Аннотация

Jepsen — это библиотека Clojure, предназначенная для проверки корректности работы баз данных, очередей, систем распределенной координации. Начиная с 2013 года с помощью Jepsen было протестировано более двух десятков систем - были обнаружены расхождения в репликах, потери данных, конфликты блокировок, чтения устаревших данных. В данной работе возможности этого инструмента применяются для определения уровней изолированности транзакций MySQL кластера.

Содержание

1 Введение

Уровень изоляции транзакций определяет в какой мере в результате выполнения логически параллельных транзакций в СУБД допускается получение несогласованных данных. В упрощённом варианте набор уровней изоляции может быть представлен как шкала, которая содержит ряд значений, проранжированных от наинизшего до наивысшего. Более низкие уровни изоляции позволяют производить большее количество параллельных транзакций, но при этом возрастает риск работы транзакций с некорректными данными. Для описания более широкого набора уровней может использоваться дерево, т.к. некоторые уровни могут быть несравнимы между собой. Разные уровни изоляции для параллельных транзакций позволяют разработчикам приложений находить компромис между пропускной способностью системы и согласованностью хранящихся данных.

1.1 Спецификация ANSI/ISO SQL-92

В стандарте **ANSI/ISO SQL-92** определяются четыре уровня изоляции транзакций:

- **READ UNCOMMITTED**
- **READ COMMITTED**
- **REPEATABLE READ**
- **SERIALIZABLE**

Определения уровней вводятся через три запрещённые последовательности действий, называемые аномалиями:

- **Dirty Read**
- **Non-repeatable Read**

- **Phantom**

Данная система уровеней изоляции была подвергнута критике. В частости в статье **”A Critique of ANSI SQL Isolation Levels”** [1] было показано, что введенные стандартом определения, как аномалий, так и уровней, недостаточно точны и вызывают разночтения. Также было отмечено, что стандарт не определяет ряд других используемых на практике уровней изоляции.

Несмотря на это, данные термины до сих пор широко используются современными базами данных как предоставляемые уровни изоляции. Например, база данных PostgreSQL предоставляет три уровня - без возможности выбрать **READ UNCOMMITTED**. База данных MySQL server позволяет выбирать любой из четырех уровней, MySQL cluster ограничен возможностями NDBCLUSTER engine - поддерживается только режим **READ COMMITTED**.

2 Уровни изоляции

2.1 Опрдедения по спецификации и их критика

Рассмотрим определения аномалий, использующиеся в стандарте **ANSI/ISO SQL-92**. Также рассмотрим их скорректированные варианты, приведённые в статье **”A Critique of ANSI SQL Isolation Levels”** [1]. Использующийся в определениях термин ”элемент данных” имеет широкий спектр значений. Это может быть строка в таблице, набор строк, целая таблица или сообщение в очереди.

Перед использованием определений из статьи нужно ввести специальную нотацию. Истории, состоящие из чтения, записи, фиксирования (commits) и отмены (aborts) изменений могут быть записаны в сокращённом виде.

1. “w1[x]” означает запись транзакцией 1 в элемент данных x
2. “r2[x]” означает чтение транзакцией 2 элемента данных x

3. Транзакиция, использующая при чтении или записи фильтрацию по предикату P, обозначается как “r1[P]” и “w1[P]” соответственно

4. Фиксирование (commits) и отмена (aborts) изменений обозначаются как “c1” и “a1” соответственно

Заметим также - уточнающие определения отличаются от оригинальных тем, что они описывают более широкий набор историй. Происходит это за счёт того, что определения из спецификации явно описывают как аномалии проявляются, указывая дополнительные операции в транзакциях (коммиты и аборты). Определения из статьи, напротив, эти детали опускают.

2.1.1 Dirty write (P0)

В стандарте **ANSI/ISO SQL-92** не приводится определение термина **dirty write**, более того, данная аномалия не используется при определении четырёх ”классических” уровенй изоляции. Поэтому текстовое определение аномалии приводится из статьи, а не стандарта. Замечание по этому упущению было также приведено в статье [1].

Определение из статьи: Транзакция T1 модифицирует элемент данных. Затем другая транзакция T2 модифицирует этот элемент данных до того, как T1 фиксирует свои изменения или отменяет их. В случае, если T1 или T2 производят отмену изменений, не ясно какое значение должен принять использованный элемент данных.

Уточнённое определение:

```
P0: w1[x]...w2[x]...
    ((c1 or a1) and (c2 or a2) in any order)
```

2.1.2 Dirty read (P1)

Определение из стандарта: Транзакция T1 модифицирует элемент данных. Затем другая транзакция T2 читает этот элемент данных до того, как T1 фиксирует или отменяет изменения. В случае, если T1 производит отмену изменений, T2 прочитала изменения, которые были отменены.

Уточнённое определение:

```
P1: w1[x]...r2[x]...
        ((c1 or a1) and (c2 or a2) in any order)
```

2.1.3 Non-repeatable read (P2)

Определение из стандарта: Транзакция T1 читает элемент данных. Затем другая транзакция T2 модифицирует или удаляет данный элемент данных и фиксирует изменения. В случае, если транзакция T1 снова читает этот элемент данных, то она наблюдает произведённые изменения.

Уточнённое определение:

```
P2: r1[x]...w2[x]...
        ((c1 or a1) and (c2 or a2) in any order)
```

2.1.4 Phantom (P3)

Определение из стандарта: Транзакция T1 читает набор элементов данных, удовлетворяющих заданному предикату. Затем транзакция T2 создает элементы данных, которые удовлетворяют заданному предикату и фиксирует изменения. В случае, если T1 снова читает набор элементов данных, удовлетворяющих заданному предикату, то она наблюдает набор отличный от полученного при первом чтении.

Уточнённое определение:

```
P3: r1[P]...w2[y in P]...
        ((c1 or a1) and (c2 or a2) any order)
```

Определения классических уровней изоляции вводится через приведённые аномалии:

Table 3. ANSI SQL Isolation Levels Defined in terms of the four phenomena				
Isolation Level	**P0 Dirty Write**	**P1 Dirty Read**	**P2 Fuzzy Read**	**P3 Phantom**
READ UNCOMMITTED	Not Possible	Possible	Possible	Possible
READ COMMITTED	Not Possible	Not Possible	Possible	Possible
REPEATABLE READ	Not Possible	Not Possible	Not Possible	Possible
SERIALIZABLE	Not Possible	Not Possible	Not Possible	Not Possible

Рис. 1: Таблица уровней

2.2 Определения в G-нотации

Рассмотрим также определения в терминах графа прямой сериализации, построенного по истории транзакций (Direct Serialization Graph, $DSG(H)$). Именно эти термины будут использоваться при тестировании системы. Данные определения приведены из [2], [3] и [4].

