а в принципе, и других, работающих под управлением виртуальной машины Java) набора библиотек Spring, задачей которой является предоставление удобного и в какой-то степени унифицированного способа работы с хранилищами данных различной природы.

За возможность работать с хранилищем документов Solr средствами Spring Data отвечает проект Spring Data Solr.

Труды третьей международной конференции "Интеллектуальные технологии обработки информации и управления", 10 - 12 ноября, Уфа, Россия, 2015

2. Заявленная поддержка транзакций

Как правило, приложения, вносящие изменения в базы данных, проводят эти изменения пакетами в рамках различных транзакций, что облегчает обеспечение согласованности хранящихся в БД данных.

Spring предоставляет приложениям поддержку транзакций со стороны Java, обеспечивая, например, автоматический запуск транзакций при запуске методов в коде приложения и их фиксацию в БД при успешном завершении работы этих методов.

Согласно разделу 5.1.5 документации по Spring Data Solr [1], Solr поддерживает транзакции на уровне сервера (не на уровне клиентского приложения), и за счет этого вносимые изменения ставятся на сервере в очередь, и фиксируются или отменяются все сразу (что именно означает «сразу» – не уточняется). Также утверждается, что репозитарии Spring Data Solr будут принимать участие в транзакциях Spring и фиксировать или отменять изменения при их завершении.

В этих утверждениях ничего не говорится об атомарности, изоляции и других ACID-свойствах, ради которых вообще введено понятие транзакции. К примеру, если вся очередь изменений вносится в Solr вместе, это еще не значит, что другие работающие с Solr процессы не смогут в какой-то момент обозревать хранилище Solr в таком состоянии, когда в него попала лишь половина этих изменений.

Помимо этой недосказанности, на практике авторы столкнулись с ошибкой, которая привела их к пониманию того, что понимание участия в транзакциях в этих утверждениях совсем не такое, как подразумевается в Spring.

3. Пример из практики

Разрабатываемый авторами программный продукт, использующий для индексации и быстрого поиска обрабатываемых им документов Solr, в транзакции Spring удалял документ из БД и затем в этой же транзакции – из индекса Solr.

В процессе разработки и тестирования выяснилось, что в одной специфической ситуации документ в Solr оставался даже после утверждения транзакции Spring. Оказалось, что на сервер Solr не уходила команда commit. В ходе анализа исходных кодов Spring Data Solr выяснилось, что если для внесения изменений в

Solr применяется SolrRepository, он участвует в транзакции Spring, а если изменения вносятся с помощью интерфейса SolrOperations (тоже из состава Spring Data Solr), то они минуют транзакцию. В документации этот факт, насколько авторам известно, не отражен.

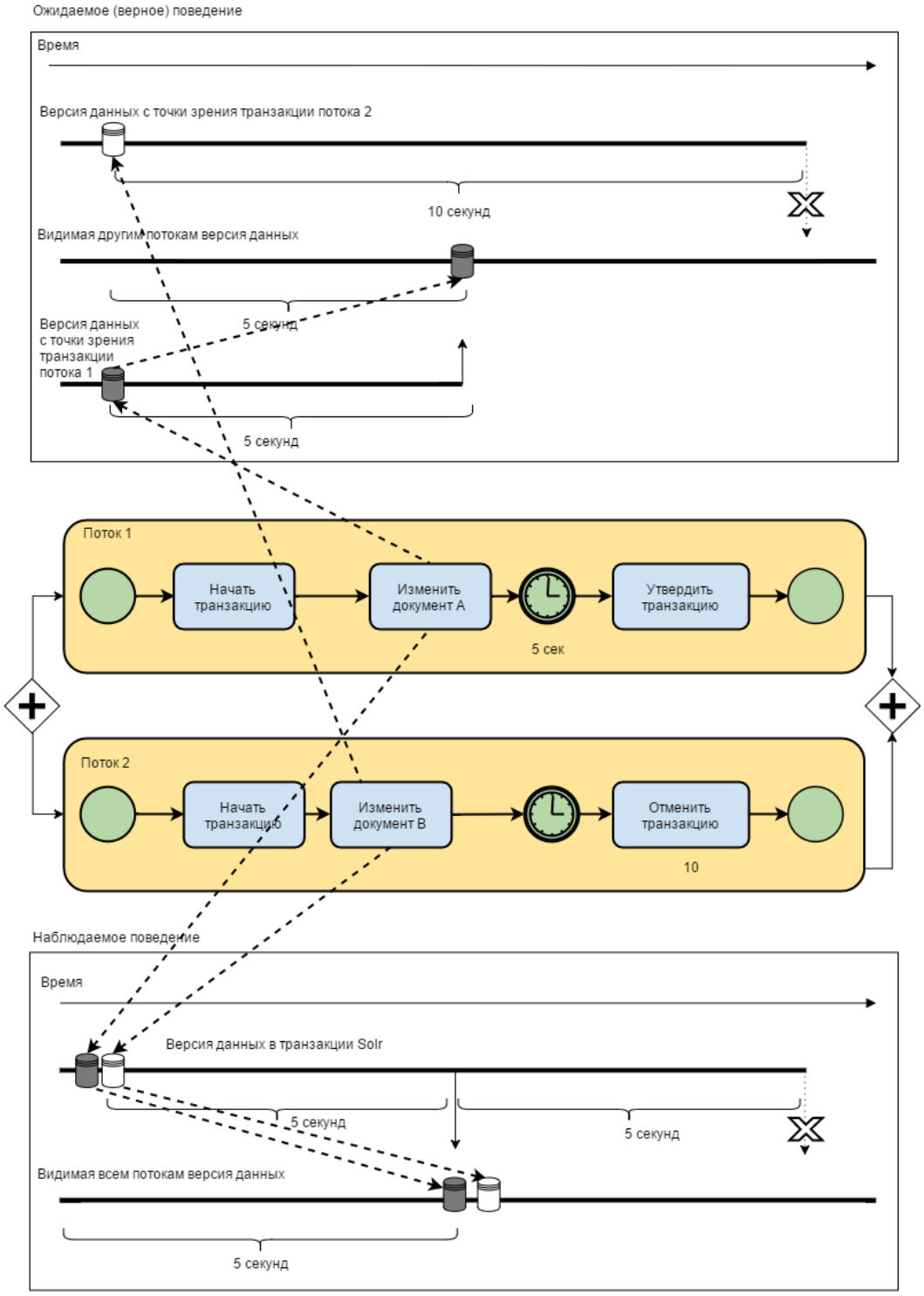

Рис. 1. Схема вычислительного эксперимента (см. раздел 4)

