K9db: Privacy-Compliant Storage For Web Applications By Construction

Kinan Dak Albab ● Ishan Sharma ● Justus Adam ●
Benjamin Kilimnik ● Aaron Jeyaraj ● Raj Paul ●
Artem Agvanian ● Leonhard Spiegelberg ● Malte Schwarzkopf

ETOS

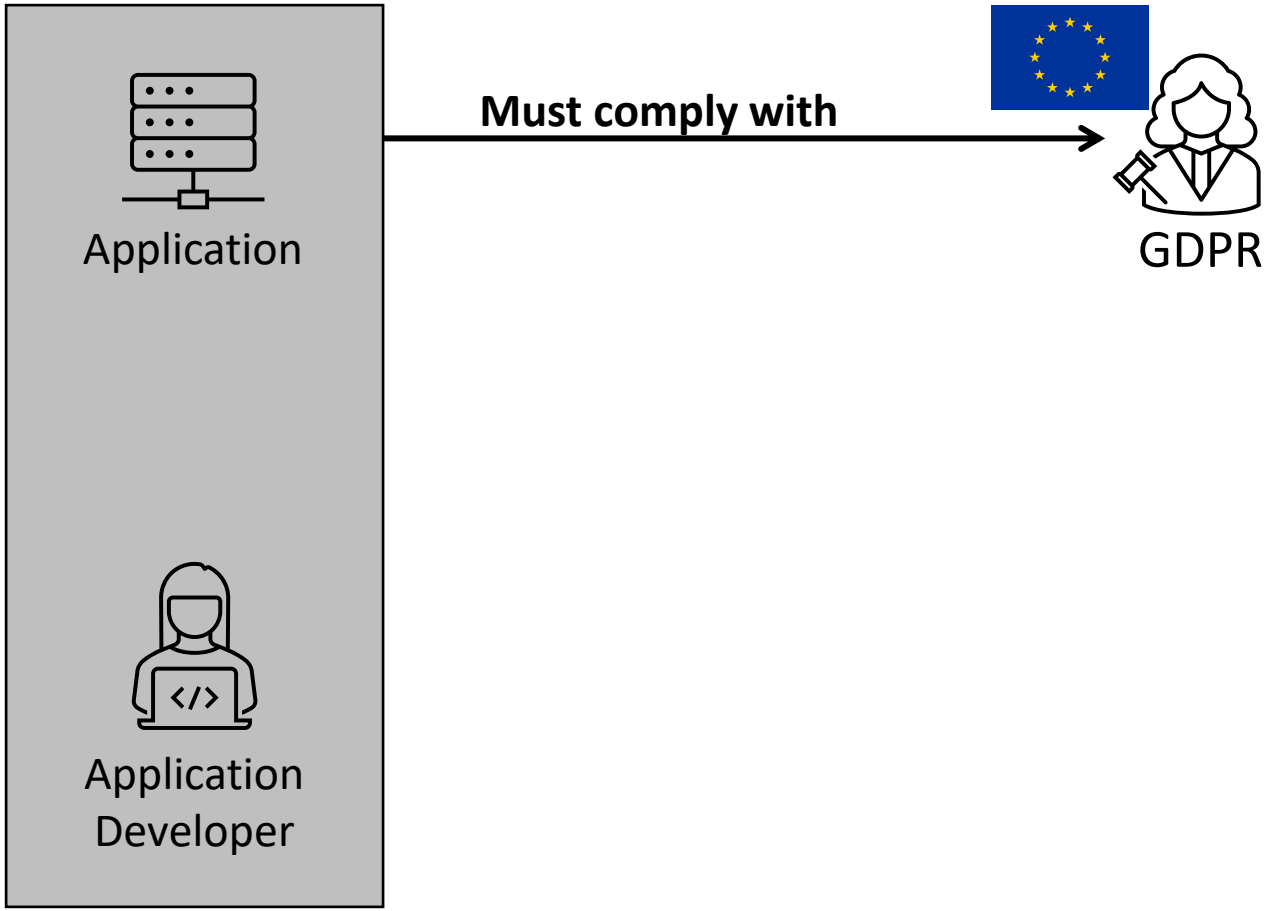
BROWN