2.2.1 Граф прямой сериализации

Определение: Графом прямой сериализации, построенным на истории H, называется граф, каждый узел которого соответствует зафиксированной транзакции и направленные рёбра которого соответствуют различным типам прямых конфликтов (зависимостей) между ними. Обозначается как $DSG(H)$.

Типы прямых конфликтов между транзакциями:

Read-dependency: Транзакция T2 имеет read-dependency к T1, если T1 производит запись версии i элемента данных x, и T2 читает x_i. Обозначается как **wr** (write-read).

Write-dependency: Транзакция T2 имеет write-dependency к T1, если T1 производит запись версии i элемента данных x, и T2 производит запись следующей после x_i версии элемента данных. Обозначается как **ww** (write-write).

Anti-dependency: Транзакция T2 имеет anti-dependency к T1, если T1 производит чтение элемента данных x, и T2 производит последующие обновление x_i. Обозначается как **rw** (read-write).

2.2.2 G0: Write cycle

G0 (Write Cycle): История H содержит аномалию G0, если $DSG(H)$ содержит направленный цикл, полностью состоящий из write-dependency рёбер (**ww**).

Уровень изоляции на котором вводится запрет на G0 обозначается PL-1 и называется READ UNCOMMITTED.

2.2.3 G1: Dirty read

Аномалия G1 (Dirty Read) отражает суть грязных чтений и состоит из G1a, G1b, G1c.

Уровень изоляции на котором вводится запрет на G1 (включая G1a, G1b, G1c) обозначается PL-2 и называется READ COMMITTED.

G1a (Aborted read): История H содержит аномалию G1a, если она содержит отменёную транзакцию T1 и зафиксированную транзакцую T2, и при этом T2 прочитала некоторый объект (возможно, через предикат), измененный T1.
G1b (Intermediate reads): История H содержит аномалию G1b, если она содержит зафиксированную транзакцию T2, которая прочитала версию элемента данных x (возможно, через предикат), записанную транзакцией T1, и при этом данная версия не была окончательной модификацией x в T1.
G1c (Circular information flow): История H демонстрирует аномалию G1c, если $DSG(H)$ содержит ориентированный цикл, состоящий исключительно из ребер типа read-dependency и write-dependency (**wr** и **ww**).

2.2.4 G-cursor: Lost update

G-cursor(x) (Lost Update): История H демонстрирует аномалию G-cursor(x), если $LDSG(H)$ (Labeled DSG, рёбра помечены элементами данных, с которыми они работают) содержит цикл с anti-dependency (**rw**) и одним или несколькими ребрами write-dependency (**ww**), и при этом все рёбра цикла помечены одним элементом данных x.

Уровень изоляции на котором вводится запрет на G-cursor обозначается PL-CS и называется Cursor Stability.

2.2.5 G-single: Read skew

G-single (Read Skew): История H демонстрирует аномалию G-single, если DSG(H) содержит цикл, состоящий из одного ребра anti-dependency (**rw**) и одного или нескольких ребер двух других типов зависимостей (**ww**, **wr**).

Уровень изоляции на котором вводится запрет на G-cursor обозначается PL-2+ и называется Consistent View.

2.2.6 G2-item: Write skew on disjoint read

G2-item (Item Anti-dependency Cycles): История H демонстрирует аномалию G2-item, если $DSG(H)$ содержит направленный цикл, имеющий одно или несколько ребер anti-dependency на элементе данных (не по предикату) (**rw**).

Уровень изоляции на котором вводится запрет на G2-item обозначается PL-2.99 и называется REPEATABLE READ.

2.2.7 G2: Write skew on predicate read

G2 (Write skew on predicate read): Определяется аналогично G2-item с поправкой - разрешены anti-dependency рёбра на предикатах.

Уровень изоляции на котором вводится запрет на G2-item обозначается PL-3 и называется SERIALIZABLE.

2.2.8 OTV: Observed transaction vanishes

OTV (Observed transaction vanishes): История H демонстрирует аномалию OTV, если $USG(H)$ (Unfolded Serialization Graph [[2]]) содержит направленный цикл, состоящий ровно из одного ребра read-dependency (**wr**) по x из T1 в T2 и множество ребер по y, содержащее хотя бы одно ребро антизависимости (**rw**) из T2 в T1, и чтение в T2 по y предшествует чтению T2 из x.

Уровень изоляции на котором вводится запрет на OTV называется Monotonic Atomic View (без специального обозначения).

2.3 Дерево уровней

Уточнение определений из **ANSI/ISO SQL-92**, а также введение дополнительных аномалий и уровней изоляций заметно увеличивает их иерархию. Определив отношение порядка на уровнях изоляции приведём их сокращённое дерево (более полная иерархия приведена на рис.6 и рис.7 при описании инструмента Elle).

Определение: Уровень изоляции L1 слабее, чем уровень изоляции L2 (или L2 сильнее L1), если все несериализуемые истории, которые подчиняются критериям L2, также удовлетворяют L1 и есть по крайней мере одна несериализуемая история, которая может происходить на уровне L1, но не на уровне L2.

Определение: Два уровня изоляции L1 и L2 эквивалентны, если множества несериализуемых историй, удовлетворяющих L1 и L2, являются идентичными.

Определение: L1 не сильнее L2, если либо L1 слабее L2, либо L1 и L2 эквивалентны.

Определение: Два уровня изоляции L1 и L2 не сравнимы, если каждый

из двух уровней изоляции допускает несериализуемую историю, которая запрещена другим.

