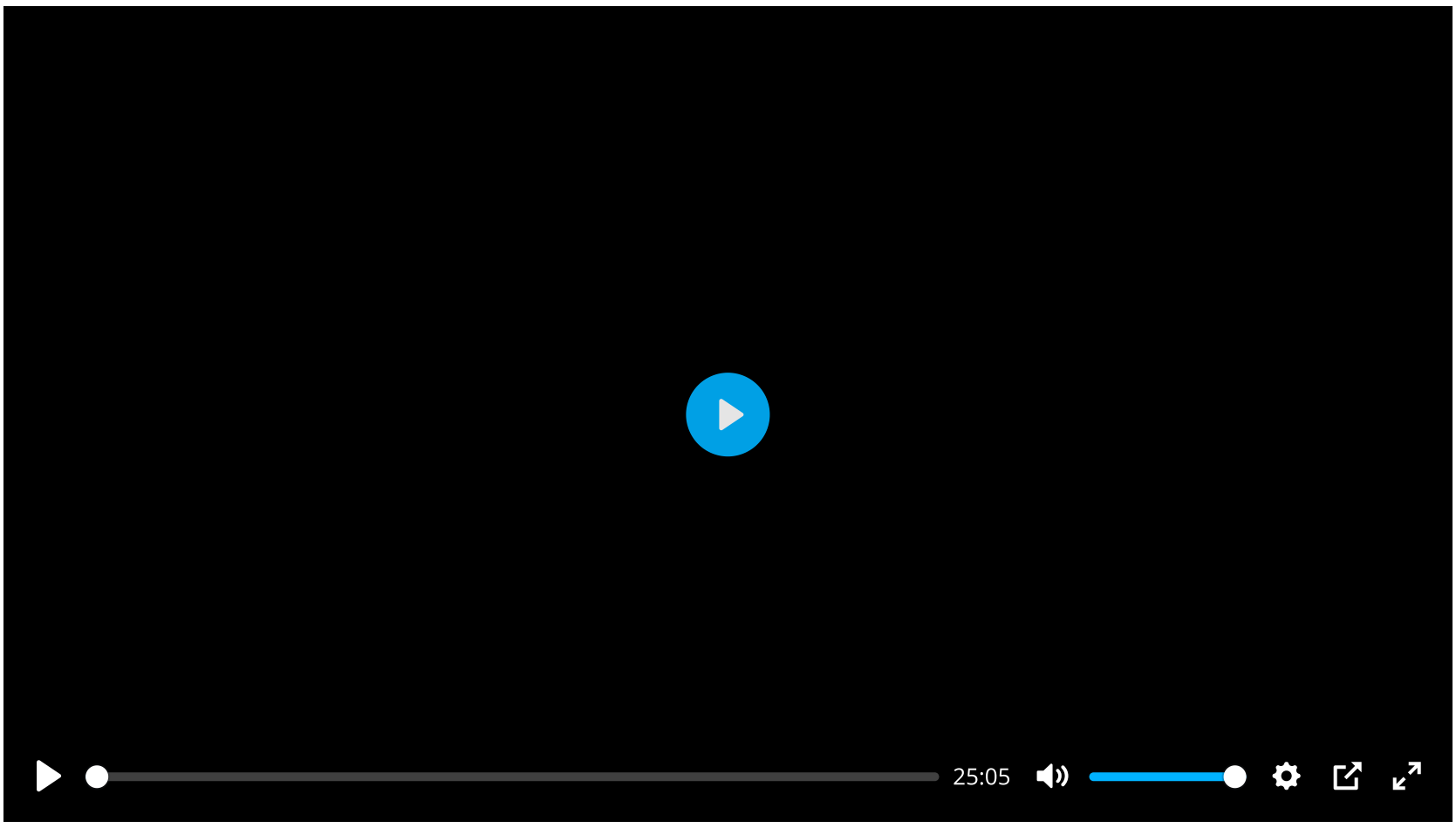
5.3 SQL. Принципы ACID.



Текстовая расшифровка видео:

ACID и транзакции

План:

- Разница между базами;
- Кислотная аббревиатура;
- Как Евлампий лошадь продавал;
- Durability;
- Consistency;
- Atomicity;
- Isolation;
- Транзакции в аналитическом хранилище;
- Транзакции в SQL.