Privacy laws are important

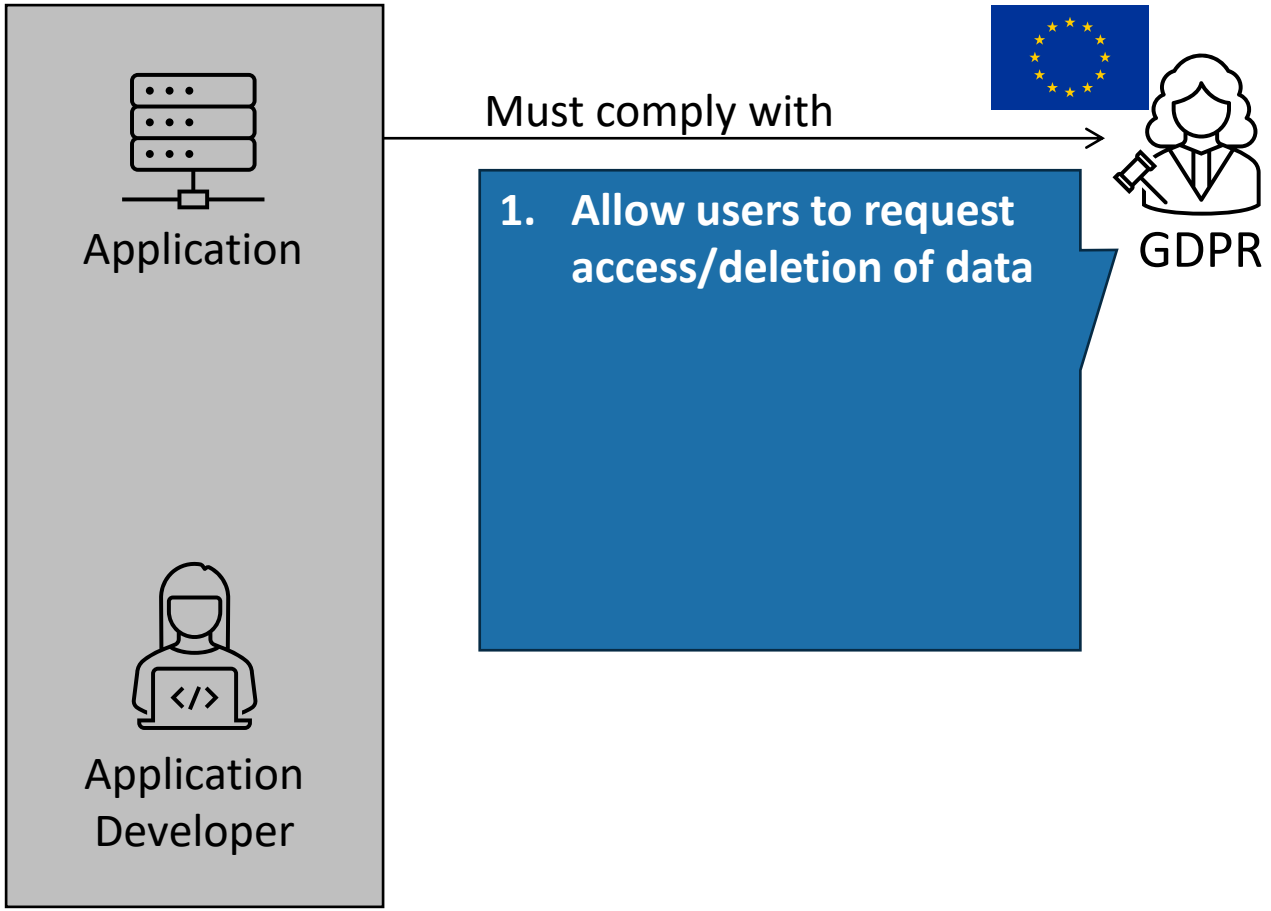
GDPR

CCPA

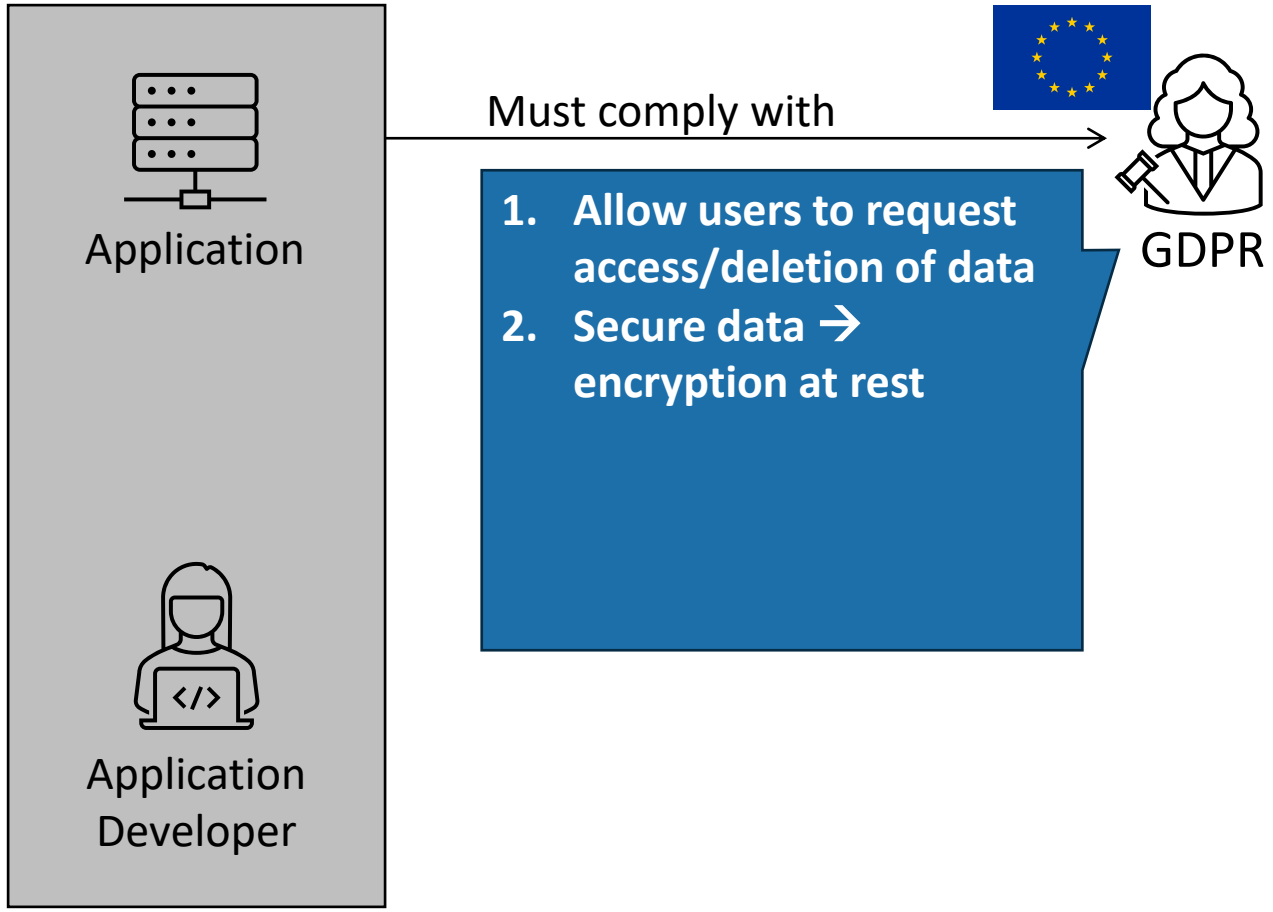
Developers must ensure their applications comply with Privacy Laws



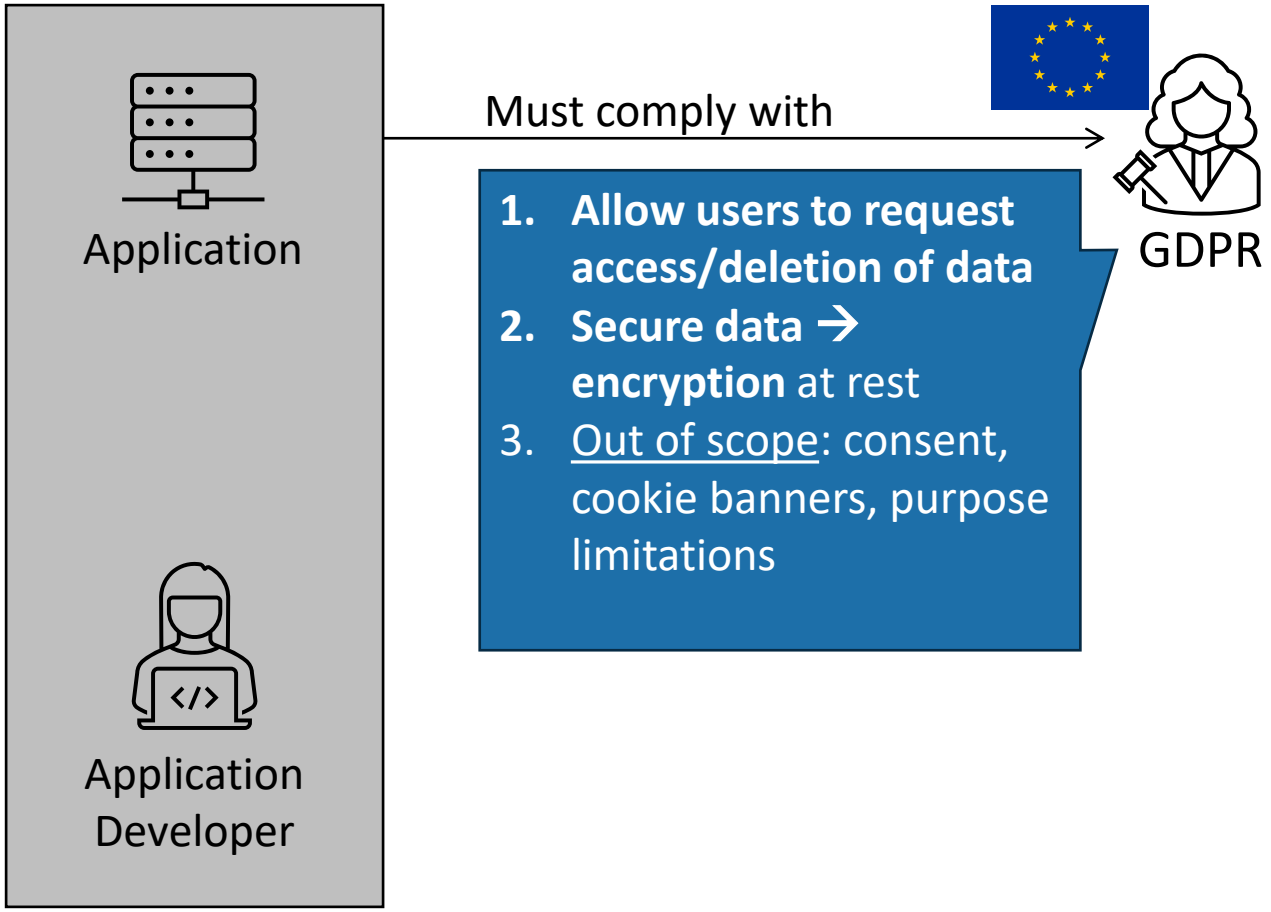
Developers must ensure their applications comply with Privacy Laws