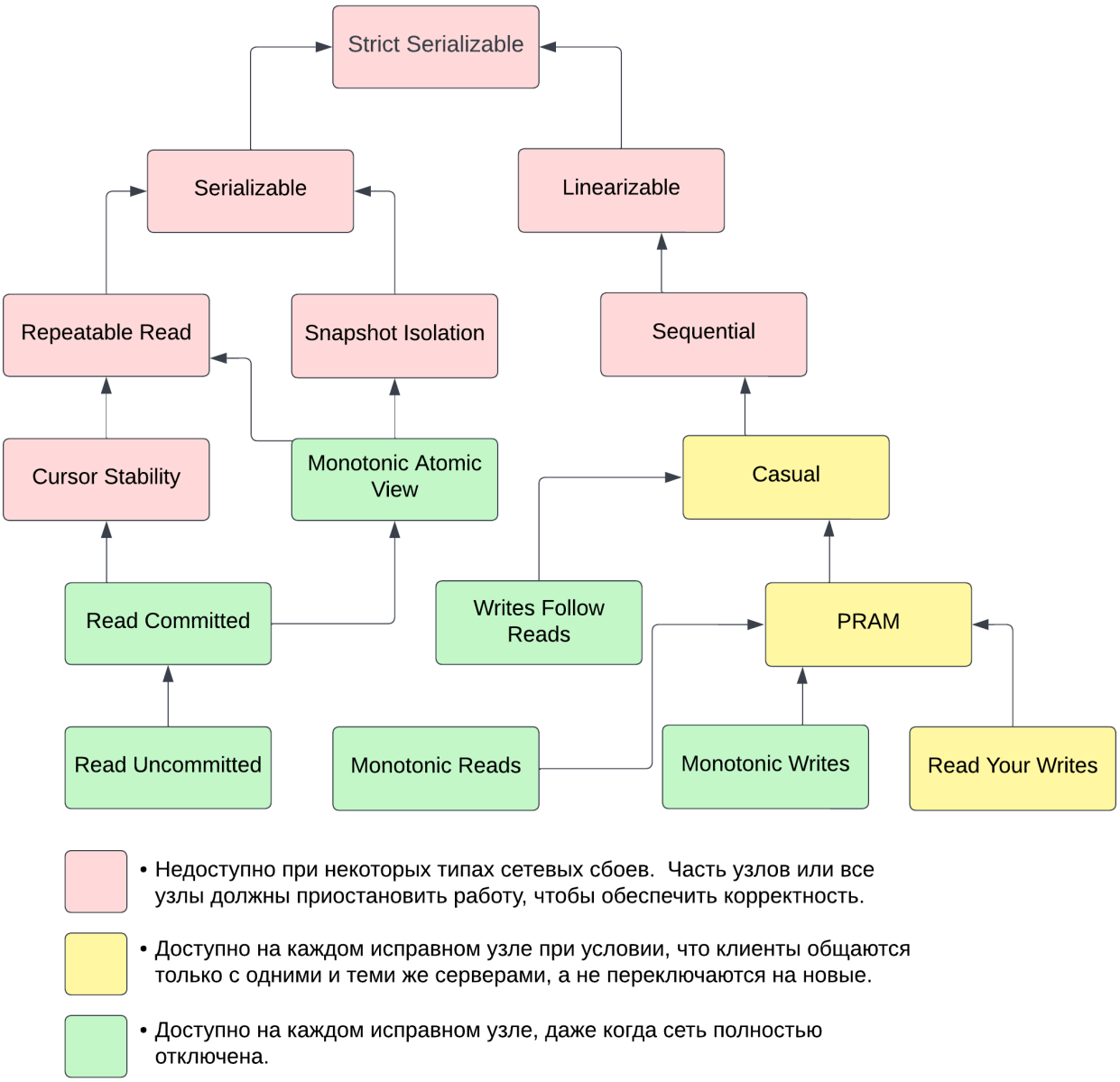

Рис. 2: Common consistency models

3 MySQL Cluster

Распределенная база данных MySQL Cluster — это технология, обеспечивающая shared-nothing (каждый запрос обрабатывается одним улом) кластеризацию с автошардингом для системы управления базами данных MySQL. Она предназначена для обеспечения высокой доступности и высокой про-

пускной способности с малой задержкой, а также почти линейную масштабируемость.

3.1 Архитектура

MySQL Cluster [5] разделяет узлы на три разных типа:

- Узел данных (**ndbd node**): используются для хранения данных. Таблицы автоматически шардируются по таким узлам. Также обрабатывают балансировку нагрузки, репликацию, самовосстановление.
- Узел управления (**ndb_mgmd node**): используется для настройки и мониторинга кластера. Они необходимы только для запуска или перезапуска узла кластера. Также могут быть настроены как арбитры, но это не обязательно (вместо этого узлы приложения могут быть настроены как арбитры).
- Узел приложения или узел SQL (**mysqld node**): сервер MySQL, который подключается ко всем узлам данных для выполнения сохранения и извлечения данных. Этот тип узла является необязательным; можно запрашивать узлы данных напрямую через NDB API, либо используя C++ API, либо один из дополнительных NoSQL API.

Как правило, ожидается, что каждый узел будет работать на отдельной физической или виртуальной машине. Однако, часто узлы управления размещаются совместно с узлами SQL. В случае, если узлы размещаются на одном хосте, следует помнить, что это может создать единую точку отказа.

3.2 Конфигурация

Первой компонентой, которая запускается в любом MySQL кластере является узел управления. Для настройки узла управления используется файл конфигурации **config.ini** - корректная конфигурация обеспечит правильную

синхронизацию и распределение нагрузки между узлами данных. С помощью этого файла мы указываем количество реплик, адреса узлов хранения и узла SQL. Также на узле управления должны быть разрешены входящие подключения от других узлов кластера.

Следующим шагом производится настройка узлов хранения. Узлы хранения данных извлекают свою конфигурацию из файла **my.cnf**. В минимальном варианте в нем достаточно указать адрес узла управления.

Настройка узла SQL также производится с помощью конфигурационного файла **my.cnf**. В нём указывается адресс узла управления, а также storage engine используемыq в MySQL кластере - NDB engine.

3.3 Автоматизация запуска кластера в Jepsen

В библиотеке Jepsen для автоматизации установки, подготовки и удаления исследуемой системы используется протокол **jepsen.db/DB**. Возможны два варианта по подготовке тестируемой системы.

Во-первых, можно подготовить исследуемое ПО отдельно от тестов: например, в случае MySQL кластера, вручную подготовить конфигурационные фаилы и запустить БД. В качестве реализации **jepsen.db/DB** в таком случае можно использовать заглушку. Такой подход проще, но потребуется следить за очисткой БД после проведения тестирования, чтобы каждый тест запускался на подготовленной системе. Также в таком подходе труднее менять версию исследуемой системы.

Во-вторых, можно имплементировать протокол **jepsen.db/DB** и использовать его для подготовки системы перед тестированием и её очистки после. Этот протокол определяет две функции, которые должны поддерживаться всеми реализациями: **(setup! db test node)** и **(teardown! db test node)**. Функция **teardown!** вызывается в начале и в конце тестирования, функция **setup!** вызывается после первоначального **teardown!** перед тестированием. Такой подход сложнее в реализации, но позволяет запускать тесты без ручного вме-

шательства в систему.

4 Инструмент Jepsen

4.1 Обзор

Jepsen [6] — это библиотека Clojure, предназначенная для проверки корректности работы баз данных, очередей, систем распределенной координации. Начиная с 2013 года с помощью Jepsen было протестировано более двух десятков таких систем - были обнаружены расхождения в репликах, потери данных, конфликты блокировок, чтения устаревших данных.

Jepsen в силу своих особенностей занимает особую нишу среди инструментов проверки корректности систем:

- Тестирование без вмешательства: производится оценка реальных систем, работающих на реальных кластерах. Это позволяет тестировать системы без доступа к их исходному коду и без вмешательства в их внутреннее устройство. Таким образом, ошибки, воспроизводимые в Jepsen, можно наблюдать и в производственной среде, а не только теоретически. Однако такой подход жертвует некоторыми сильными сторонами формальных методов: тесты недетерминированы, и Jepsen не даёт возможности доказать корректность системы, а только находит ошибки.

