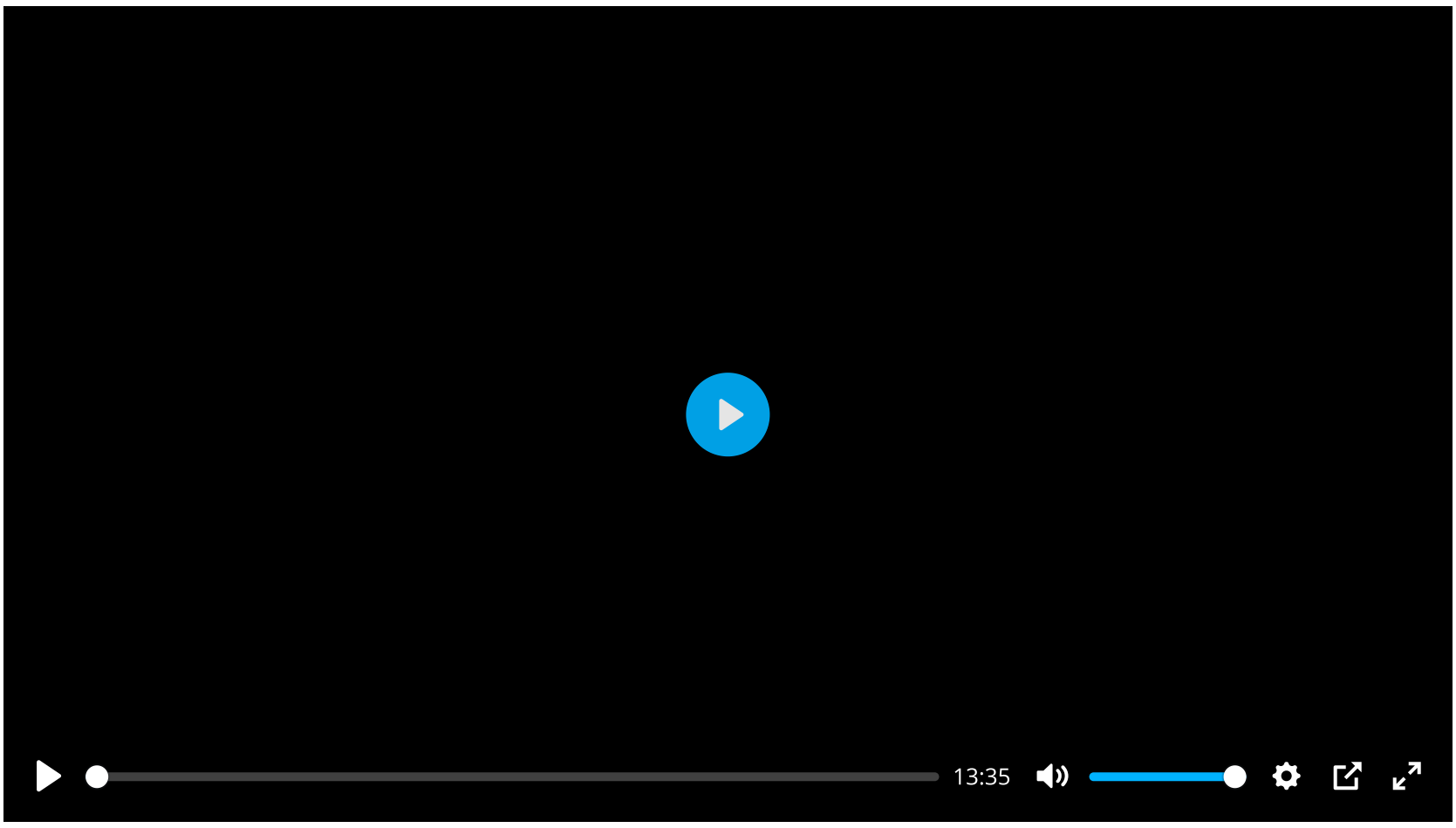
5.2 Реляционные базы данных. Введение



Текстовая расшифровка видео:

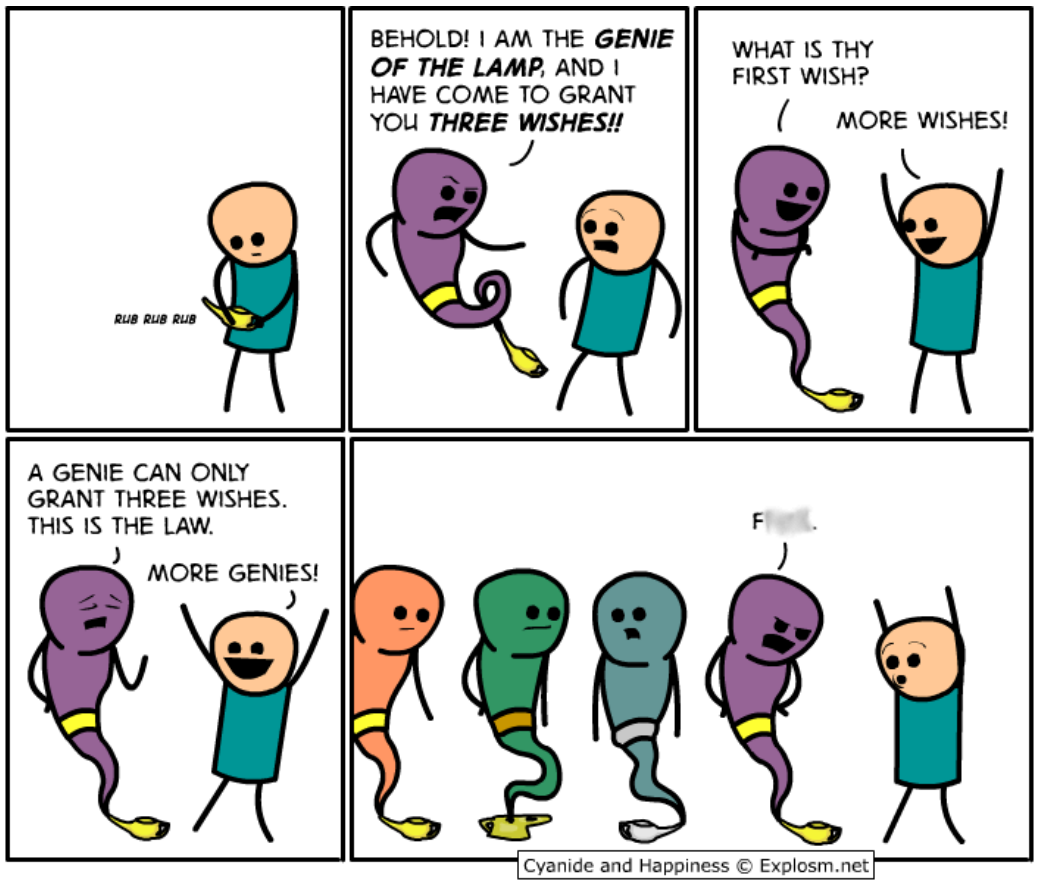
КАК БАЗЫ МАСШТАБИРУЮТСЯ

- Вертикальное и горизонтальное масштабирование;
- Репликация;
- Типы репликации;
- Партиционирование и шардинг;
- О решении проблем при падении;
- Траблшутинг.

Вертикальное и горизонтальное масштабирование

Одно из базовых делений – это вертикальное и горизонтальное масштабирование.

Данный комикс отлично иллюстрирует идею:



Вертикальное масштабирование – это подход, при котором мы можем в одну и ту же машину/ физический сервер дописать больше оперативки, поставить более мощный процессор, поставить больше жестких дисков, переписать какой-либо алгоритм, чтобы он работал быстрее, эффективнее и т.д.

Такой подход достаточно сложен и имеет пределы.

Горизонтальное масштабирование – это подход, при котором мы начинаем раскидывать запросы больше, чем на одну машину в кластере.

Репликация

Основная идея: разбиваем данные на куски, делаем N копий (фактор репликации) каждого куска. Каждую копию записываем на отдельную машину.

Основные положения:

- Данные географически ближе к пользователям;
- Выше Availability (доступность). Если одна из машин упадет, то копии будут лежать на других машинах. Так клиентский запрос можно будет перенаправить на другую машину.
- Выше Throughput (мы размазываем нагрузку на большее количество машин).

Фактор репликации – это количество кусков, на которое мы бьем данные.

То, куда раскидываются реплики, определяется политиками.

Типы репликаций

Различают **асинхронную репликацию и синхронную репликацию.**

Суть синхронной репликации:

Мы записываем данные в машину. Пока данные физически не распределятся по какому-то количеству машин в кластере, нам не возвращается управление.