То есть, как минимум, хотя solr-репозитарий участвует в транзакциях Spring, к SolrOperations это не относится. Но что же означает «минуют транзакцию»? Это означает, что изменения отправляются сразу на сервер Solr, но в таком случае можно ожидать, что запрошенное удаление из Solr произойдет еще до окончания транзакции, а оно не происходило даже по ее окочании. Дело в том, что хотя изменение выполняется вне тразнакции Spring, оно тем не менее выполняется в некой транзакции Solr, и до утверждения (commit) этой транзакции оно не видимо тем, кто читает индекс Solr.

Далее авторы задались вопросом, почему же в большинстве случаев эта ошибка не проявлялась? А также – почему если после описанной операции

удаления документа происходила – в другой транзакции – еще одна операция, тоже вносящая свои изменения в Solr, документ из индекса наконец удалялся. Эта вторая операция отличалась от первой в части работы с Solr главным образом тем, что изменения вносились через SolrRepository, а значит, по завершении транзакции на сервер Solr отправлялась команда commit. Но ведь commit, отправленный второй транзакцией, никак не должен был утвердить изменения, внесенные первой!

У авторов возникло предположение, что фактически сервер solr поддерживает единственную «транзакцию» для каждого репозитария, и в силу этого в описанной ситуации первый набор изменений, прошедший в обход первой транзакции Spring, ставится в эту очередь, затем вторая транзакция в эту же очередь ставит свои изменения, а по окончании посылает на сервер commit, который утверждает все находящиеся в очереди изменения – в т.ч. и прошедшие мимо первой транзакции.

4. Проверка гипотезы

Для проверки этого предположения был написан следующий тест. Запускались два потока, каждый начинал собственную транзакцию, каждый вносил свой набор изменений в Solr через SolrRepository, и один поток после выполнения своих изменений ждал 5 секунд и делал утверждение транзакции (речь о транзакции spring), а второй ждал 10 секунд и затем делал отмену (rollback) своей транзакции (см. рис. 1)

После выполнения этого теста в индексе Solr были видны изменения, внесенные той транзакцией, которая была отменена! То есть, хотя формально репозитарий spring data solr поучаствовал в транзакции и отправил на сервер команду rollback при отмене транзакции, поведение сервера не соответствовало семантике транзакций Spring: он при получении commit внес все изменения, заданные обеими транзакциями к этому моменту (конечно же, за первые 5 секунд – пока один из потоков не дошел до выполнения commit –оба потока успели свои изменения в своих транзакциях выполнить), а не заданные только той, которой был вызван commit.

Более того, даже если транзакции вносили изменения в разные репозитарии, наблюдалась такая же картина.

5. Дальнейшие изыскания

Как оказалось, один из других источников информации о Solr (самом Solr, не Spring Solr Data) содержит утверждение, которое могло бы подтолкнуть читателя к этим выводам: “The most appropriate way to update solr is with a single process in order to avoid race conditions when using commit and rollback” [2]. Вот только «single process», как выясняется, это еще преуменьшение, и на самом деле речь вообще идет о модели, в которой исключено одновременное внесение различных изменений в документы Solr.

В настоящее время авторы еще не пришли к решению о том, как наиболее эффективно справиться с этой проблемой, учитывая, что 1) вносимые транзакцией в Solr изменения должны оказаться отражены в индексе не через неопределенное время, а, условно, сразу же, чтобы пользовательский интерфейс мог после сразу окончания транзакции перечитать данные из Solr и эти изменения «увидеть», 2) приложение выполняет операции в ответ на поступающие по протоколу HTTP запросы от множества клиентов, и непродуктивно было бы запрещать одновременную обработку различных запросов разными потоками, 3) в идеале, внесенные изменения далее в этой же транзакции должны быть видимы при опросе Solr, хотя они в Solr еще не утверждены.

Предполагалось, что одним из возможных решений может стать работа непосредственно с Apache Lucene, который используется и в Solr, что позволит преодолеть ограничения, налагаемые Solr на поддержку транзакций. Но краткое изучение документации по Lucene пока не дало уверенности в том, что ограничения наложены Solr, а не самой библиотекой Lucene. Так, известно, что экземпляру IndexWriter разрешается накапливать набор изменений в индексе, невидимый всем, кто читает индекс (впрочем, есть и способ их увидеть), а затем атомарно утверждать или отменять эти изменения. Но о том, могут ли несколько IndexWriter существовать одновременно и вносить изменения, и будут ли их изменения изолированы друг от друга, либо вовсе умалчивается, либо авторам при первоначальном анализе не удалось найти эти указания.

Список используемых источников

1. Strobl C., Gierke O., Pollack M., Risberg Th. «Spring Data Solr» / [Электронный ресурс]. – Режим доступа: http://docs.spring.io/spring-data/solr/docs/current/reference
2. «Solr Wiki» / [Электронный ресурс]. – Режим доступа: https://wiki.apache.org/solr/Solrj