- Тестирование распределенных систем с использованием нештатных сценариев: потери соединения, рассинхронизированные часы и частичный отказ узлов системы. Такой подход позволяет Jepsen проводить тестирование поведения системы в нестандартных условиях.

- Генеративное тестирование: к системе параллельно применяются случайно сгенерированные операции. История применения этих операций проверяется на коррекность соответствующей моделью. Такой подход

позволяет выявлять пограничные случаи с редко используемыми на практике комбинациями операций.

Тест Jepsen — это программа на Clojure, которая использует библиотеку Jepsen для настройки распределенной системы, запуска набора операций в этой системе и проверки того, что история применения этих операций соответствует гарантиям, которые предоставляются этой системой. Jepsen использовался для проверки широкого набора систем: от eventually-consistent баз данных, до линеаризуемых систем координации и распределенных планировщиков задач. Он также предоставляет возможности по визуализации тестирования: создание графиков производительности и доступности системы, иллюстрация обнаруженных аномалий в транзакциях (с помощью вспомогательного инструмента Elle).

4.2 Структура теста в Jepesn

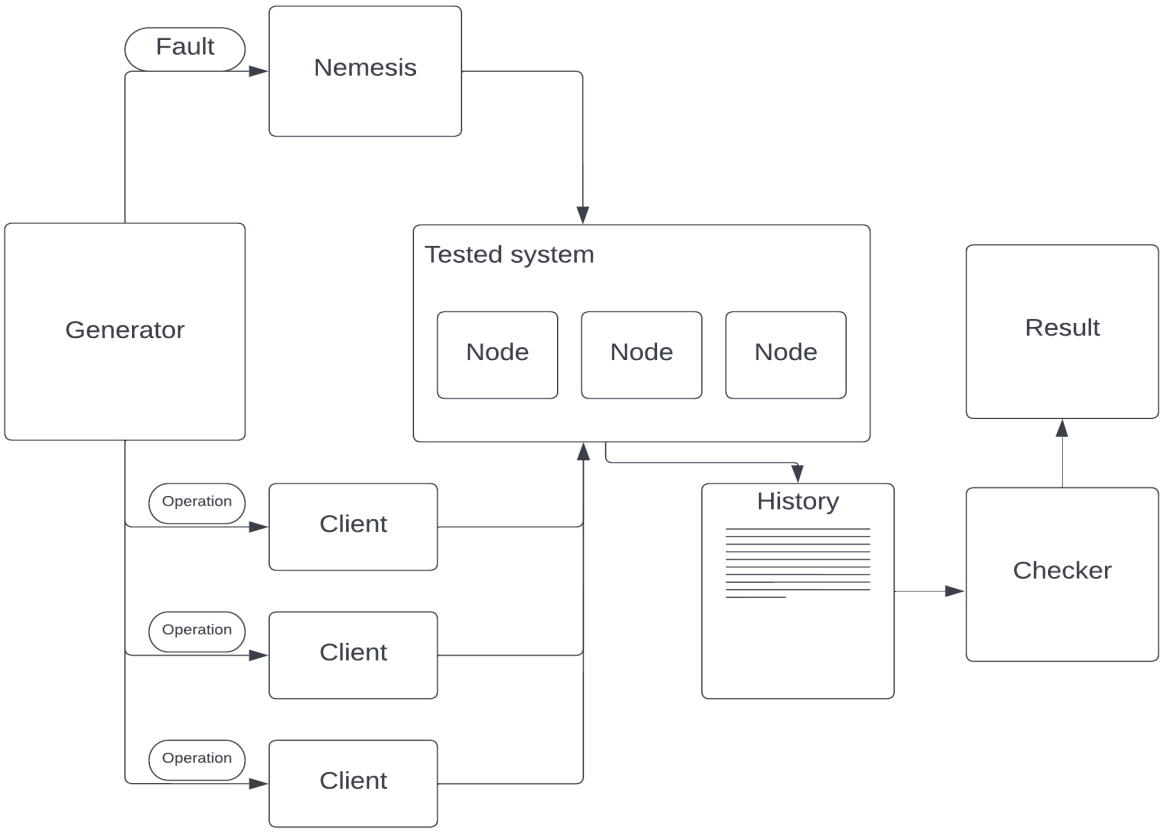

Рис. 3: Структура теста в Jepesn

Тест Jepsen запускается как Clojure программа на управляющем узле. Эта программа использует SSH для входа в группу узлов, где она настраивает распределенную систему, которую предстоит тестировать, используя подготовленные реализации протоколов **jepsen.os/OS** и **jepsen.db/DB**.

Когда система запущена, управляющий узел запускает набор логически однопоточных процессов, каждый из которых имеет свой клиент для работы с распределенной системой. Клиентом является реализация протокола **jepsen.client/Client**, специально подготовленная для исследуемой системы. Генератор производит новые операции для каждого процесса. Затем процессы с помощью клиентов применяют эти операции к системе. Начало и окончание каждой операции записывается в историю. При выполнении операций

специальный процесс nemesis вносит в систему сбои, также запланированные генератором.

Вконце происходит очитска системы. Далее Jepsen использует средство проверки для анализа истории теста на корректность, а также для создания отчетов, графиков и других элементов визуализации. Тест, история, анализ и любые дополнительные результаты записываются в файловую систему для последующего просмотра.

5 Дизайн тестирования

5.1 Операции над системой

Тестовая ифраструктура рандомизированно генерирует транзакции: операции добавления и чтения над набором списков. Операции производятся с экспоненциальной частотой. Каждый список идентифицируется уникальным целочисленным логическим ключом и хранится как строка в одной из нескольких таблиц, которая определяется по хешу ключа. Ключи объектов хранятся в двух полях: primary key ID и secondary key SK, который используется для проверки доступа путем сканирования таблицы. Значение каждого списка хранится в виде столбца TEXT, разделенного запятыми.

В ходе тестирования производятся добавления уникальных целочисленных элементов в списки, идентифицируемые по ключу (через ID или SK), Для этого используется запрос:

```
INSERT ... ON CONFLICT DO UPDATE.
```

Чтения возвращают текущее состояние определенного списка - упорядоченный набор целых чисел для определенного объекта. Производится через

```
SELECT (val) FROM txn0 WHERE id = ?.
```

Тест применяет эти транзакции к БД с помощью драйвера JDBC MySQL Connector/J (версия 8.0.29). Также для общения с БД используется библиотека Clojure seancorfield/next.jdbc версии ”1.0.445”. Репозиторий с тестами [7].

5.2 Elle checker

Анализ полученной истории производится с помощью средства проверки изоляции транзакций Elle. Elle выводит граф зависимости транзакций по полученной в ходе эксперимента истории применения операций и ищет циклы (и нециклические аномалии) в этом графе.