Разница между базами

Основная разница между БД:

- В количестве писателей и читателей;
- В профиле нагрузки;
- В гарантиях, которые они предоставляют.

Кислотная аббревиатура

ACID – это аббревиатура, состоящая из свойств, которыми должна обладать БД для определенной сертификации:

A – Atomicity (атомарность);

C – Consistency (консистентность);

I – Isolation (изоляция);

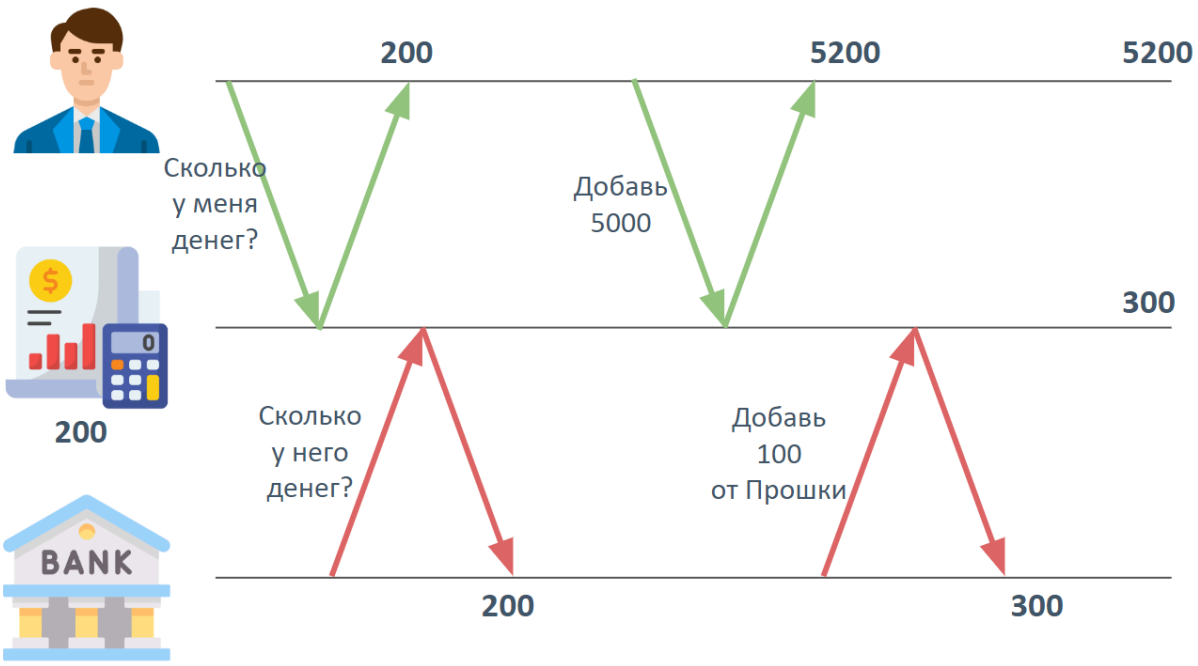
D – Durability (надежность).

В разных контекстах данные слова предстают под разными понятиями, из-за этого возникает путаница.

Транзакция – операция или группа операций, которая представляется в виде **логически независимого** действия поверх данных, которое либо **выполняется целиком**, либо **не выполняется вообще**.

Как Евлампий лошадь продавал

Рассмотрим на примере:



Евлампий пошел продавать лошадь на рынок и успешно сделал это. Лошадь была продана за электронную валюту. В это время товарищ Евлампия – Прошка решил вернуть долг в размере 100 рублей.

Мы видим, что одновременно запустились две операции:

- у Евлампия пришли деньги за продажу лошади;
- пришли деньги от Прошки.

Каждая из этих операций состоит из двух частей:

- проверить общую сумму денег;
- добавить или убрать соответствующую сумму к ним, записать результат на счет.