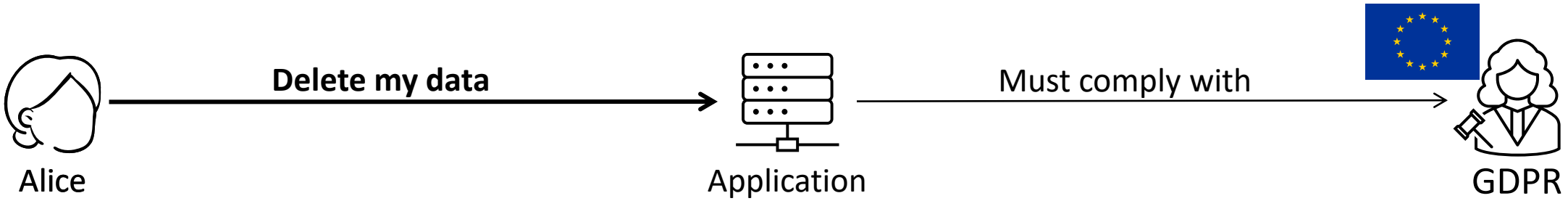
Developers must ensure their applications comply with Privacy Laws

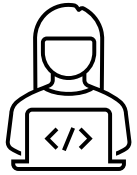


Developers must ensure their applications comply with Privacy Laws



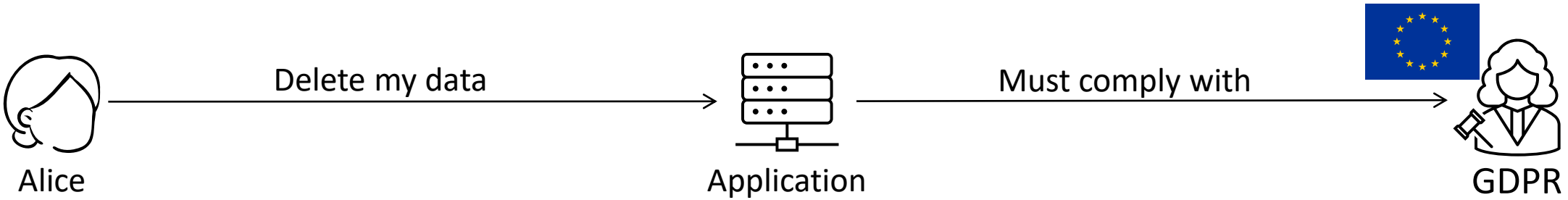
Compliance is challenging for developers