Инструмент позволяет обнаруживать широкий спектр аномалий из определеных в статье Адьи, Лискова и О’Нила **“Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions”** [2], включая **G0 (dirty write)**, **G1a (aborted read)**, **G1b (intermediate read)**, **G1c (cyclic information flow)**, **G1-single (read skew)** и **G2-item (anti-dependency cycle)**. Также проверяется внутренняя согласованность транзакций: проверяется, что транзакции наблюдают значения, согласующиеся с их собственными предыдущими записями.

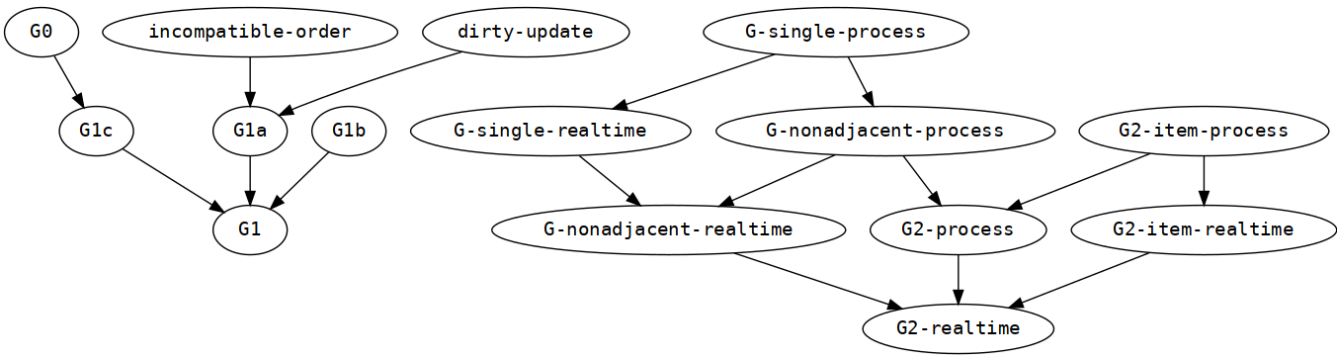

Рис. 4: Дерево аномалий в Elle, часть 1

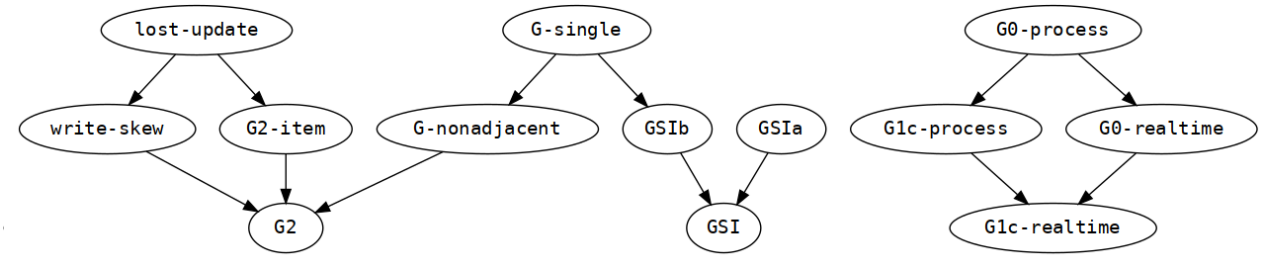

Рис. 5: Дерево аномалий в Elle, часть 2

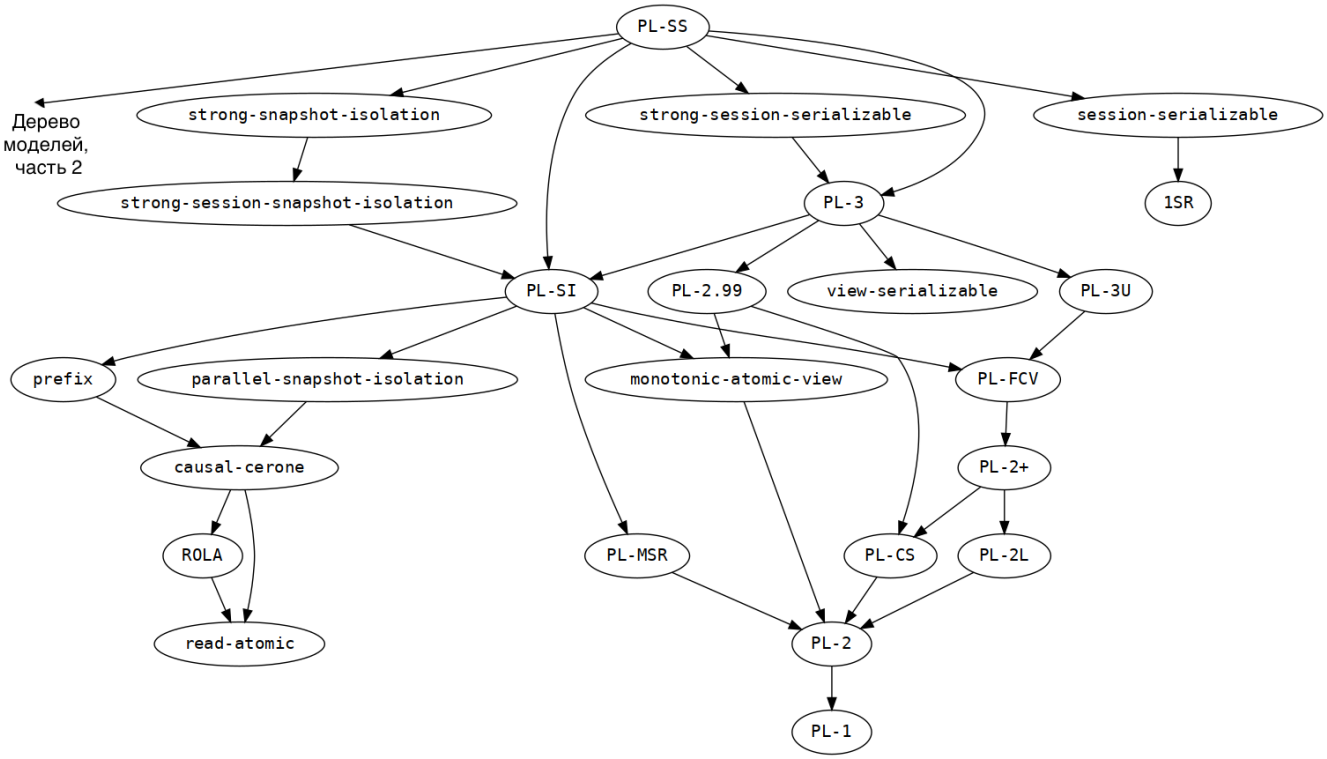

Рис. 6: Дерево моделей в Elle, часть 1

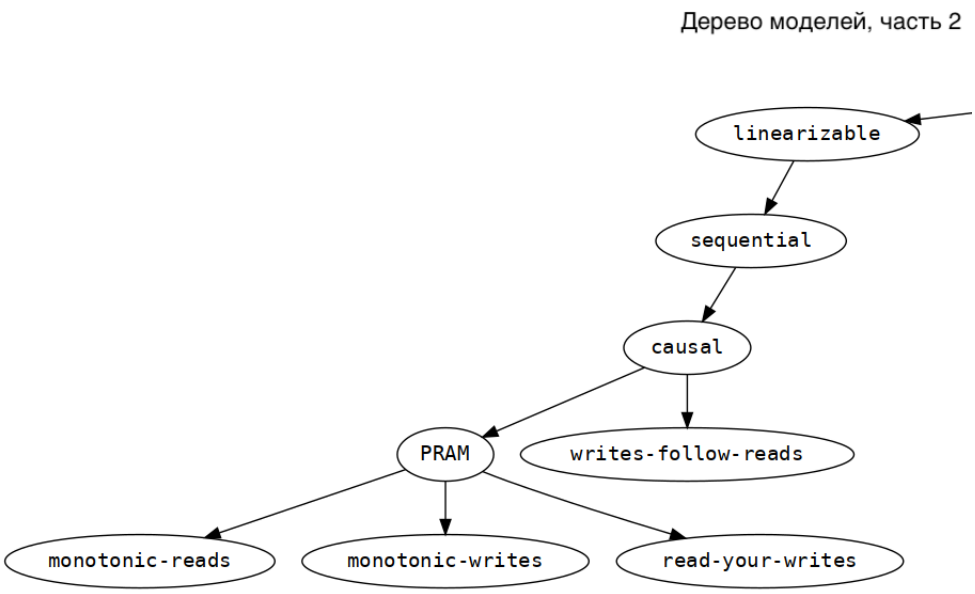

Рис. 7: Дерево моделей в Elle, часть 2

6 Результаты тестирования - MySQL server

При тестировании одноузловой MySQL БД установка и конфигурация производилась вручную перед запуском тестов и после их завершения. Версия тестируемой БД - ”8.0.29”. Использовался InnoDB engine. Была проведена проверка для всех 4 предоставляемых уровней изоляции. Во всех четырех случаях было использовано три таблицы и пять процессов, применяющих операции.

6.1 READ UNCOMMITTED

6.1.1 Elle checker - READ UNCOMMITTED

Запуск не выявил аномалий, нарушающих уровень изоляции **READ UNCOMMITTED**. Это означает, что в истории применения операций

не было найдено экземпляров **G0: write cycle (dirty write)**.

6.1.2 Elle checker - READ COMMITTED

Попробуем выяснить предоставляет ли БД более мощные гарантии на данном уровне. Запустим тестирование с моделью **READ COMMITTED**. Ожидаемо тест обнаружил аномалии:

1. **Incompatible order** (ведёт к G1a);
2. **G1b: Intermediate Reads (dirty reads)**;
