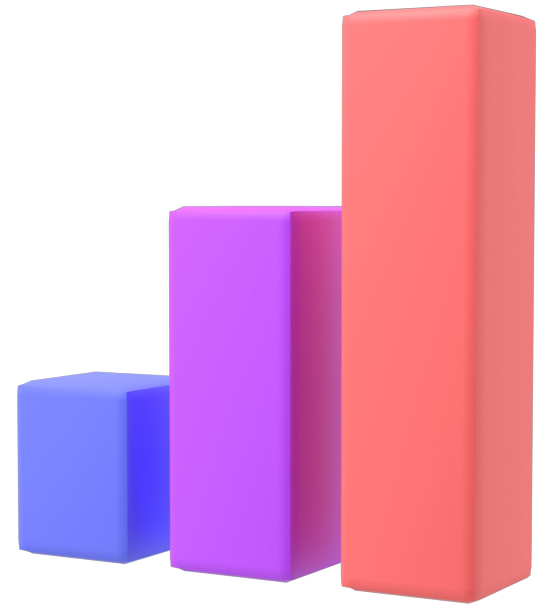


System Design

Хранение данных

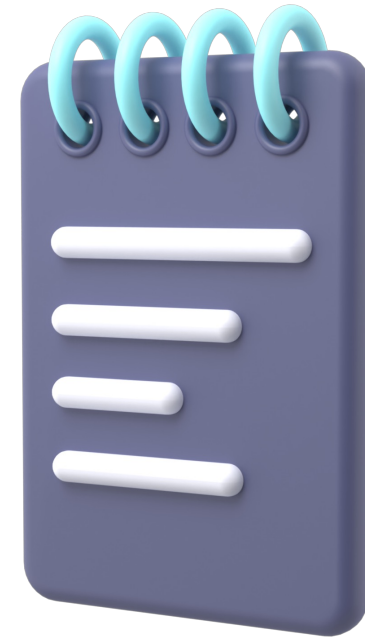


Проверь запись



Маршрут занятия

- Виды баз данных
- Классы баз данных
- Индексы
- Транзакции
- Объекты баз данных
- Брокеры сообщений
- Альтернативные способы хранения данных



Виды баз данных

Реляционные

UserID	Name	CityID
23	user_1	1
233	user_2	2
12	user_5	1

CityID	Name
1	Moscow
2	Berlin



Документоориентированные

```
{  
  "name": "Petr",  
  "surname": "Ivanov",  
  "city" : "Moscow",  
  "interests" : [  
    "football",  
    "hockey"  
  ],  
}
```



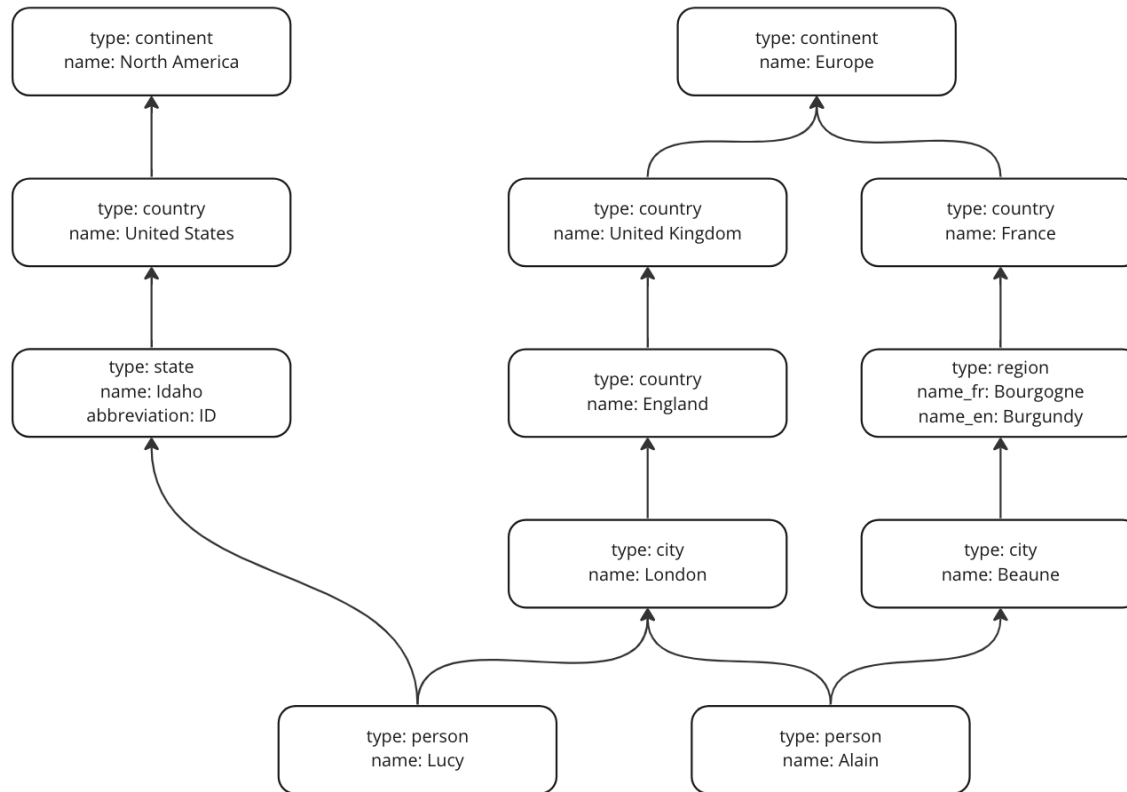
Поисковые движки



elasticsearch



Графовые



Key-value

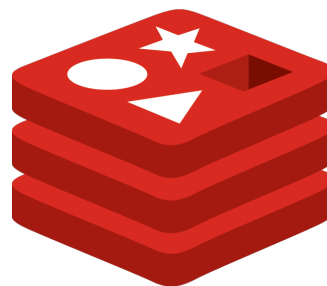
Key	Value
video_1	4232
video_2	23
video_3	2342344
video_4	2323
video_5	434