Если эти операции выполняются неатомарно и их нельзя откатить одновременно, то может возникнуть следующая ситуация:

Когда Евлампий продал лошадь, первой операцией была проверка количества денег на счету (условно было 200 рублей). В это же время пришел транш от Прошки, запрос о количестве денег на счету. Когда прошел транш за лошадь в размере 5000 рублей, было взято число, которое было у Евлампия на счету (200), к нему прибавились 5000, и вся сумма в итоге записалась на счет.

Одновременно с этим со стороны Прошки произошла та же самая проверка:

- проверка количества денег на счету Евлампия;
- добавление суммы от Прошки в размере 100 рублей;
- запись результирующей суммы на счету.

Получилось так, что Евлампий продал лошадь, но у него на счету осталось 300 рублей, потому что возникло состояние, из-за которого модификация произошла неатомарно: сумма, пришедшая от Прошки, перезаписала ту, что пришла от продажи лошади.

Чтобы понять, чем могут помочь транзакции, поочередно разберем каждую букву в аббревиатуре «ACID».

Durability

Durability объяснятся как подтвержденная физическая запись данных.

Durability может быть не 100% (она может выражаться в процентах). Так, например, Amazon Cloud представляет durability 12/9. Это означает, что с такой вероятностью при записи данных в Amazon S3 мы получим их обратно при попытке прочитать.

Consistency

Данное слово встречается в CAP-теореме. Значение данного слова зависит от контекста.

Часто под «Consistency» в программировании, в разговорах о БД подразумеваются данные, которые не ломаются. **В случае ACID контекст другой.**

Любая операция по работе с базой, которая представляет собой единую транзакцию, должна обязательно переводить базу из одного стабильного состояния в другое.

Atomicity

Именно с этим словом возникает путаница.

Atomicity никак не связана с конкурентностью/параллельностью! **Она описывает обработку ошибок.** Все, что написано в транзакциях при работе с базой либо проходит целиком и приводит из одного стабильного состояния в другое стабильное состояние, либо, если что-то в середине пошло не так, все операции откатываются, а базы откатываются к начальному состоянию.

Варианты того, что может произойти:

- Падение/отключение питания;
- Нарушение constraint’ов;
- Глюки сети;
- Дедлоки.

Isolation

Одна из основных идей – **serializability**, то есть отсутствие разницы между **одновременным** выполнением транзакций и **последовательным**. Однако это слишком медленно, поэтому были придуманы уровни изоляции транзакции (самые известные):

- Read Uncommitted;
- Read Committed ← по умолчанию в большинстве баз;
- Repeatable Read/Snapshot Isolation (MVCC);
- Serializable.

Ссылка на почитать

Транзакции в аналитическом хранилище

Вопрос: нужны ли транзакции в аналитическом хранилище?

Ответ: на наш взгляд, не нужны. Приведем в пример стандартизованный формат, с которым работают все Data Lake, – Parquet. Parquet не поддерживает на уровне формата ACID. Parquet’ы записываются в хранилище в режиме read only.

Важный нюанс касается порядка ставки. Поскольку одной из основных проблем в распределенных системах является доставка сообщений, то возникает вопрос: «насколько важен порядок?». Когда мы вставляем данные в базу, может произойти ситуация, когда порядок вставки внезапно окажется нехронологическим. Теоретически транзакции могли бы это подправить и зафорсить, но это не всегда актуально.

Parquet не поддерживает транзакции. Современные форматы, например, ORC/Iceberg/YTSaurus умеют с ними работать.

Ссылка на почитать

Транзакции в SQL

В SQL с транзакциями работа предельно простая.

Основные шаги:

- Пишем «**Begin**»;
- Пишем любые запросы, которые делают вставку, чтение и т.д.;
- Пишем «**Commit**» в конце;
- В середине можем использовать «**Rollback**», если внезапно в процессе выполнения транзакции выясним, что она не актуальна и по какой-то причине не может быть завершена. Rollback откатывает базу к стандартному состоянию на момент начала.

Ссылка на почитать

ИТОГИ

О чем поговорили:

- Что такое SQL и как начать с ним работать;
- Об основах работы с PostgreSQL. Что почитать для углубленного изучения SQL;
- ACID и транзакции. Гарантии и подводные камни;
- Нужны ли транзакции в аналитическом хранилище.

Как вам урок? 😠 😐 🔥

Изучил, далее ›