rus.bce_events_flow

Alexandr Kirilov (https://github.com/alexandrkirilov)

Blockchain пример для Event Flow.

Пример реализации blockchain решения для хранения событий (events) основан на статье "Event Driven" и понимание того, что это только пример архитектурного решения, и конечная реализация зависит от многих факторов:

- Какой тип СУБД вы используете SQL или noSQL, или еще что-то еще малоизвестное?
- Какую СУБД вы используете (MySQl, Mnesia, Postgre, CouchDB, Oracle и т.д) и на какой платформе (Windows, Linux-like или Freebsd-like и т.д)?
- Используются ли данные хранящиеся таким способом в проектах High-Load или Big-data? Если да то как?
- И т.д.

И основной вопрос - что и от кого или чего нужно защитить путем использования blockchain? Нужно сформировать основную идею (idea) той цепи (chain) которая будет заблокирована (block).

Дано:

Есть поток событий (event), который содержит в себе разнородные данные и которые строго привязаны ко времени. Примером такого потока может быть журнал событий в системе (log-book), личная карточка студента в университете (где сдача экзамена и получение оценки - event, получение диплома - event и т.д), личная карточка пациента (где поток событий привязан к имени пациента и постановка диагноза - event, выписывание рецепта - event, выздоровление пациента - event и т.д). Примером может служить любая event-driven система.

Задача:

Избежать возможности манипуляцией событиями относительно времени (избежать возможности изменения зафиксированного времени события во время хранения данных о событии) и содержания события (обеспечить фиксацию данных содержащихся в записи о событии) настолько, насколько это максимально возможно.

Решение:

Для формирования решения нужно определить ключевые элементы на основании которых будет выстроена схема blockchain (исходя из статьи о 3W упомянутой ранее - "Event Driven"):

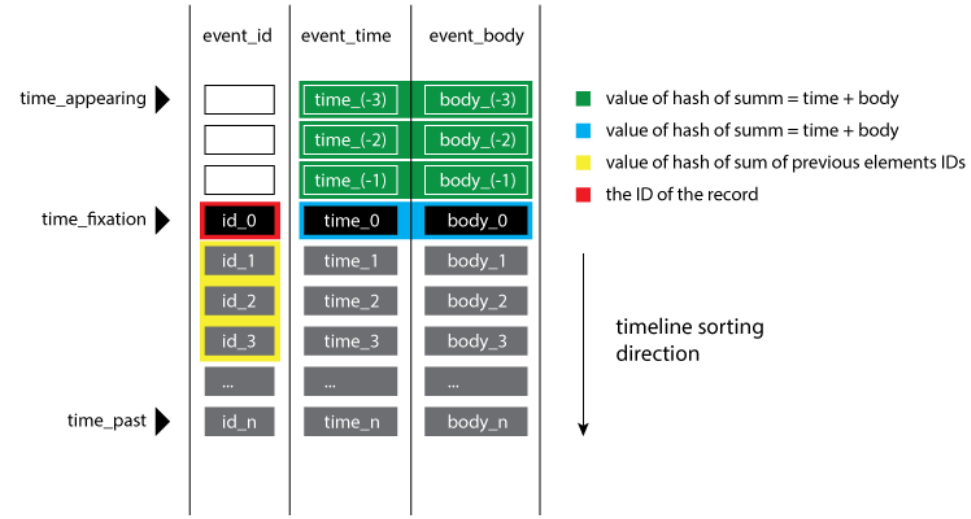
- When? (время): это следует из описания структуры данных, event flow где расположение событий на timeline очень важно. Нельзя допустить чтобы через год выяснилось что событие произошло на 15 минут позже или раньше чем это было на самом деле. Это и есть цепь (chain) которую мы должны блокировать (block).
- What? и Where? (тело события): это непосредственно данные о событии которое произошло, конкретные детали нас не интересуют на этом уровне по отдельности. Мы должны зафиксировать целиком What? и Where?

Следующий этап - выяснение того, что будет являться уникальным для каждого выделенного ключевого элемента

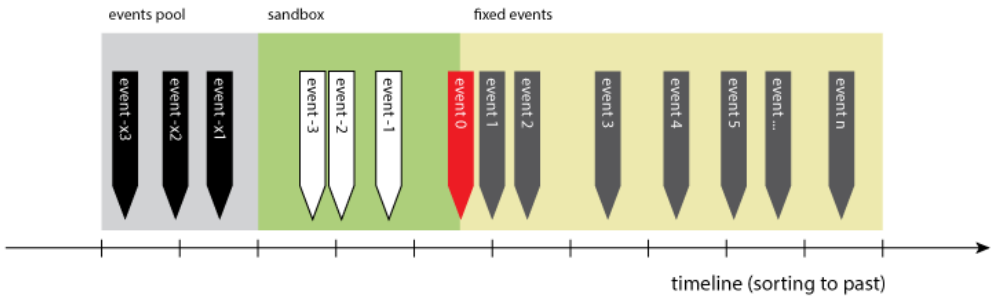
- для фиксации события на timeline - время, скорей всего это будет UNIX-timestamp
- для фиксации тела события будет md5 или md4 от тела события
- уникальной для 3W схемы будет md5 или md4 от суммы первых двух уникальных значений (помечено голубым и зеленым цветом на рисунке)

Нужно эти уникальные элементы связать между собой для того чтобы изменение одной записи, повлекло изменение остальных записей. Невозможность изменить только одну запись является механизмом блокировки всей таблицы от изменения.

Основным способом блокировке является способ генерации unique ID для события и метод его записи в базу данных.



Как изображено на рисунке unique ID для event является hash от суммы ID 3-х предидущих событий и суммы hash-ей от event_body и event_time от трех впереди стоящих событий (ID не используются на этом этапе т.к. они еще не зафиксированы). Данный способ фиксирует связь между элементами и делает практически не возможным незаметное изменений данных. При использовании данного способа нужно будет пересчитать все ID всех записей что вызовет огромный скачек нагрузки на сервера хранящие данные и простейшая система мониторинга это определит практически мгновенно.



Метод записи данных в базу состоит из трех этапов (основной принцип этого метода можно назвать - "застывающий бетон" или "застывающий клей"):

- event pool: первичное место появления событий в системе, здесь не происходит никакой фиксации, рекомендуется использовать "на всякий случай" давая автору

события изменить его данные в течении короткого времени, например в случае опечатки дается 5 минут на все изменения

- sand box: условная часть метода, на этом этапе событие уже записывается в основную базу но не фиксируется, или фиксируется каким-то специальным ID связанным с позицией в sand-box и временем события
- fixed events: основная часть базы данных в которых значение ID уже зафиксировано

Если вы используете BSD License based СУБД это позволит достаточно легко (при наличии определенных навыков) встроить триггер на время исполнения запроса к базе (пример: если время выше, чем допустимое значение для единичного запроса то система заблокирует транзакцию), что в свою очередь не оставит шанса изменить данные незаметно.