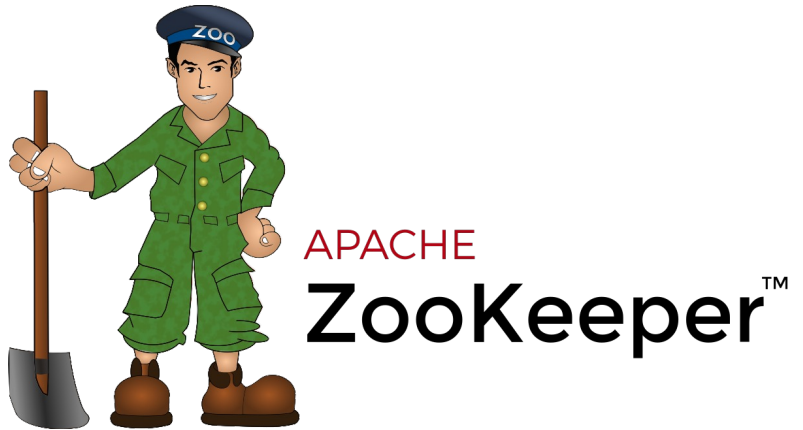
MEMCACHED



Tarantool



redis



Колоночные

UserID	Name	City
23	user_1	Moscow
233	user_2	Berlin
12	user_5	Moscow



ClickHouse



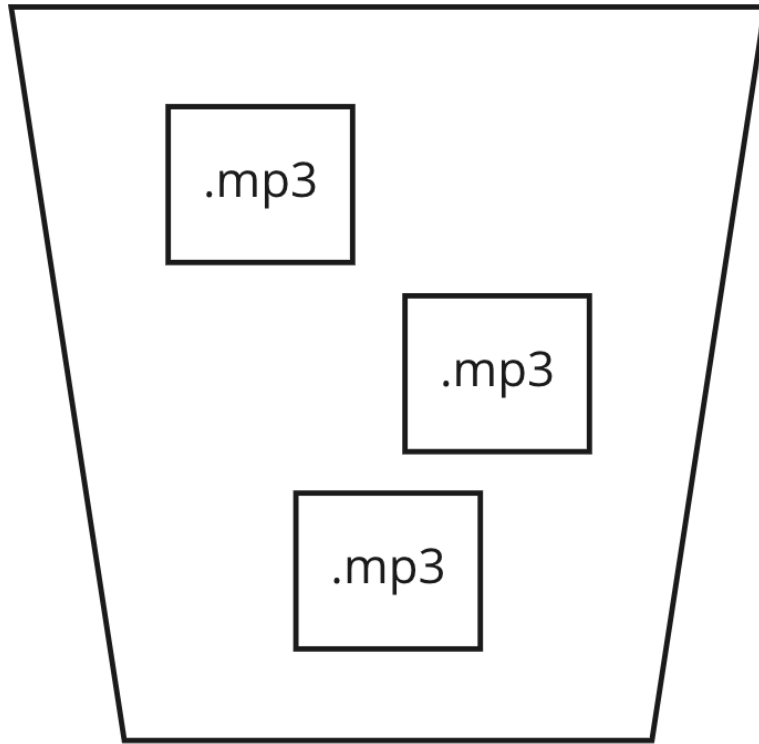
Cassandra

Time series

Timestamp	Temperature
10.01.2022 17:00	37.2
10.01.2022 17:01	37.1
10.01.2022 17:02	36.8



Blob store



Выбор базы данных

1. Транзакции
2. Формат данных
3. Навык работы с технологией
4. Характер обращений к данным
5. Сообщество и зрелось технологии
6. Частота изменяемости формата данных

Выбор базы данных

1. Денежные средств пользователя
2. Количество просмотров под видео
3. Анкеты пользователей
4. Отношения между пользователями
5. Исходный код программ
6. Показатели температурных датчиков

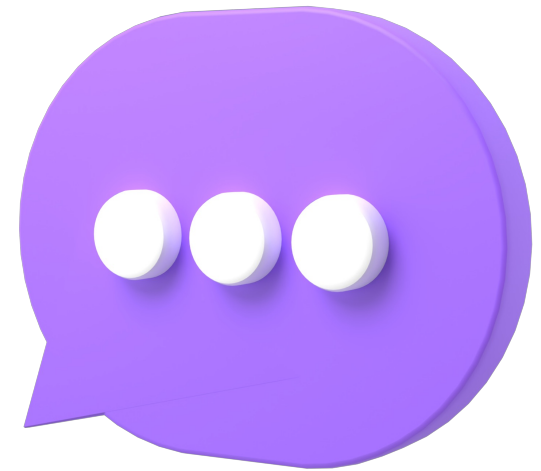
Выбор базы данных

Нужно сохранять и искать посты пользователей с использованием полнотекстового поиска, хранить аналитику и кэшировать популярные посты

FAQ:

Виды баз данных

- Реляционные
- Документориентированные
- Графовые
- Key-value
- Колоночные
- Time series
- Blob store



Классы баз данных

OLAP vs OLTP

Online Analytical Processing и Online Transaction Processing

HTAP

Hybrid transactional / analytical processing

Persistent базы данных

Базы данных, хранящие свои данные на дисках

In-memory базы данных

- Можно периодически записывать на диск копии состояния БД
- Можно записывать на диск журналы изменений (что-то вроде упомянутых выше WAL), но не сами данные
- Можно проводить репликацию состояния оперативной памяти на другие машины

Embedded database

An **embedded database** is a system, which is tightly integrated with an application software it is embedded in the application

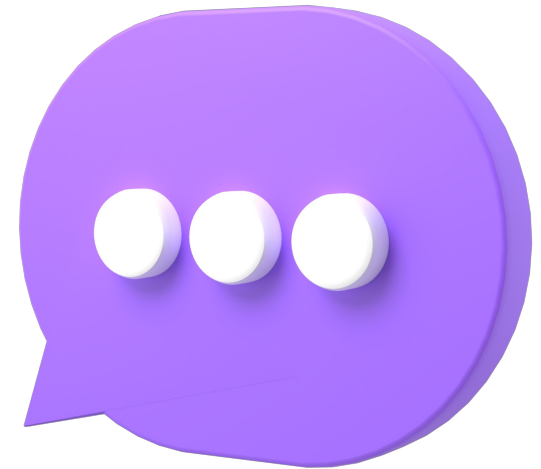
Single file database

That single file contains the database layout as well as the actual data held in all the different tables and indexes

FAQ:

Классы баз данных

- OLAP / OLTP
- In – memory
- Persistent
- Embedded
- Single file

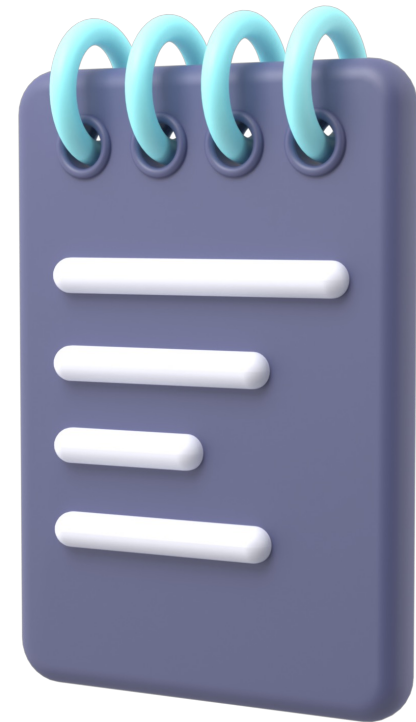


Индексы

Ускоряют чтение

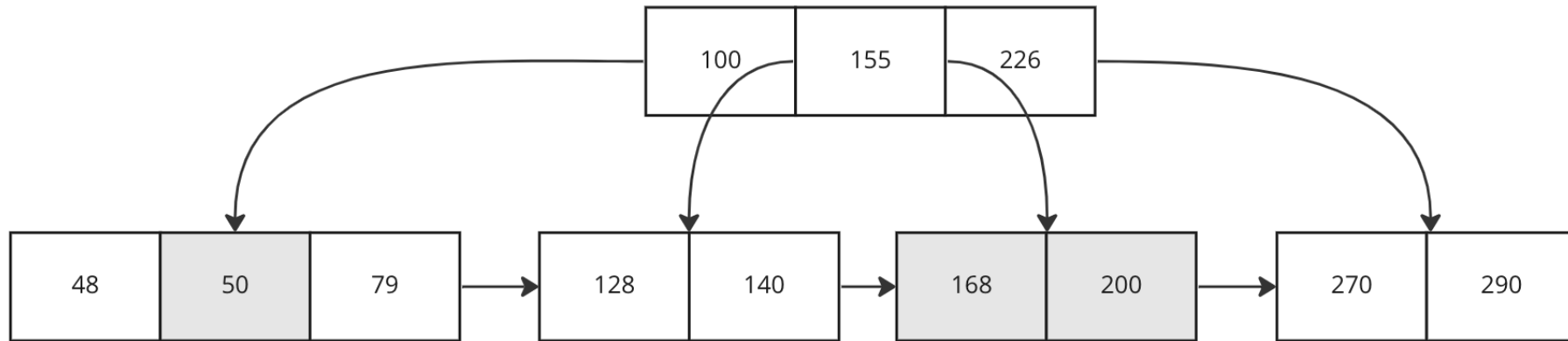
Замедляют запись

Используют дополнительную память

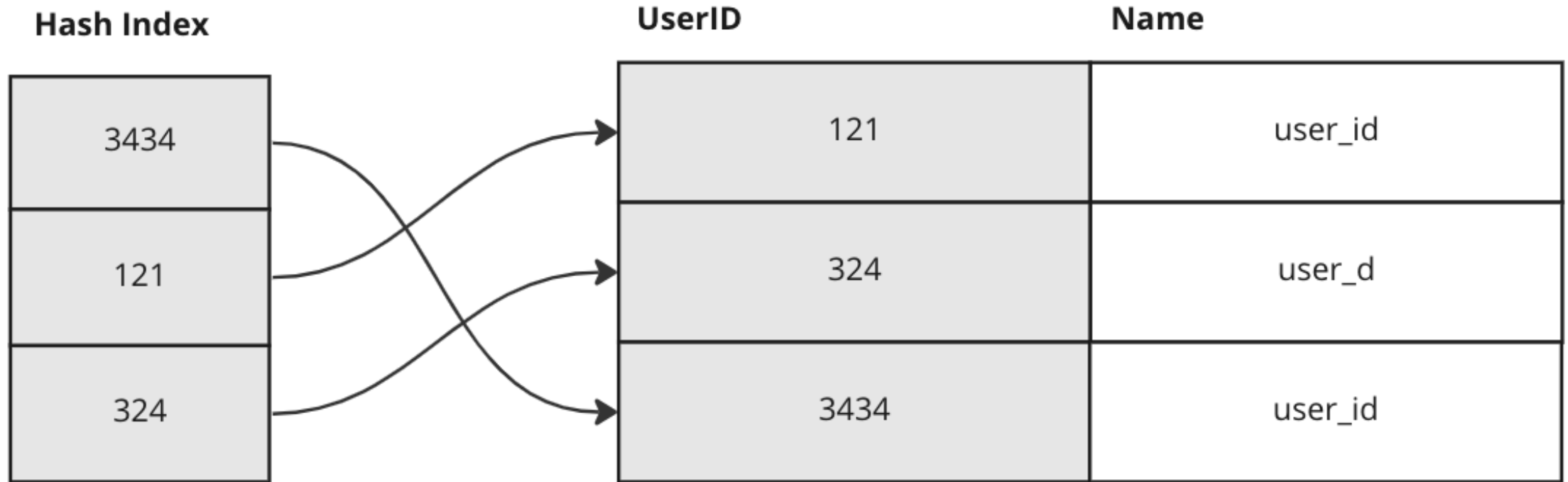


BTree

UserID	Name
50	user_id
168	user_d
200	user_id



Hash



Bitmap

Bitmap Index

3434	1	1
121	1	0
324	0	1

UserID

Smoking

Drinking

121	true	false
324	false	true
3434	true	true

1



Y

Reversed

Word	TextIDs
a	[2]
banana	[2]
is	[0, 1, 2]
it	[0, 1, 2]
what	[0, 1]

ID	Text
0	it is what it is
1	what is it
2	it is a banana

Разряженный индекс

Характеризуется тем, что каждый ключ ассоциируется с определённым указателем *на блок* в отсортированном файле данных, а не с какой-то определенной записью

Покрывающий индекс

Покрывающими индексами называются **некластеризованные индексы**, которые разрешают **один или несколько схожих результатов запроса напрямую, без доступа к базовой таблице и без уточняющих запросов.**

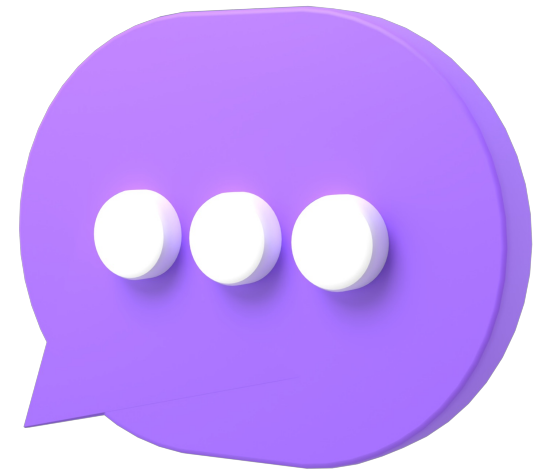
Кластерные и некластерные

При наличии кластерного индекса строки таблицы упорядочены по значению ключа этого индекса. Если в таблице нет кластерного индекса, таблица называется кучей. Некластерный индекс, созданный для такой таблицы, содержит только указатели на записи таблицы

FAQ:

Индексы

- Btree, Hash, Bitmap, Reversed, Spatial
- Кластерные и некластерные
- Покрывающие



Транзакции

ACID

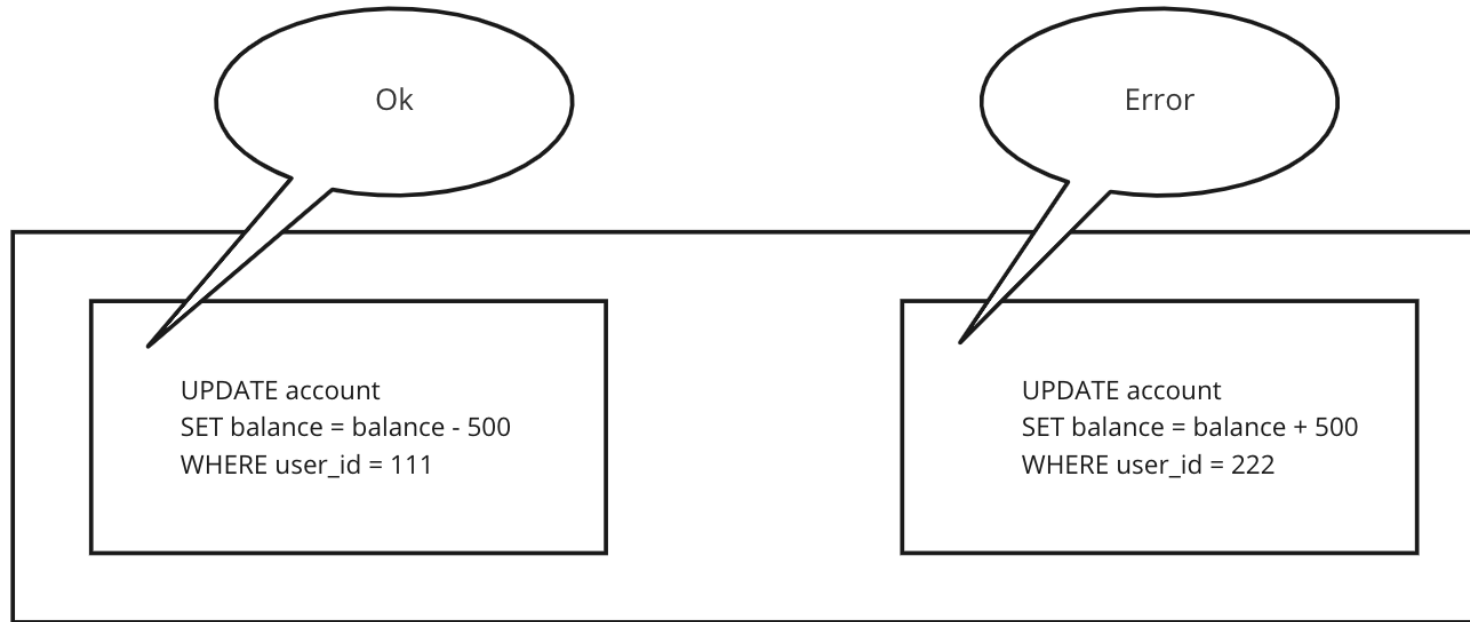
Популярные SQL базы данных, появились как раз на почве ACID.