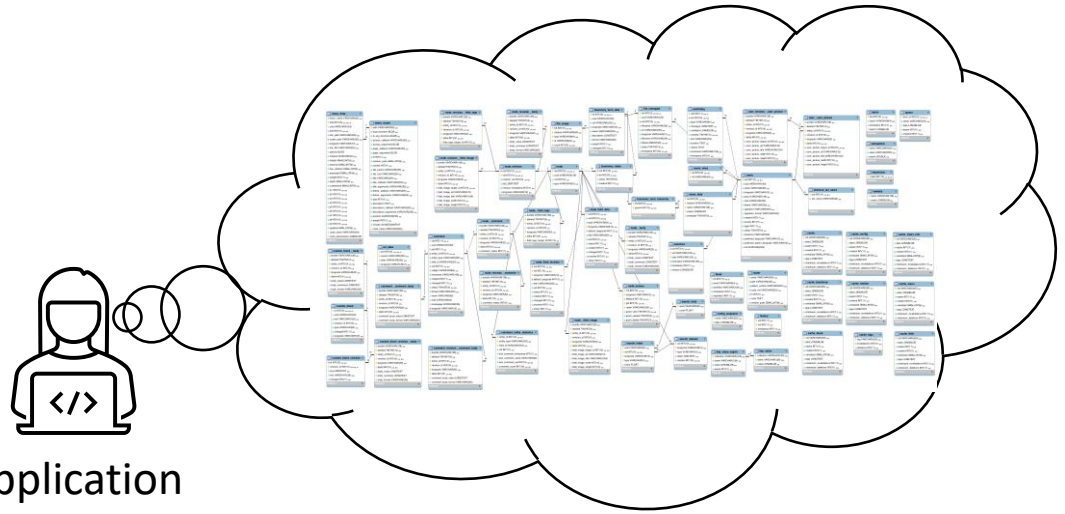


Application Developer

Compliance is challenging for developers

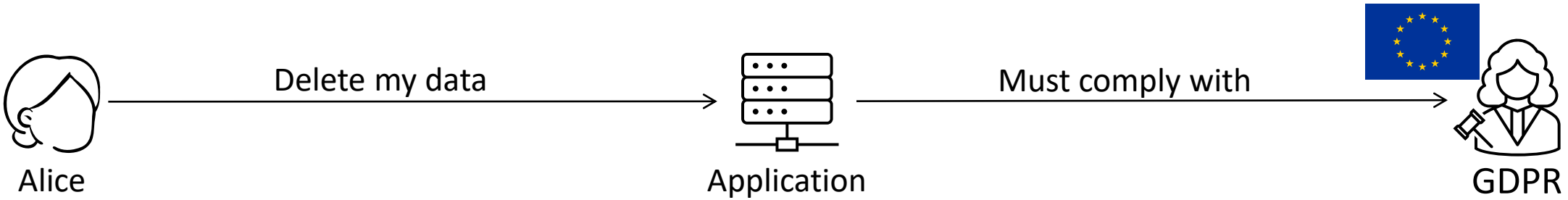


1. **Identify Alice's data in complex schema**

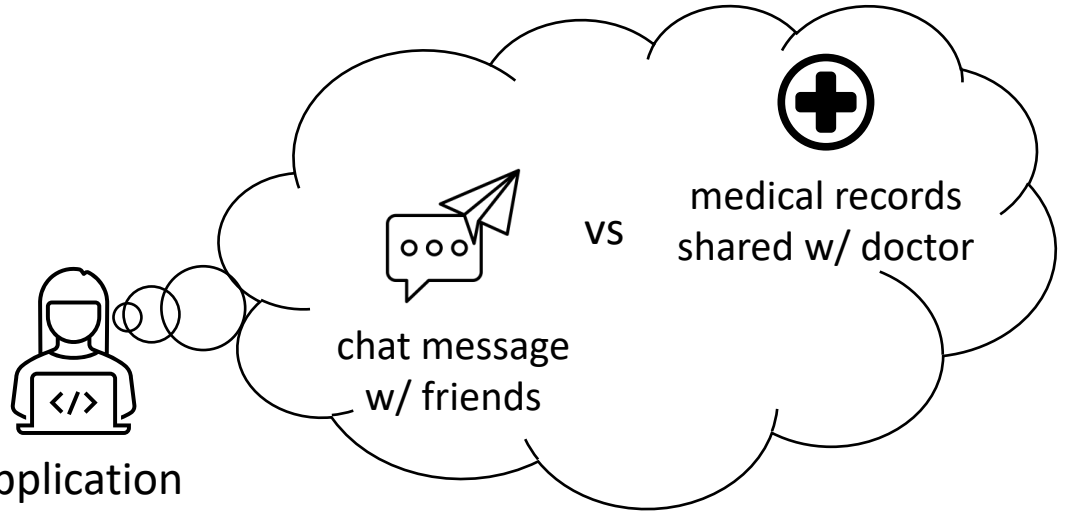


Application Developer

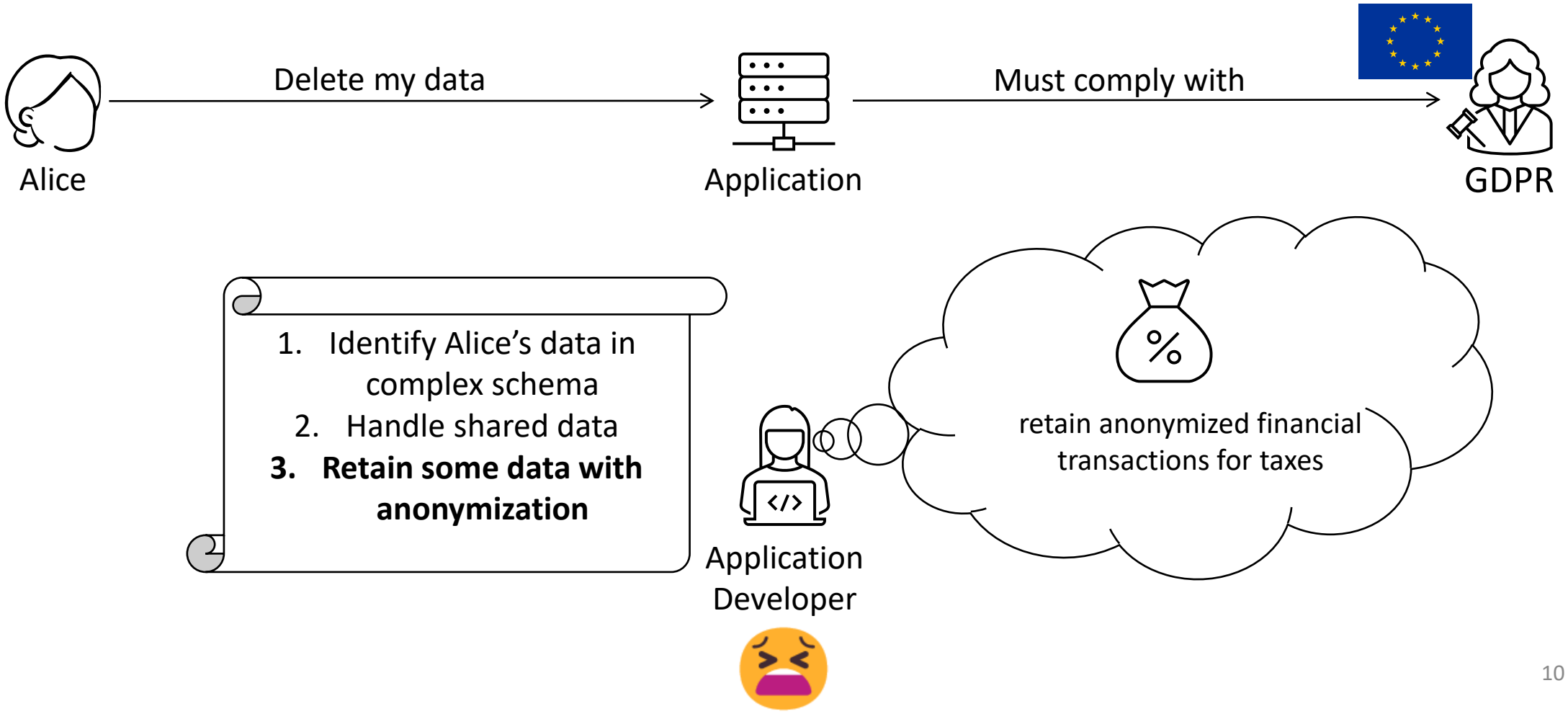
Compliance is challenging for developers



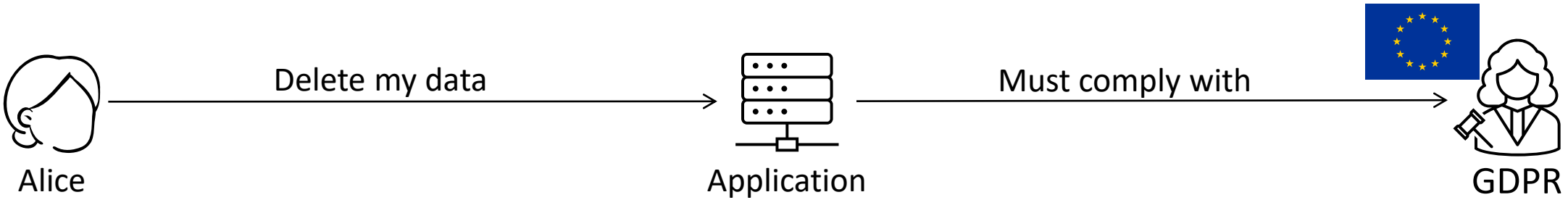
1. Identify Alice's data in complex schema
2. **Handle shared data**



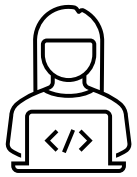
Compliance is challenging for developers



Compliance is challenging for developers

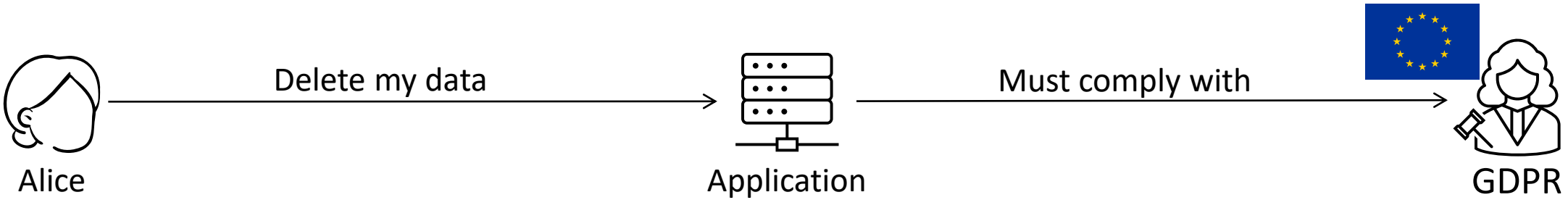


1. Identify Alice's data in complex schema
2. Handle shared data
3. Retain some data with anonymization
4. **Update cache, backups**