3. **G1c: Circular Information Flow (dirty reads)**.

Для **G1c** инструмент Elle сгенерировал визуализацию:

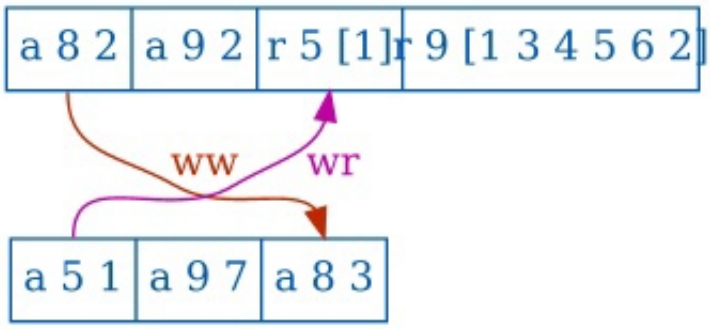

Рис. 8: G1c, запуск RU - RC

Первая транзакция производит добавление второго элемента в список №8, затем в рамках этой же транзакции производится чтение из спика №5 - список состоит из первого элемента. Вторая транзакция добавляет в список №5 первый элемент, который был считан во второй транзакции. Отсюда можно сделать вывод, что первая транзакция шла после второй. Однако позже во второй транзакции в список №8 добавляется третий элемент, который не был найден вначале первой транзакции.

6.2 READ COMMITTED

6.2.1 Elle checker - READ COMMITTED

Запуск не выявил аномалий, нарушающих уровень изоляции **READ COMMITTED**. Это означает, что в истории применения операций не было найдено экземпляров

1. **G1a: Aborted Reads (dirty reads, cascaded aborts)**;
2. **G1b: Intermediate Reads (dirty reads)**;
3. **G1c: Circular Information Flow (dirty reads)**.

6.2.2 Elle checker - MONOTONIC ATOMIC VIEW

Выясним предоставляет ли БД более мощные гарантии на данном уровне. Запустим тестирование с моделью **MONOTONIC ATOMIC VIEW**.

При использовании указанной модели не были обнаружены нарушающие её аномалии. В частности, не было обнаружено экземпляров **OTV: Observed Transaction Vanishes**.

6.2.3 Elle checker - REPEATABLE READ

Попробуем ещё раз повысить уровень изоляции - теперь используем **REPEATABLE READ** модель. Данный запуск выявил **G2-item: Item Anti-dependency Cycles (write skew on disjoint read)**.
Для **G2-item** инструмент Elle сгенерировал визуализацию:

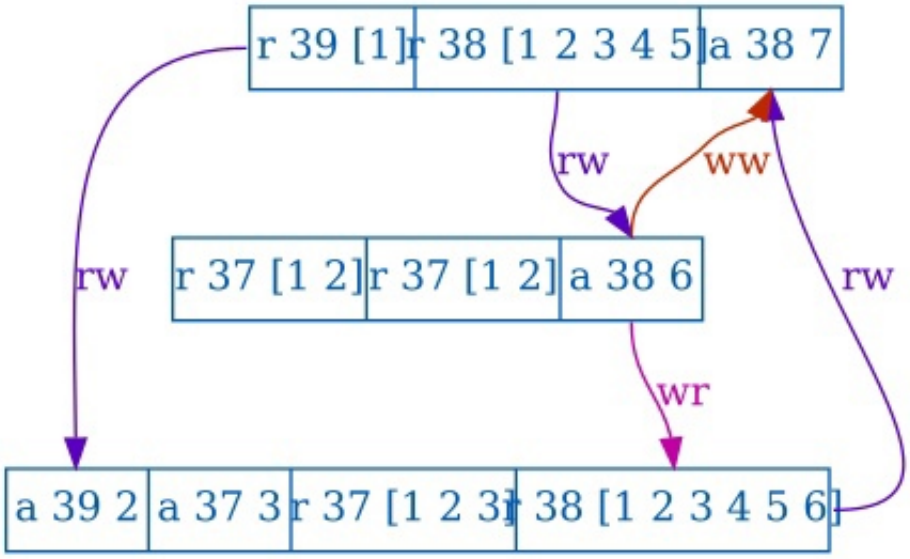

Рис. 9: G2-item, запуск RC - RR

Обозначим транзакции на рисунке как T1, T2, T3, где T1 - транзакция записанная сверху, T2 - посередине, T3 - снизу.