ACID – это стандарт того, какие гарантии должна давать база данных, чтобы поддерживать транзакции (он не указывает деталей реализации)

Атомарность

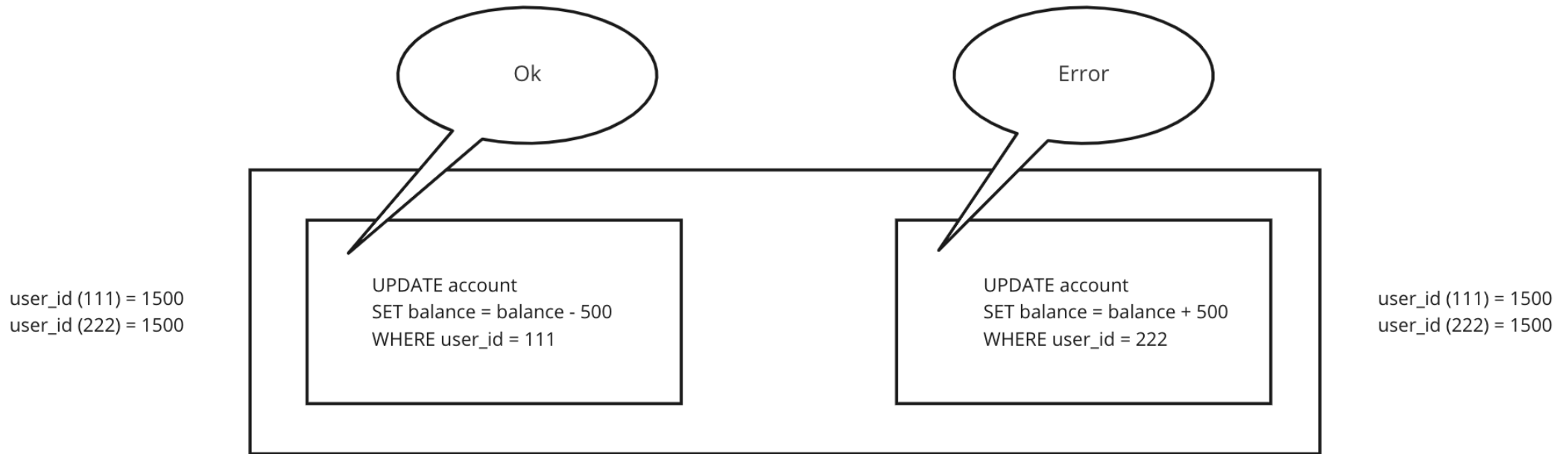
Каждая транзакция базы данных является единым блоком, который использует подход «все или ничего» к выполнению. Если какой-либо оператор в транзакции терпит неудачу, вся транзакция откатывается

user_id (111) = 1500
user_id (222) = 1500

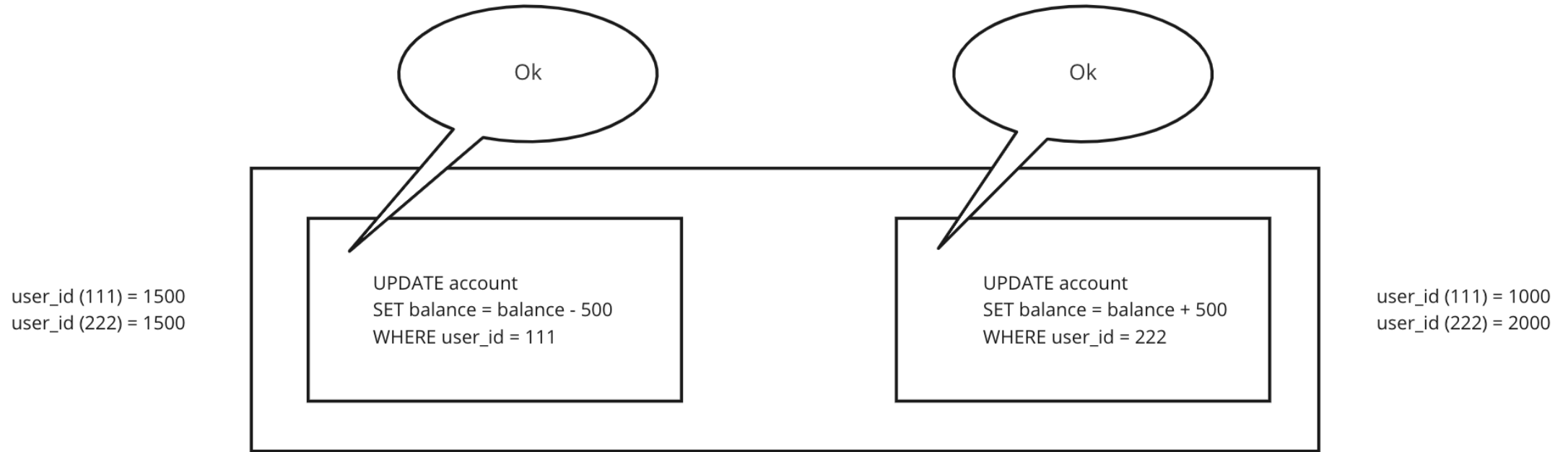


user_id (111) = 1000
user_id (222) = 1500

Rollback



Commit




Согласованность

Различные утверждения относительно данных
(инварианты) должны всегда оставаться справедливыми

Constraint	Description
NOT NULL	values cannot be null
UNIQUE	values cannot match any older value
PRIMARY KEY	used to uniquely identify a row
FOREIGN KEY	references a row in another table
CHECK	validates condition for new value
DEFAULT	set default value if not passed
CREATE INDEX	used to speedup the read process

Deferrable transactions



```
1 CREATE TABLE husbands (  
2   id int PRIMARY KEY,  
3   wife_id int NOT NULL  
4 );  
5  
6 CREATE TABLE wives (  
7   id int PRIMARY KEY,  
8   husband_id int NOT NULL  
9 );  
10  
11 ALTER TABLE husbands ADD CONSTRAINT h_w_fk  
12   FOREIGN KEY (wife_id) REFERENCES Wives;  
13  
14 ALTER TABLE wives ADD CONSTRAINT w_h_f  
15   FOREIGN KEY (husband_id) REFERENCES husbands;
```

Изоляция транзакций

Каждая транзакция происходит до или после каждой другой транзакции, и представление базы данных, которое транзакция видит в своем начале, изменяется только самой транзакцией до ее завершения. Ни одна транзакция не должна видеть промежуточный продукт другой транзакции

Потерянное обновление

video_id (10) = 100

```
UPDATE view  
SET count = count + 1  
WHERE video_id = 10;
```

video_id (10) = 101

video_id (10) = 100

```
UPDATE view  
SET count = count + 1  
WHERE video_id = 10;
```

video_id (10) = 101

Грязное чтение

user_id (111) = 1500

```
START TRANSACTION;
```

```
UPDATE account  
SET balance = balance - 500  
WHERE user_id = 111;
```

```
ROLLBACK;
```

user_id (111) = 1500

user_id (111) = 1500

```
SELECT balance  
FROM account  
WHERE user_id = 111;
```

user_id (111) = 1500

user_id (111) =
1000

Неповторяющееся чтение

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;  
  