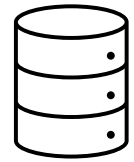


Application Developer

We need a better way…

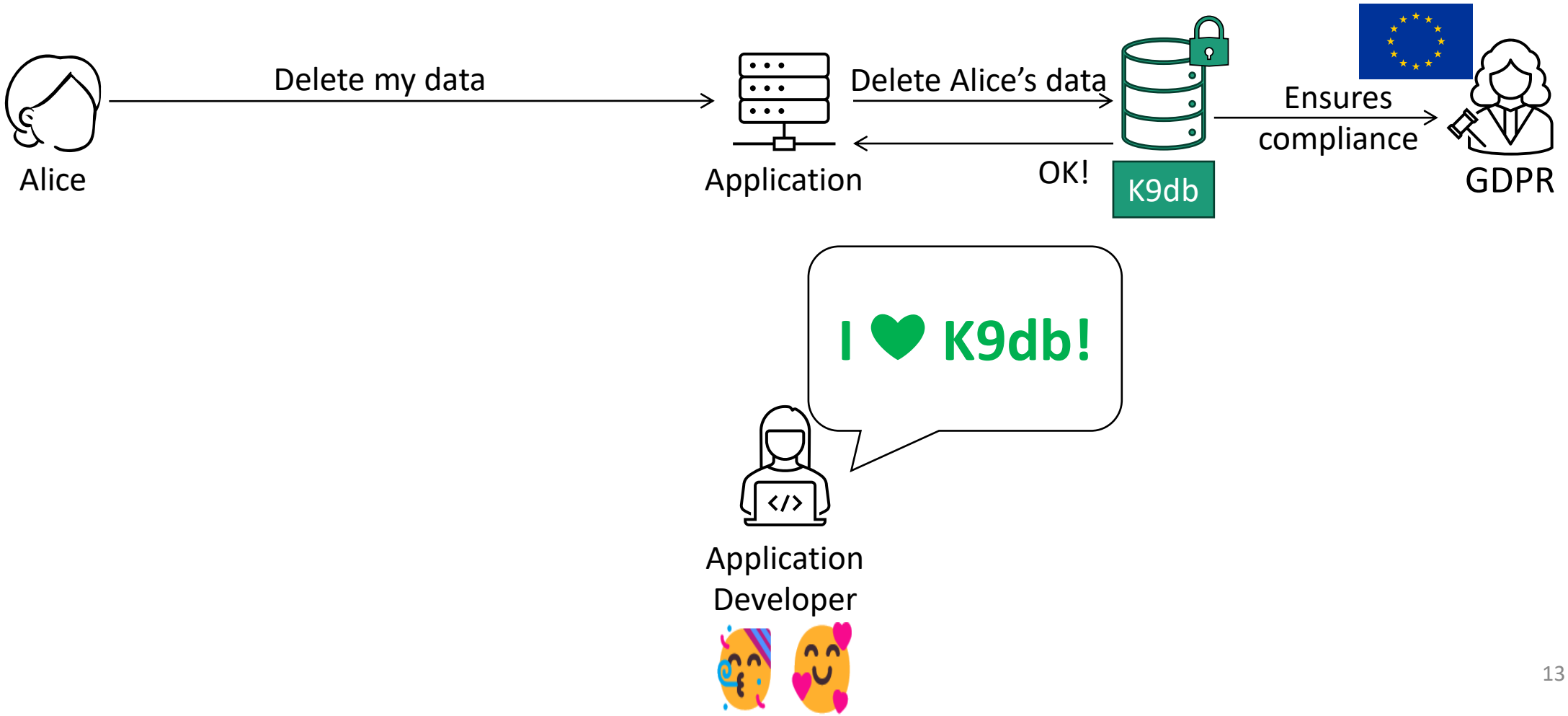


1. Identify Alice's data in complex schema
2. Handle shared data
3. Retain some data with anonymization
4. Update cache, backups



Traditional DBs are unhelpful

K9db: compliance by construction



K9db Goals

- Help developers get compliance right
- Low developer effort
- Performance comparable to widely-used SQL databases

K9db Goals

- Help developers get compliance right
- Low developer effort
- Performance comparable to widely-used SQL databases

Assumption:
Developers are honest but fallible
Penalties deter malicious behavior

Challenges

Capture application-specific compliance policy

Correctly handle access/deletion requests and enforce compliance invariants

Maintain good performance

Challenges

Capture application-specific compliance policy

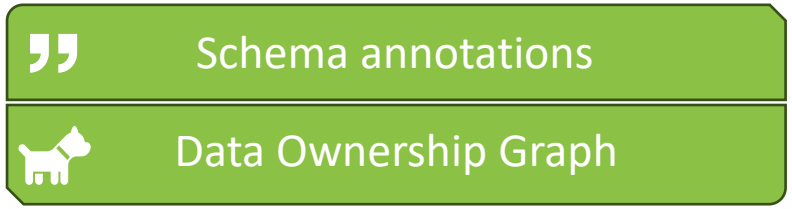
- Schema annotations
- Data Ownership Graph

Correctly handle access/deletion requests and enforce compliance invariants

Maintain good performance

Challenges

Capture application-specific compliance policy



Correctly handle access/deletion requests and enforce compliance invariants

Maintain good performance

Application scenario: direct messages

```sql
CREATE TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT REFERENCES users(name),
  receiver TEXT REFERENCES users(name)
);
```

Application scenario: direct messages

```sql
CREATE TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT REFERENCES users(name),
  receiver TEXT REFERENCES users(name)
);
```

users
name
Alice
Bob
Carol

Application scenario: direct messages

```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT REFERENCES users(name),
  receiver TEXT REFERENCES users(name)
);
```

users
name
Alice
Bob
Carol

Application scenario: direct messages

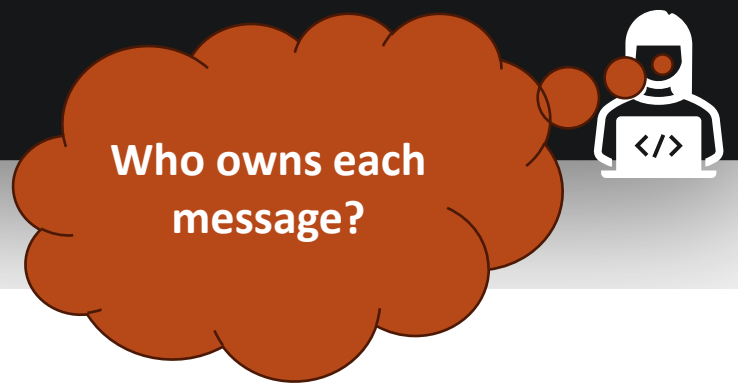
```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT REFERENCES users(name),
  receiver TEXT REFERENCES users(name)
);
```

Alice requested deletion

messages		
body	sender	receiver
…	Alice	Bob
…	Alice	Carol
…	Bob	Carol

Compliance policy is application specific

```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT REFERENCES users(name),
  receiver TEXT REFERENCES users(name)
);
```



Alice requested deletion

messages		
body	sender	receiver
…	Alice	Bob
…	Alice	Carol
…	Bob	Carol

Compliance policy is application specific

```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT REFERENCES users(name),
  receiver TEXT REFERENCES users(name)
);
```

Alice requested deletion

messages		
body	sender	receiver
…	Alice	Bob
…	Alice	Carol
…	Bob	Carol

Developers express policy via schema annotations

```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT OWNED_BY users(name),
  receiver TEXT OWNED_BY users(name)
);
```