1. T3 последней операцией считывает список №38 и наблюдает в нем шесть элементов (числа в списки кладутся по возрастанию), таким образом T3 наблюдает результат последней операции T2, ведь именно T2 добавила в список №38 шестой элемент. Получили последовательность: T3 после T2.

2. Также T3 считывая список №38 не видит в нём седьмого элемента, который добавляется в последней операции T1. Отсюда получаем последовательность: T1 после T3.

3. При этом T1 второй операцией считывает список №38 и видит в нём пять элементов, значит нужно предположить, что T1 идёт перед T2. Ведь T2 последней операцией увеличивает количество элементов в списке №38 до шести. Получили: T2 после T1.

В итоге имеем цикл: T1 после T3, T3 после T2, T2 после T1.

6.3 REPEATABLE READ

6.3.1 Elle checker - REPEATABLE READ

Запуск транзакций в режиме **REPEATABLE READ** с такой же проверяемой моделью в Elle. Тестирование выявило аномалии: так же как и в предыдущем тесте обнаружены экземпляры
G2-item: Item Anti-dependency Cycles (write skew on disjoint read).

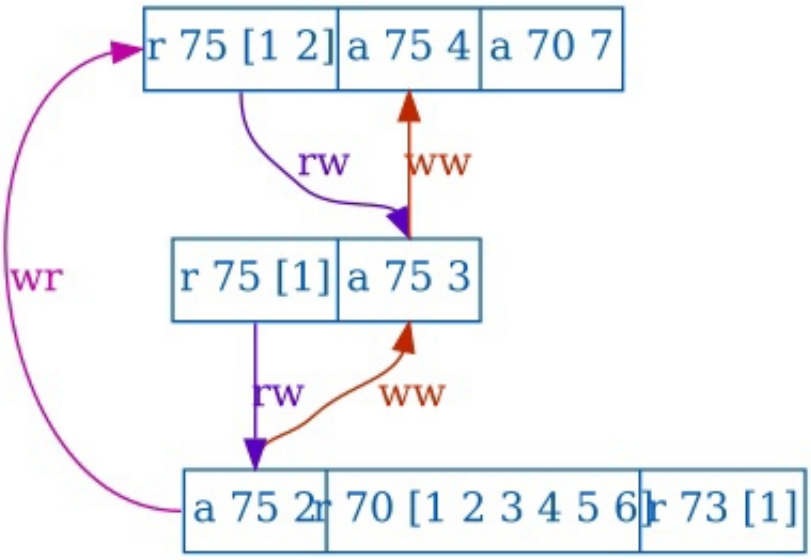

Рис. 10: G2-item, запуск RR - RR

Таким образом уровень изоляции **REPEATABLE READ** в MySQL server не соответствует определению приведённому в статье **”Generalized Isolation Level Definitions”** авторов Атула Адьи, Барбары Лисков, Патрика О’Нил [3]. Выясним какому уровню в уточнённой иерархии уровней изоляции соответствует MySQL server **REPEATABLE READ**.

6.3.2 Elle checker - MONOTONIC ATOMIC VIEW

Также, как и в случае проверки **READ COMMITTED** против **MONOTONIC ATOMIC VIEW**, не было обнаружено экземпляров **OTV**.

Следовательно уровень **READ COMMITTED** соответствует уровню **MONOTONIC ATOMIC VIEW** по терминологии [2].

6.3.3 Elle checker - SERIALIZABLE

В целях иллюстрации аномалий, запрещённых на уровне **SERIALIZABLE**, запустим тест REPEATABLE READ с проверкой на SERIALIZABLE.

Ожидаемо обнаруживаются экземпляры аномалий:

1. **G2-item**;
2. **G-single**: Single Anti-dependency Cycles (read skew);
3. **G-nonadjacent** - ведёт к **G2**.

Аномалия **G2-item** должна предотвращаться начиная с уровня **REPEATABLE READ**, но как было показано ранее уровень **REPEATABLE READ** в MySQL server соотетствует уровню **MONOTONIC ATOMIC VIEW** по [2].

Рассмотрим подробнее найденные экземпляры **G-single**, **G-nonadjacent** с точки зрения иерархии аномалий (Рис. 5). Если уровень изоляции не предотвращает **G-single**, то он также не защищает от **G-nonadjacent**. Наличие **G-nonadjacent**, в свою очередь, означает наличие **G2**.

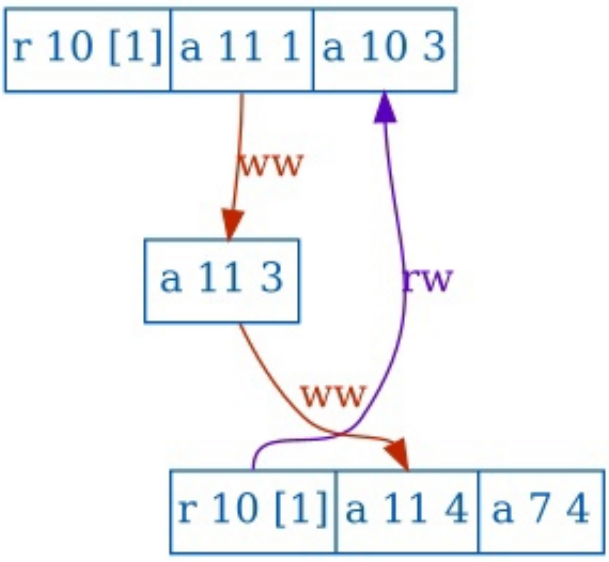

Рис. 11: G-single, запуск RR - S

G-single: Обозначим транзакции на рисунке как T1, T2, T3, где T1 - транзакция записанная сверху, T2 - посередине, T3 - снизу.

1. T3 первой операцией считывает список №10 и видит в нём только единицу. При этом T1 последней операцией добавляет 3 в список №10. Отсюда выводим, что T1 идёт после T3.

2. T1 второй операцией добавляет в список №11 первый элемент. T2 состоит только из одной операции - добавление в список №11 третего элемента. Отсюда выводим, что T2 идёт после T1.

3. T3 своей второй операцией производит добавление четвёртого элемента в список №11. Это означает, что T3 идёт после T2, которая добавляет в список №11 третий элемент.

В итоге имеем цикл: T2 после T1, T1 после T3, T3 после T2.