UPDATE account  
SET balance = balance + 500  
WHERE user_id = 111;  
  
COMMIT;
```

user_id (111) = 2000
user_id (222) = 2500

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;  
  
SELECT SUM(balance)  
FROM account;  
  
SELECT SUM(balance)  
FROM account;
```

sum = 4000

sum = 4500

user_id (111) = 2000
user_id (222) = 2500

Чтения «фантомов»

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;  
  
INSERT INTO  
  account(user_id, balance)  
VALUES (333, 1000);
```

```
COMMIT;
```

user_id (111) = 2000
user_id (222) = 2500
user_id (333) = 1000

user_id (111) = 1500
user_id (222) = 2500

```
START TRANSACTION;
```

```
SELECT SUM(balance)  
FROM account;
```

sum = 4000

```
SELECT SUM(balance)  
FROM account;
```

sum = 5000

user_id (111) = 2000
user_id (222) = 2500
user_id (333) = 1000

Уровни изоляции

READ_UNCOMMITTED - могут происходить грязные чтения, неповторяющиеся чтения, фантомные чтения и потерянное обновление

READ_COMMITTED - грязные чтения предотвращены, но могут возникать неповторяющиеся чтения, фантомные чтения и потерянные обновления

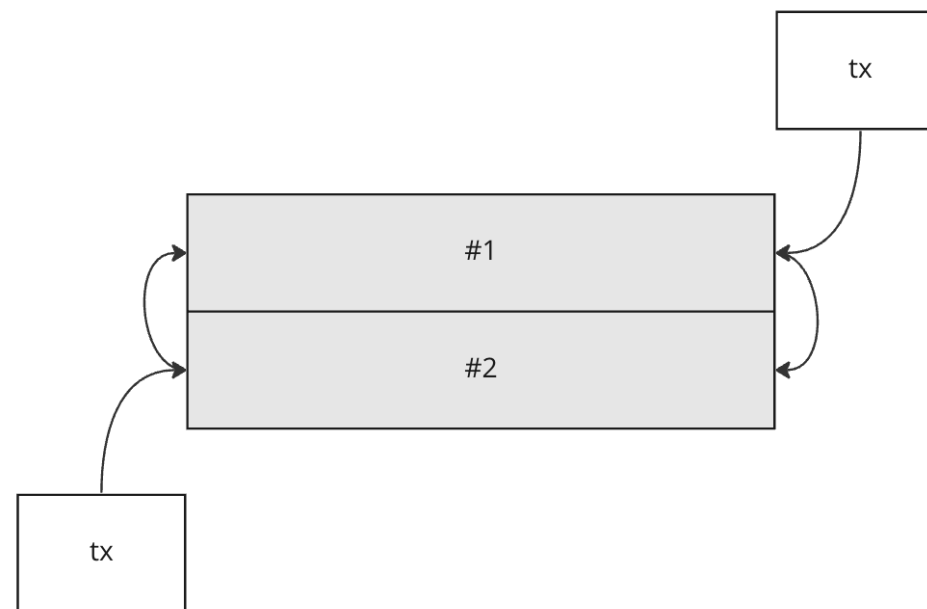
REPEATABLE_READ - грязные чтения, неповторяющиеся чтения и потерянное обновления предотвращены, но могут возникать фантомные чтения

SERIALIZABLE - транзакции полностью изолированы - исключено влияние одной транзакции на другую в момент выполнения

2PL

фаза расширения - запрашиваются все необходимые для определённой транзакции блокировки и никакие блокировки не высвобождаются

фаза сжатия - все полученные на фазе роста блокировки высвобождаются



MVCC

t_xmin	t_xmax	...	user_data
--------	--------	-----	-----------

t_xmin – идентификатор транзакции, создавшей запись

t_xmax – идентификатор транзакции, удалившей запись

Устойчивость

Гарантирует, что после фиксации транзакции в базе данных она постоянно сохраняется с помощью резервных копий и журналов транзакций. В случае сбоя эти механизмы могут использоваться для восстановления зафиксированных транзакций

WAL

Изменения в файлах с данными должны записываться только после того, как эти изменения были занесены в журнал. Записывать страницы данных на диск после подтверждения каждой транзакции нет необходимости, потому что мы знаем, что если случится сбой, то у нас будет возможность восстановить базу данных с помощью журнала

BASE

Как правило, NoSQL базы данных предоставляют ограниченные версии атомарности и изоляции, но им нужно продавать себя в красивой упаковке, поэтому они придумали свою аббревиатуру

Basically Available

Базы данных будут обеспечивать доступность данных, распространяя и реплицируя их по узлам кластера базы данных

Soft State

Из-за отсутствия строгой согласованности значения данных могут меняться со временем. Модель BASE разрывается с концепцией базы данных, которая обеспечивает собственную согласованность, делегируя эту ответственность разработчикам

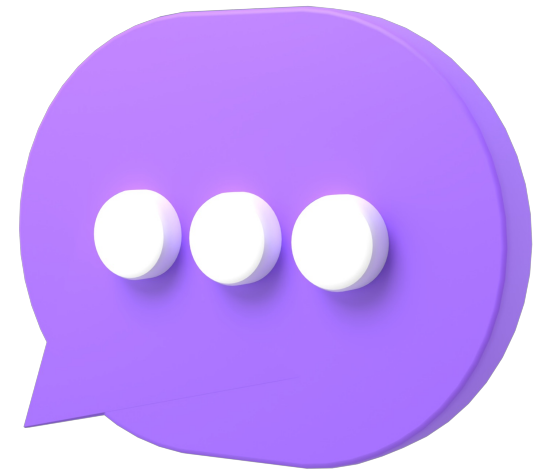
Eventual Consistency

Вы в конце концов увидите действительные данные, но есть вероятность, что ваша транзакция прочитает недействительные значения – то есть, временные, или частично обновлённые, или устаревшие. Возможно, данные станут согласованными в «ленивом» режиме при чтении

FAQ:

Транзакции

- ACID / BASE
- Уровни изоляции
- 2PL / MVCC
- WAL



Объекты баз данных

Хранимые процедуры



```
1 CREATE PROCEDURE us_customers (  
2 LANGUAGE SQL  
3 AS $$  
4     SELECT customer_id, first_name  
5     FROM Customers  
6     WHERE Country = 'USA';  
7 $$;
```



```
1 CALL us_customers( );
```

Триггеры

```
1 CREATE OR REPLACE FUNCTION log_last_name_changes()  
2 RETURNS TRIGGER  
3 LANGUAGE PLPGSQL  
4 AS  
5 $$  
6 BEGIN  
7     IF NEW.last_name <> OLD.last_name THEN  
8         INSERT INTO  
9             employee_audits(employee_id,last_name,changed_on)  
10            VALUES(OLD.id,OLD.last_name,now());  
11     END IF;  
12     RETURN NEW;  
13 END;  
14 $$
```

```
1 CREATE TRIGGER last_name_changes  
2 BEFORE UPDATE  
3 ON employees  
4 FOR EACH ROW  
5 EXECUTE PROCEDURE log_last_name_changes();
```

Materialized View