Alice requested deletion

→ K9db retains messages because they are shared with others

messages		
body	sender	receiver
…	Alice	Bob
…	Alice	Carol
…	Bob	Carol

Developers express policy via schema annotations

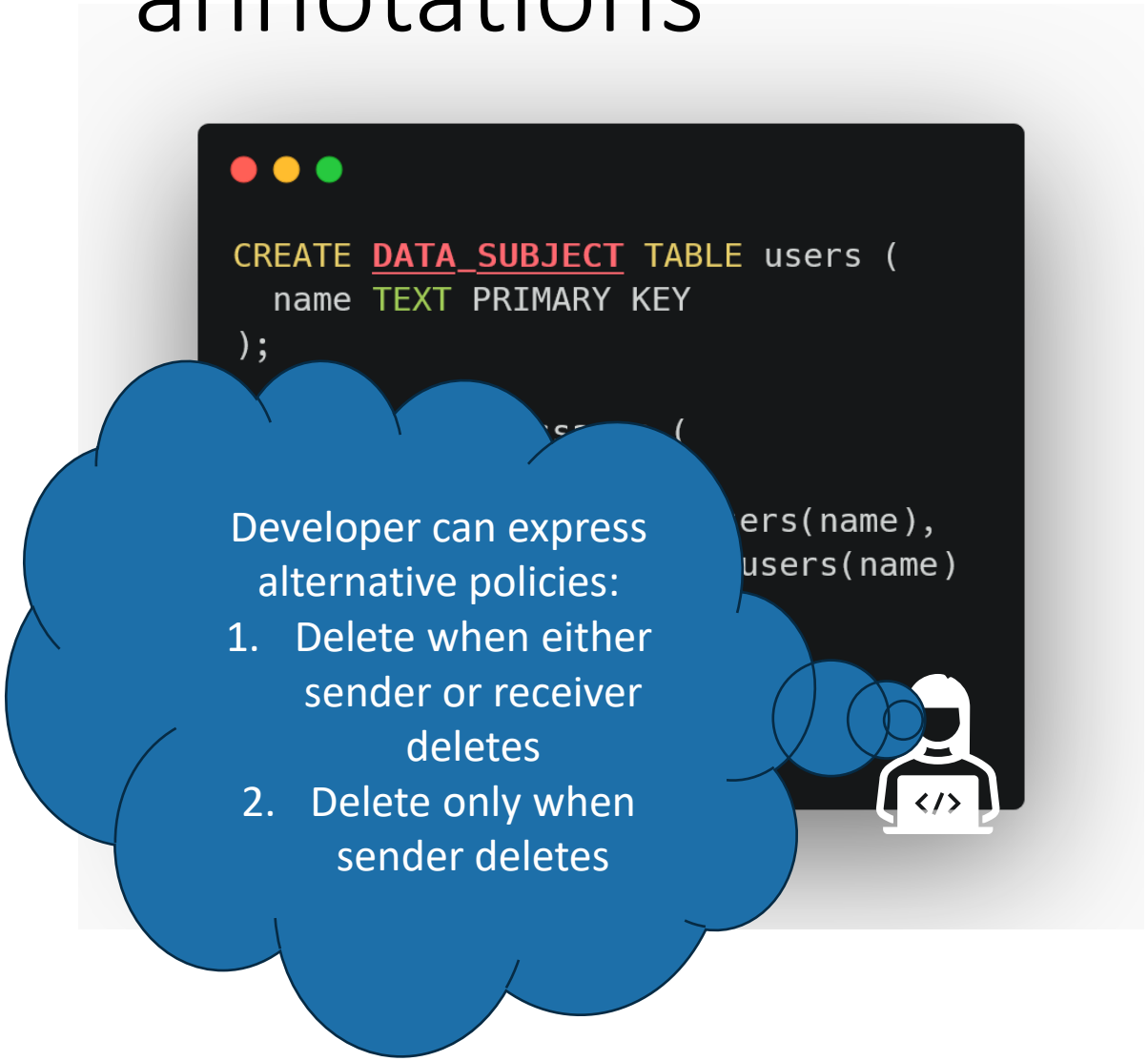
```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT OWNED_BY users(name),
  receiver TEXT OWNED_BY users(name)
);
```

Bob requested deletion

→ K9db only deletes the first message because both Alice and Bob are gone

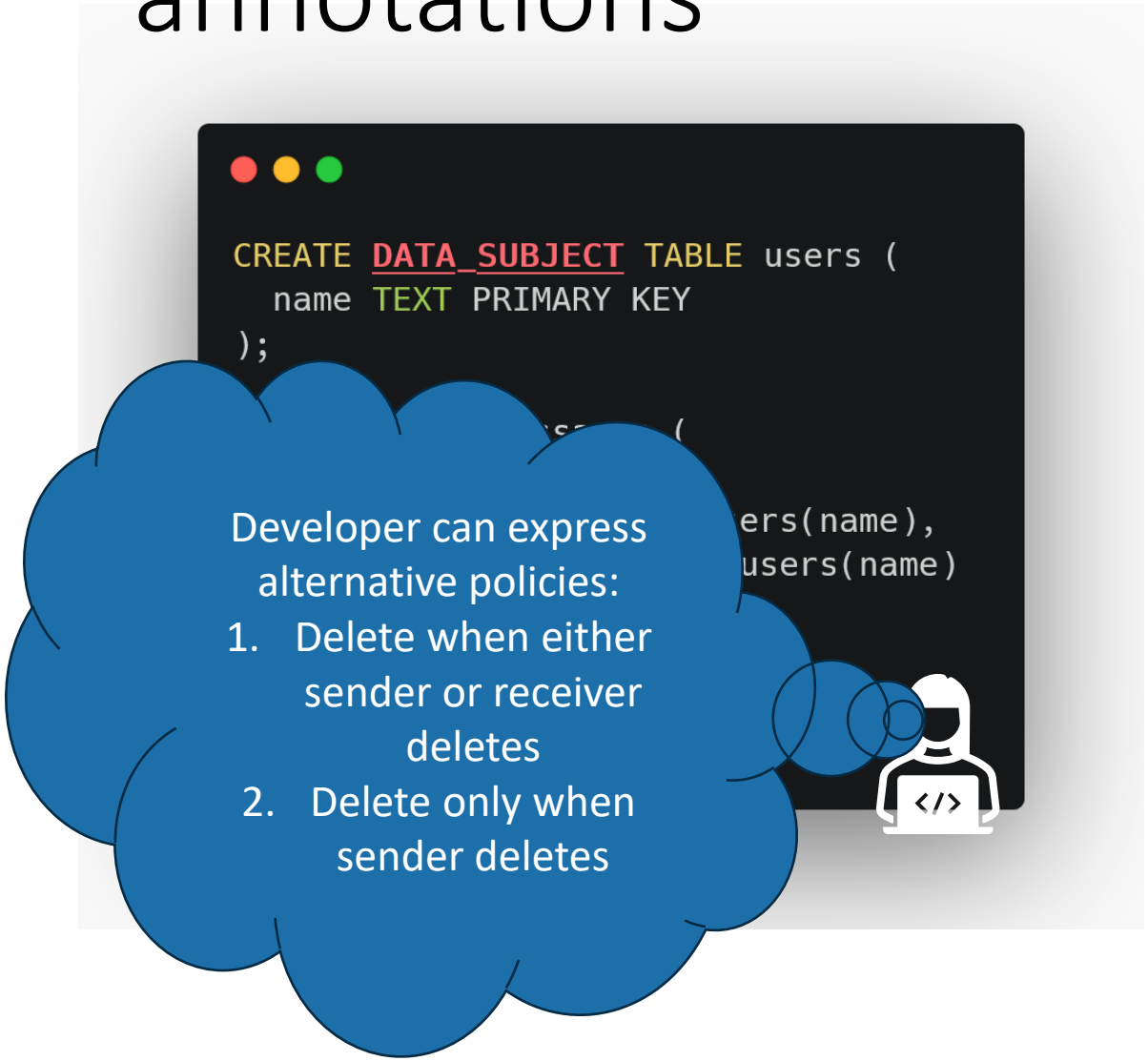
messages		
body	sender	receiver
~~…~~	~~Alice~~	~~Bob~~
…	Alice	Carol
…	Bob	Carol

Developers express policy via schema annotations

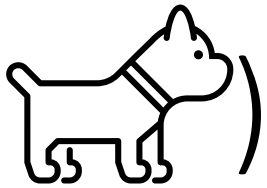


- K9db provides more schema annotations (see paper)

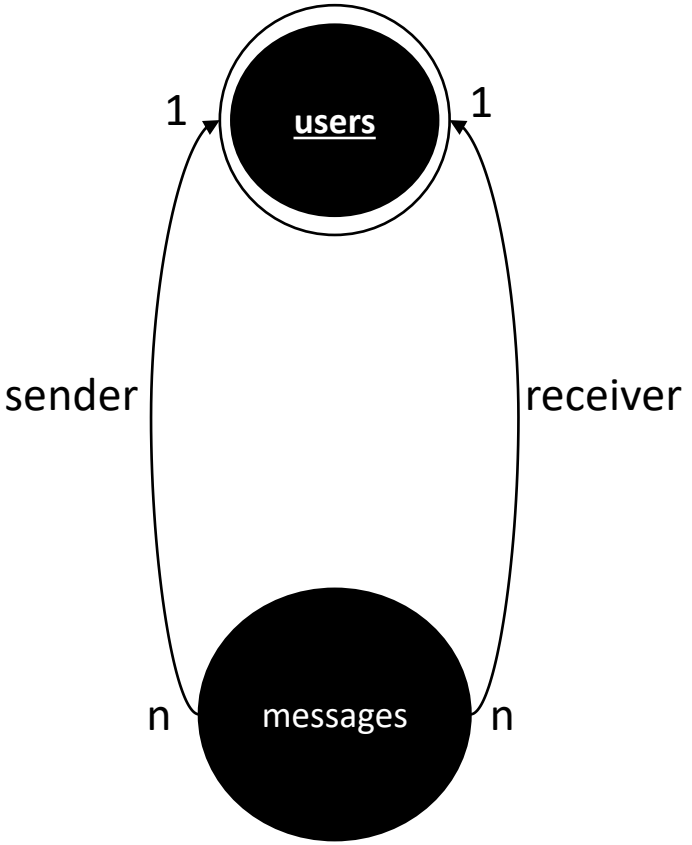
Developers express policy via schema annotations