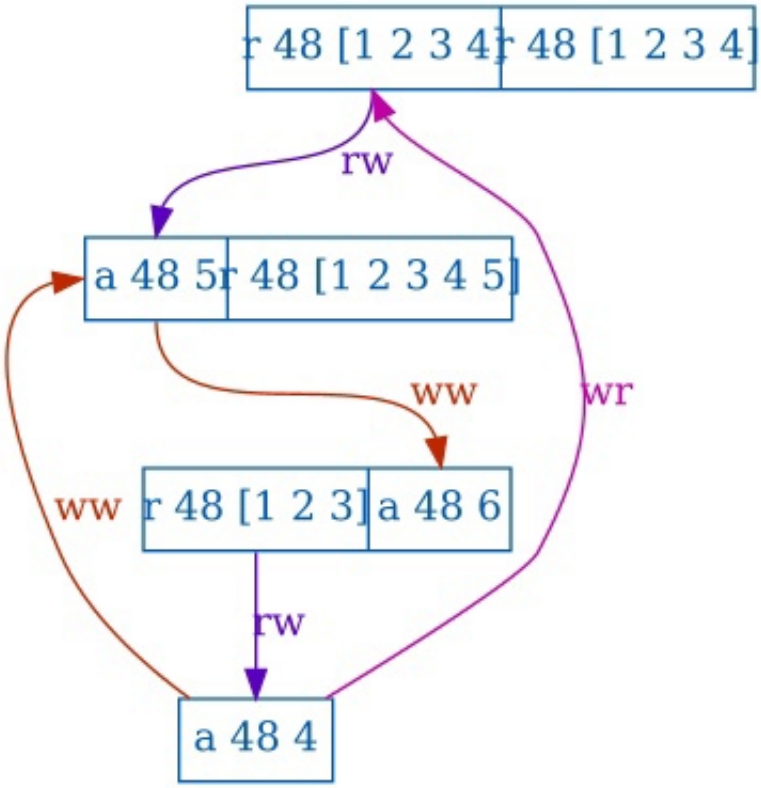

Рис. 12: G-nonadjacent, запуск RR - S

G-nonadjacent: Обозначим транзакции на рисунке как T1, T2, T3, T4 где T1 - транзакция записанная сверху, T2 - посередине сверху, T3 - посередине снизу, T4 - снизу.

1. T2 записывает пятый элемент в список №48. T3 записывает шестой элемент в список №48. Делаем вывод, что T3 идёт после T2.

2. T3 первой операцией считывает список №48 и наблюдает в нём три элемента. T4 состоит из записи четвёртого элемента в этот список. Заключаем, что T4 идёт после T3.
3. T1 в свою очередь видит запись четвёртого элемента в список №48. Тогда T1 идёт после T4.
4. T1 считывая список №48 не видит в нём пятый элемент, который записывается T2. Выходит, T2 идёт после T1.
В итоге имеем цикл: T2 после T1, T1 после T4, T4 после T3, T3 после T2.

6.4 SERIALIZABLE

6.4.1 Elle checker - SERIALIZABLE

Запуск не выявил аномалий нарушающих уровень **SERIALIZABLE**. В частности, это означает что не были обнаружены экземпляры

1. **G-single: Single Anti-dependency Cycles (read skew)**;
2. **G-nonadjacent**;
3. **G2: Anti-dependency Cycles**.

7 Результаты тестирования - MySQL cluster

7.1 READ COMMITTED

При тестировании MySQL cluster установка и конфигурация производилась автоматизированно средствами Jepsen. Для создания окружения использовался инструмент Docker, кластер был запущен на трёх контейнерах. Четвёртый контейнер использовался как узел для запуска тестирования. MySQL cluster использует NDBCLUSTER engine, который позволяет использовать только уровень изоляции **READ COMMITTED**. Для хранения списков Было использовано три таблицы. Применяли операции пять процессов.

7.1.1 Elle checker - READ COMMITTED

Также как и в случае с одноузловой БД запуск не выявил аномалий, нарушающих уровень изоляции **READ COMMITTED**. Это означает, что в истории применения операций не было найдено экземпляров

1. **G1a: Aborted Reads (dirty reads, cascaded aborts)**;
2. **G1b: Intermediate Reads (dirty reads)**;
3. **G1c: Circular Information Flow (dirty reads)**.

7.1.2 Elle checker - REPEATABLE READ

Так как MySQL cluster предоставляет только один возможный уровень изоляции, то очевидно предположение, что MySQL cluster не выполняет гарантии модели **REPEATABLE READ**. В противном случае, логично предположить, что именно этот уровень был бы указан, как единственно возможный. Действительно, тестирование выявило аномалии: обнаружены экземпляры

G2-item: Item Anti-dependency Cycles (write skew on disjoint read).

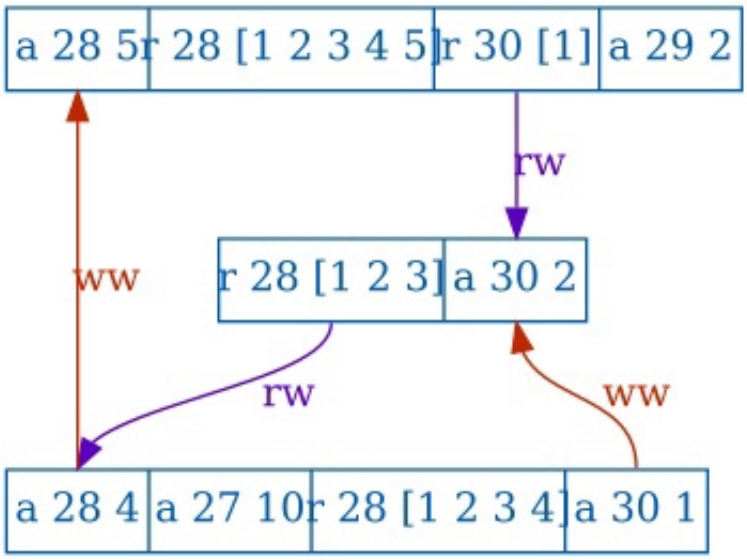

Рис. 13: G2-item, запуск cluster - RC - RR

Наблюдаем **G2** цикл с двумя anit-dependency рёбрами.

Список литературы

1. A Critique of ANSI SQL Isolation Levels / H. Berenson [и др.] // Technical Report MSR-TR-95-51. — 1995. — URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-95-51.pdf.
2. *Adya A.* Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions. — 1999. — URL: https://pmg.csail.mit.edu/papers/adya-phd.pdf.
3. *Adya A.*, *Liskov B.*, *O'Neil P.* Generalized Isolation Level Definitions // Proceedings of the IEEE International Conference on Data Engineering. — 2000. — URL: https://pmg.csail.mit.edu/papers/icde00.pdf.
4. *Kleppmann M.* Designing Data-Intensive Applications. — 2017. — URL: https://dataintensive.net/.
5. MySQL Official. — https://www.mysql.com/.
6. Jepsen - Distributed Systems Safety Research. — https://jepsen.io/.
7. Test repository. — https://github.com/TheNeonLightning/mysql-cluster-jepsen-test.