```
1 CREATE MATERIALIZED VIEW dept_stats
2 REFRESH COMPLETE
3 AS
4 SELECT
5     d.dept_id,
6     d.dept_name,
7     COUNT(e.*) AS emp_count,
8     SUM(e.salary) AS total_salary
9 FROM department d
10 INNER JOIN employee e ON d.dept_id = e.dept_id
11 GROUP BY d.dept_id, d.dept_name;
```

```
1 SELECT dept_id, dept_name, emp_count, total_salary
2 FROM dept_stats;
```

Watch API

```
tx01 $ etcdctl --endpoints=$ENDPOINTS
```

I



gyuho@tx01: ~ 67x6

```
tx01 $ etcdctl --endpoints=$ENDPOINTS _
```

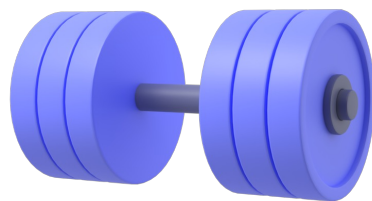

FAQ:

Объекты баз данных

- Триггеры
- Хранимые процедуры
- Materialized View
- Watch API



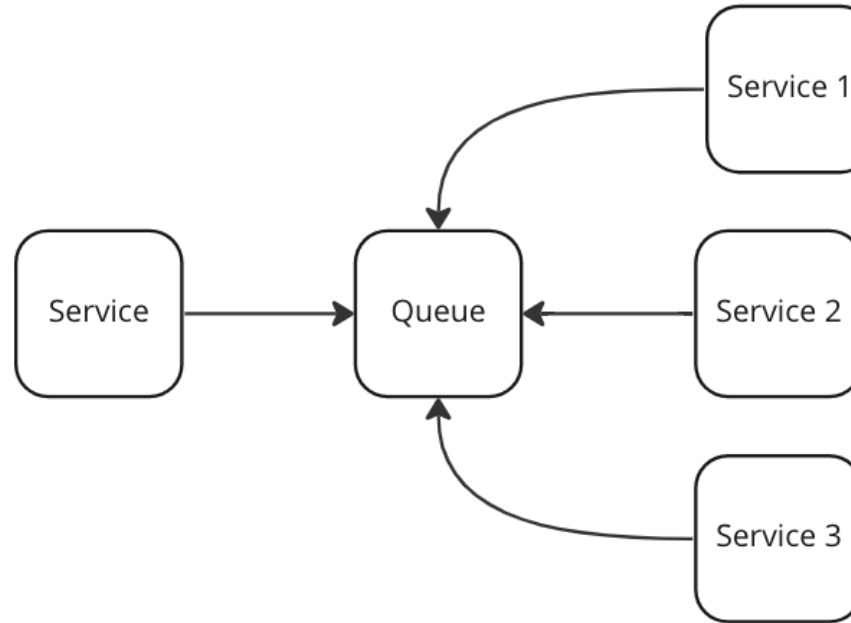
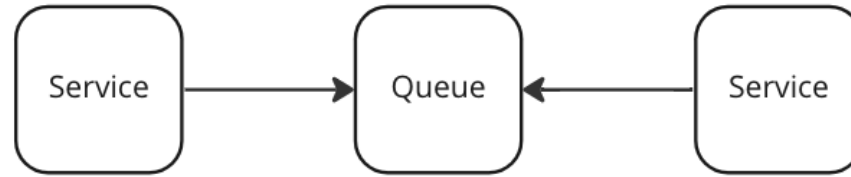
**Как периодически удалять
данные в базе данных?**



Let's practise

Брокеры сообщений

Брокеры сообщений