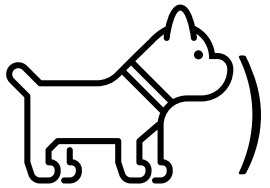


- K9db provides more schema annotations (see paper)
- K9db provides **EXPLAIN COMPLIANCE** command to help developers reason about their policy and annotations

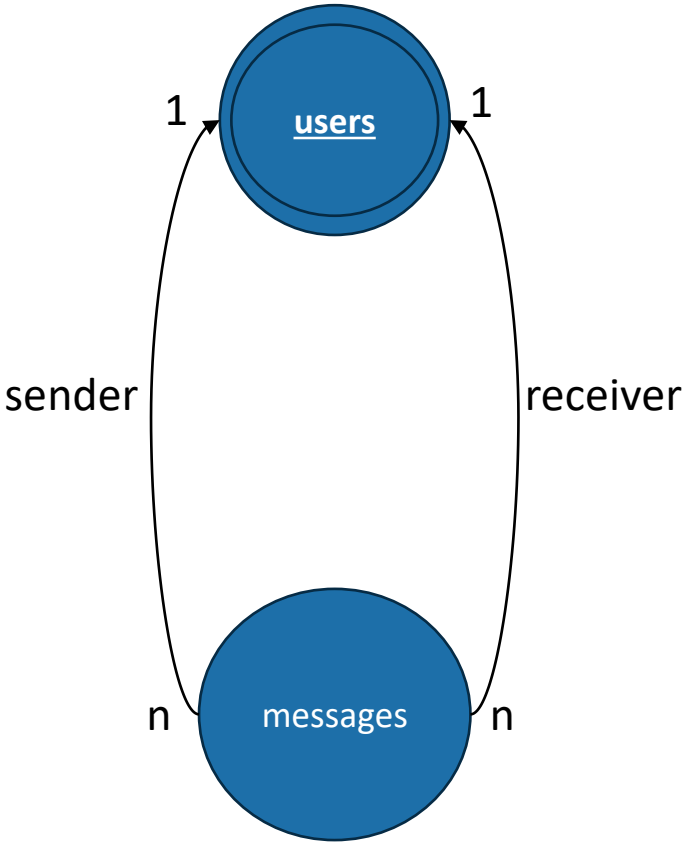
Data Ownership Graph (DOG )

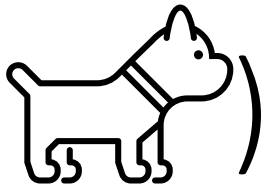
```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT OWNED_BY users(name),
  receiver TEXT OWNED_BY users(name)
);
```



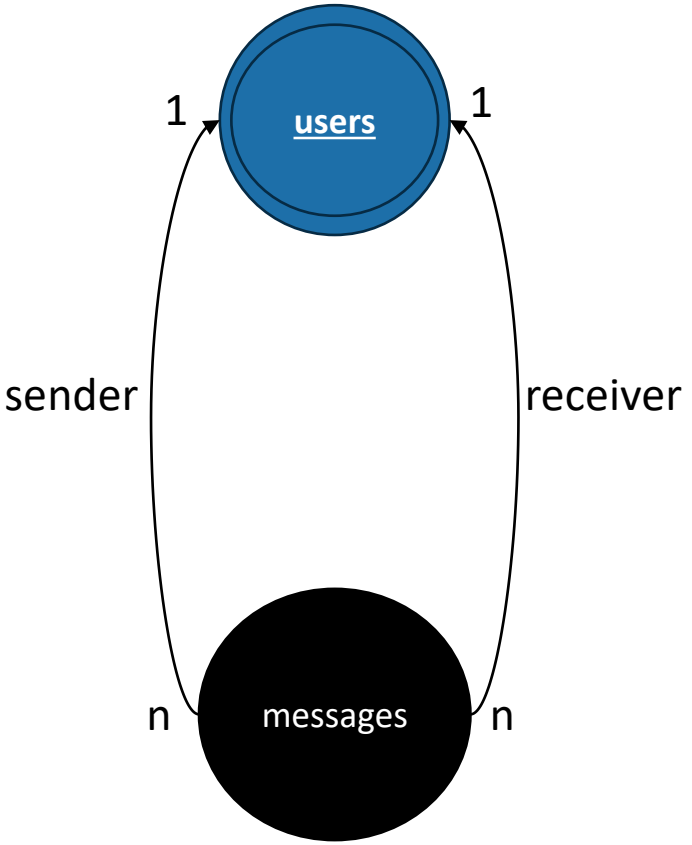
Data Ownership Graph (DOG )

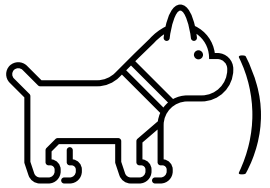
```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT OWNED_BY users(name),
  receiver TEXT OWNED_BY users(name)
);
```



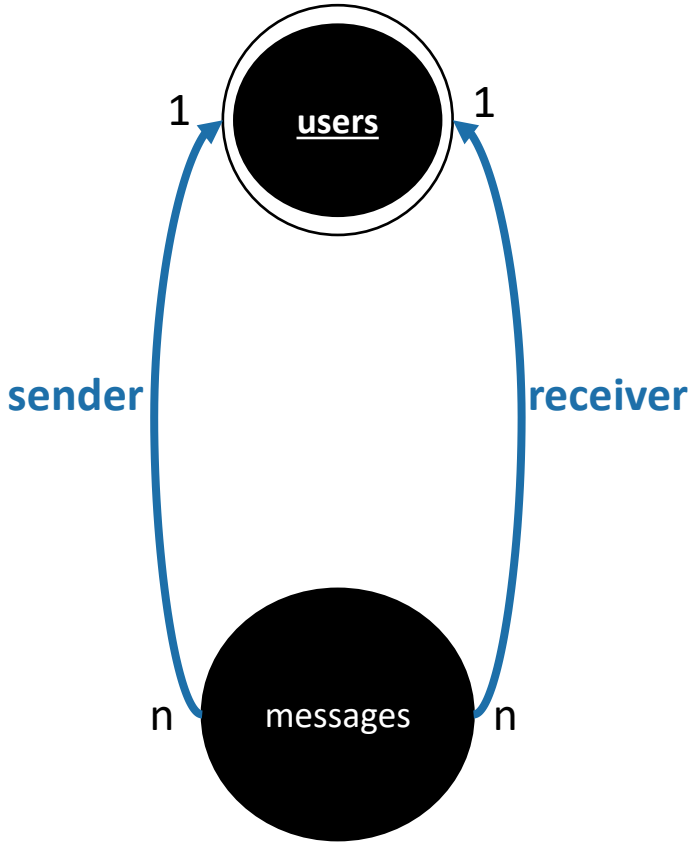
Data Ownership Graph (DOG )

```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT OWNED_BY users(name),
  receiver TEXT OWNED_BY users(name)
);
```