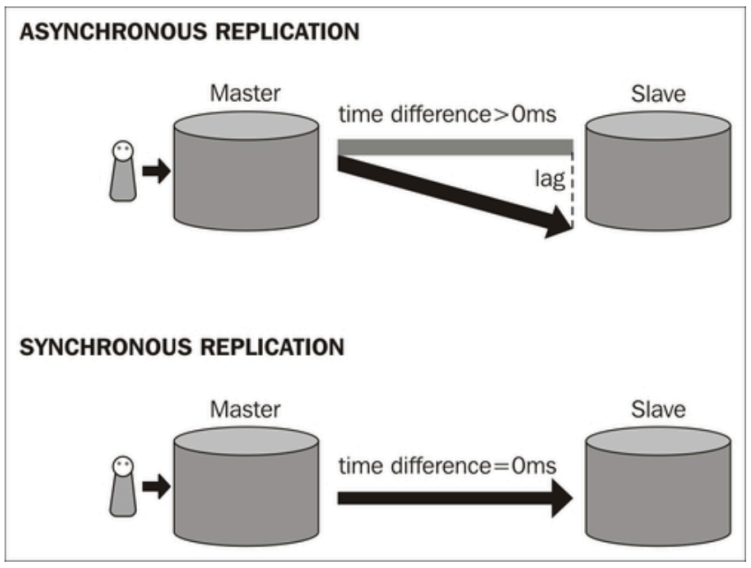
Суть асинхронной репликации:

Результат возвращается быстрее. Мы записали данные, получили управление обратно; в будущем настанет ситуация, когда данные перекинутся на все дочерние узлы.

Часто есть разделение:

Leader -> Follower

Leader -> Leader



Есть Master Slave (он же Leader Follower), когда для записи используется конкретная нода master'а. Она реплицируется на slave'ы, где мы, например, можем только читать (а писать только в master). Часто там используется именно асинхронная репликация. Также может быть Master Master-репликация или Leader Leader-репликация (это более сложный вариант, где есть несколько головных машин). Между мастерами часто используются либо полностью синхронная репликация, либо хитрые разновидности асинхронной, так как привести все к консистентному состоянию довольно сложно.

Партиционирование и шардинг

Довольно часто возникает путаница между терминами «Репликация», «Партиционирование» и «Шардинг».

Партиционирование и Шардинг во многих источниках используются как синонимы (однако некоторые так не считают). Суть данных терминов на наш взгляд одинакова:

- у нас есть кластер и табличка;
- табличку бьем на куски;
- куски этой таблички раскидываем по кластеру по определенным правилам.

Чтобы это работало должен быть **ключ шардирования**, то есть какой-то признак, по которому мы будем разбивать данные на куски, чтобы дальше их раскладывать.

Выбор ключа в разных базах происходит по-разному:

- используют естественный ключ или порядковый индекс;
- используют хэширование.

Помним: если при партиционировании с использованием хэша колонка была выбрана неудачно, могут получиться сильно смещенные данные.

Предлагаем вам ознакомиться с методикой «Consistent Hashing» (эта информация – один из вариантов того, как организовать хэширование в распределенном кластере и решить проблему с добавлением и исключением нод в кластере).

О решении проблем при падении

Есть «демократический» принцип, который в Computer Science (в исследовании распределенных систем) называется «Leader relaction» (когда выключается машина, происходит голосование по определенным правилам, где ноды договариваются и выбирают нового лидера):

В мире БД существует более жесткая практика – Fencing. Fencing – это своего рода таймаут. Когда машина ушла в offline и какое-то время не получала важные апдейты от кластера, есть риск, что по возвращении в online, она окажется в неконсистентном виде. Возвращение к консистентному состоянию может быть весьма затратно, нежели мы физически отключим к ней сеть, переформатируем и подключим в кластер как новую машину.

Траблшутинг

Выводы, которые мы можем сделать:

- Мониторинг, а еще лучше observability – главный помощник в решении любых проблем с БД.
- Одна замедленная или неконсистентная нода может сломать весь кластер.
- Не факт, что вам потребуется настраивать все это руками, но умение разговаривать на одном языке с DBA может крайне пригодиться в «боевых условиях».

Как вам урок? 😠 😐 🔥

Изучил, далее ›