Брокеры сообщений

Буферизация

Асинхронная связь

Слабое связывание

Масштабируемость

Отказоустойчивость

Понимание потоков данных



push / pull модель

Kafka

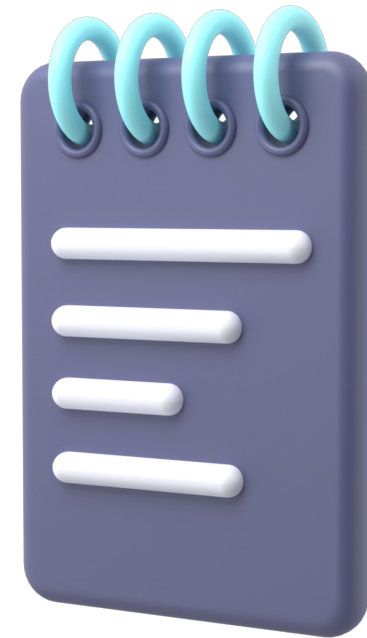
Producer – писатель

Consumer – читатель

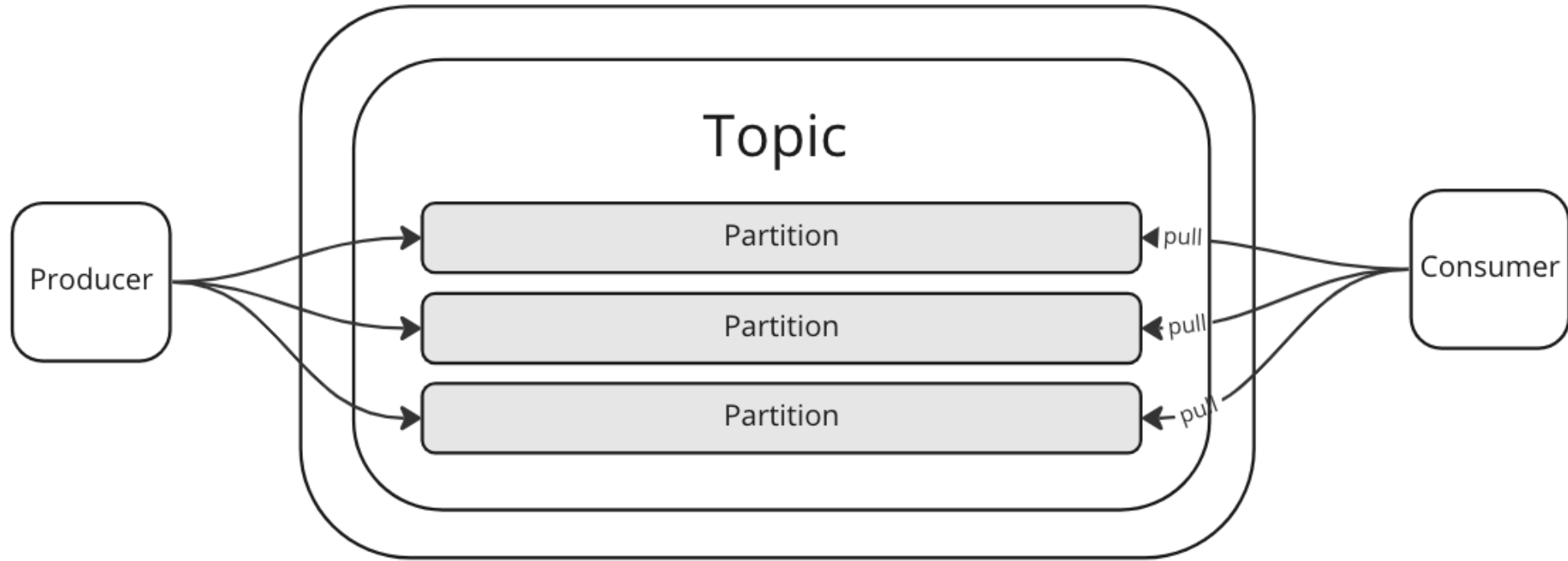
Broker – один узел Kafka

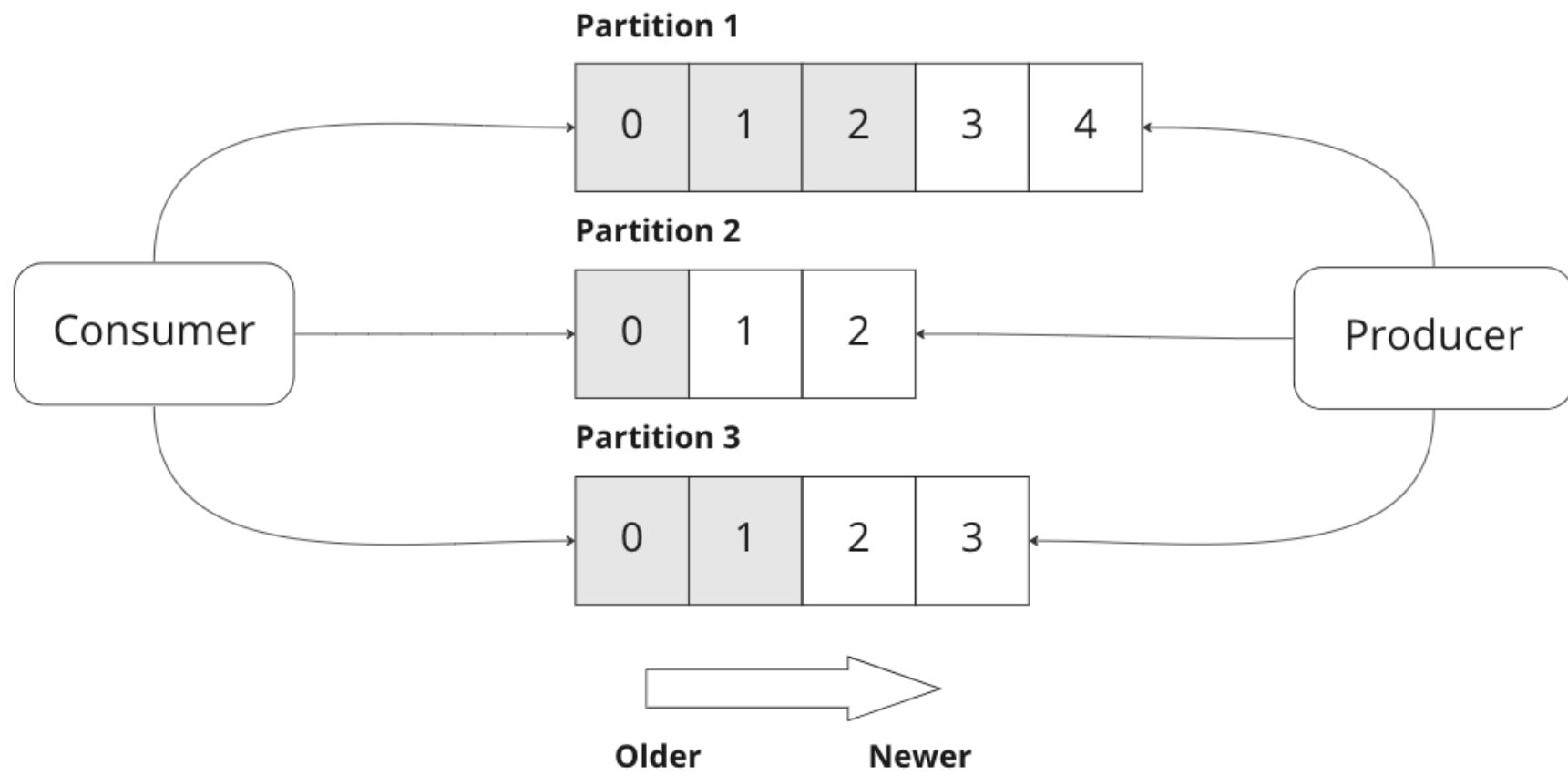
Topic – логическая очередь

Partition – физическая часть очереди

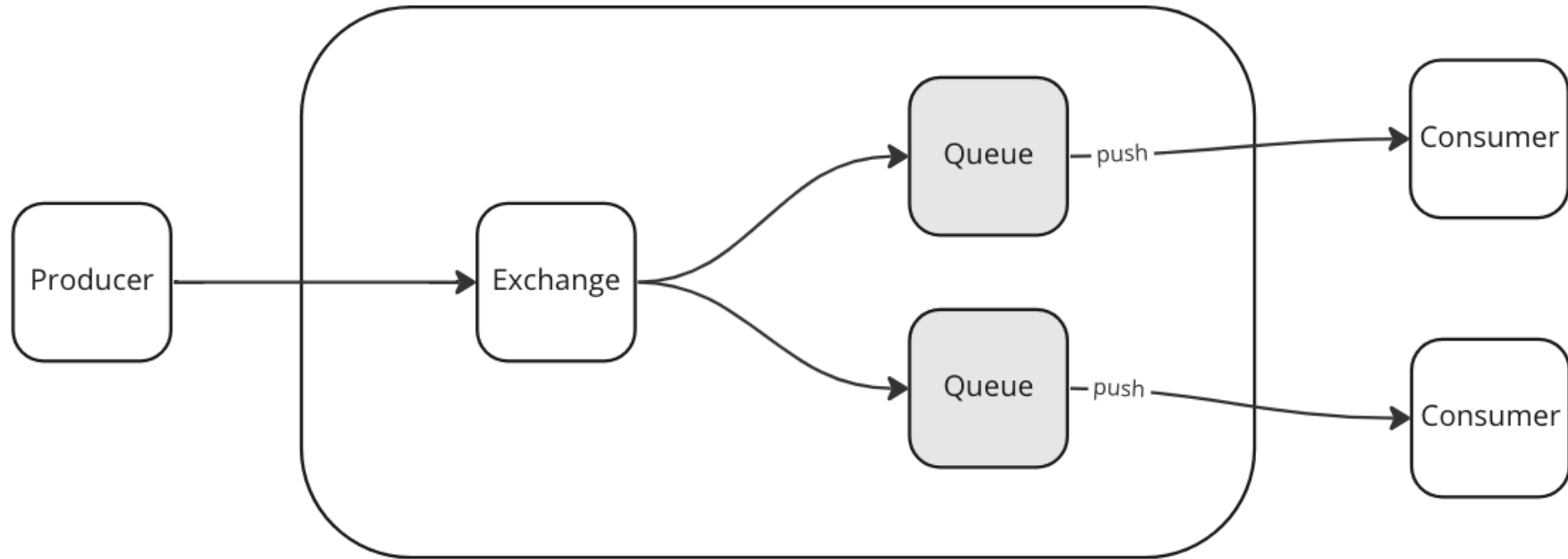


Kafka





RabbitMQ



Data retention

Хранение данных определяет политики управления постоянными данными и записями для соблюдения юридических и бизнес-требований к архивированию данных.

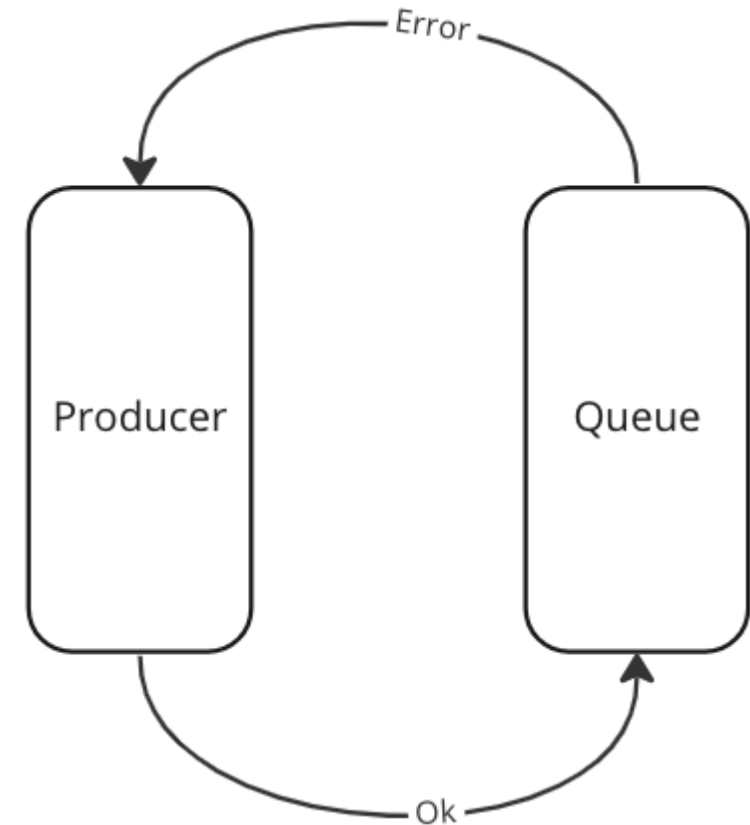


Гарантии доставки

At least once - сообщение будет доставлено хотя бы один раз

At most once - сообщение будет доставлено не более одного раза

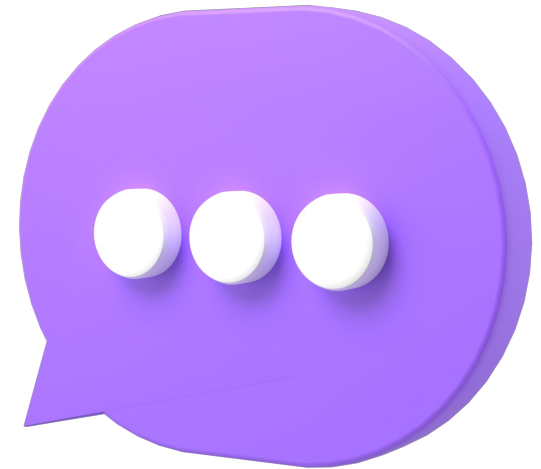
Exactly once - сообщение будет доставлено ровно один раз



FAQ:

Брокеры сообщений

- RabbitMQ / Kafka
- Data retention
- Гарантии доставки

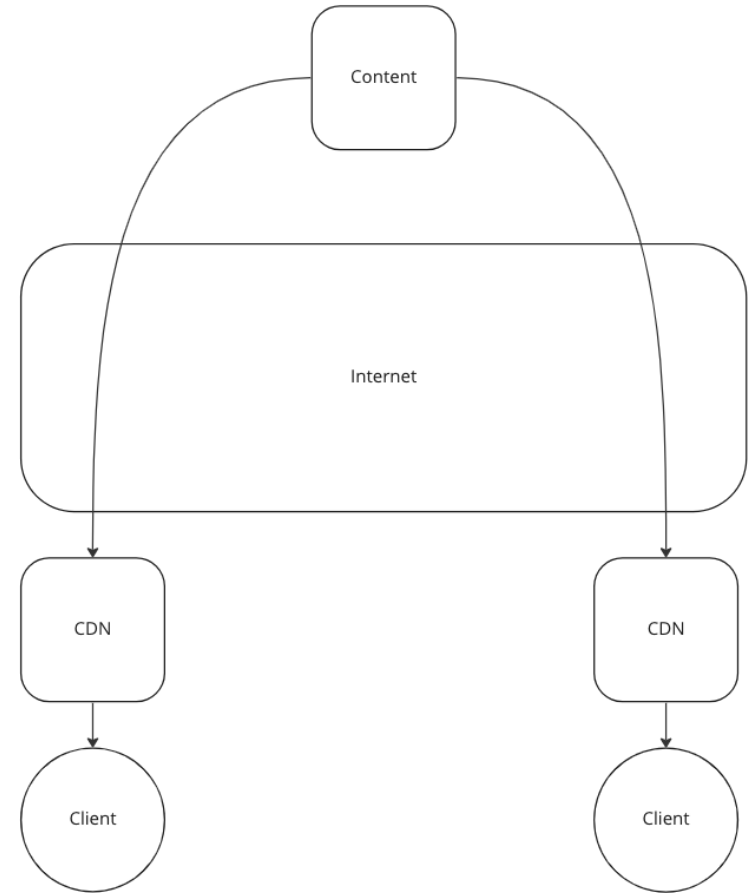
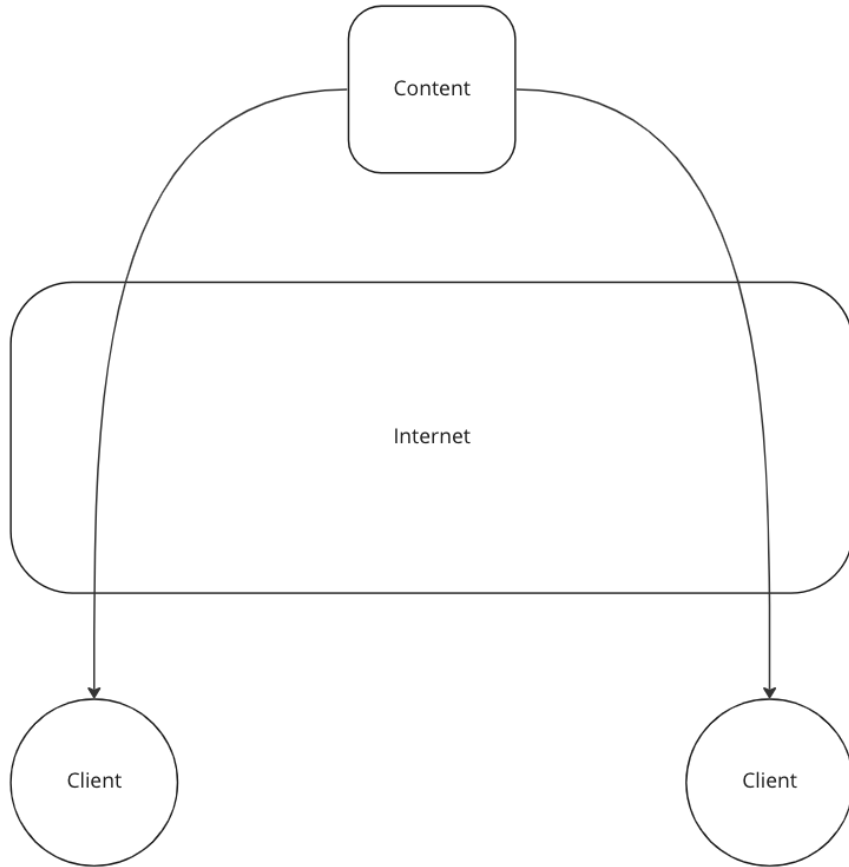


Альтернативные способы хранения данных

Хранение на клиенте



CDN



CDN

- **Исходный (origin)**, на котором размещен запрашиваемый сайт, а также связанный с ним визуальный, музыкальный, видеоконтент
- **PoP (point of presence)** – точка присутствия вспомогательных серверов. Их сеть размещается в различных регионах
- **Proху-сервер** – это промежуточное звено между пользователем и исходным сервером. Он отвечает за перенаправление, оптимизацию и преобразование передаваемого трафика.

FAQ:

- CDN
- Хранение на клиенте



Домашнее задание