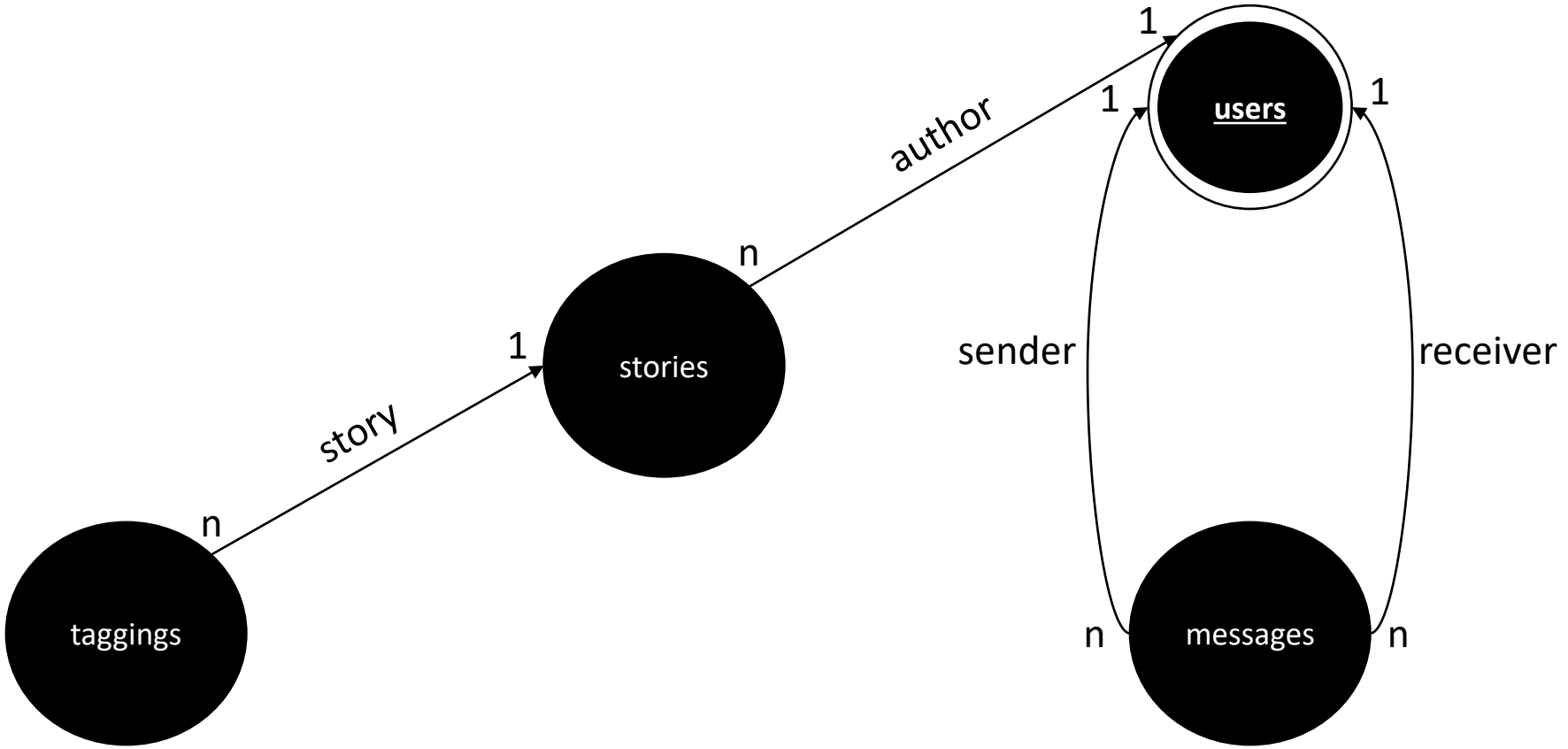


Data Ownership Graph (DOG )

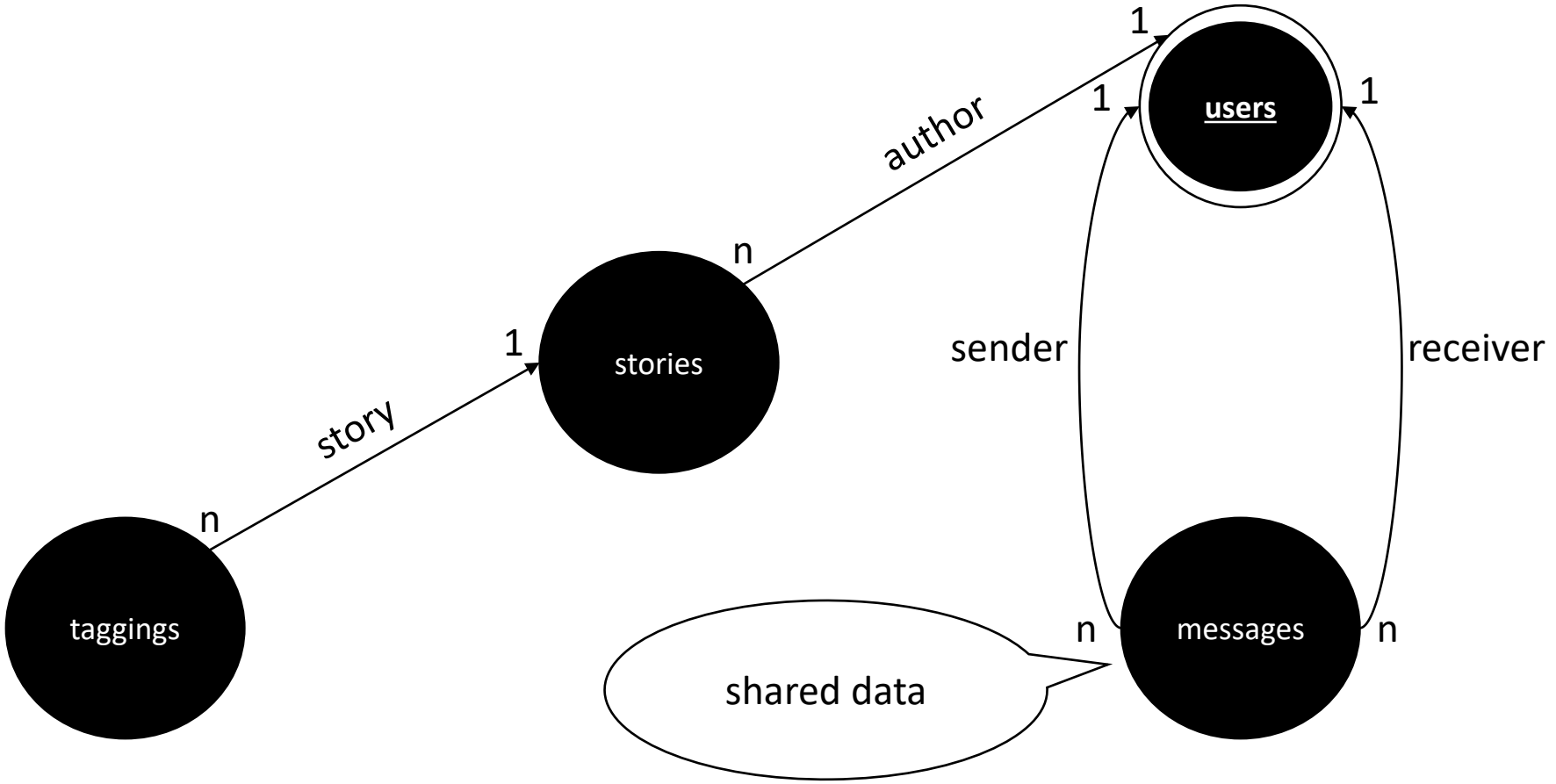
```
CREATE DATA_SUBJECT TABLE users (
  name TEXT PRIMARY KEY
);

CREATE TABLE messages (
  body TEXT,
  sender TEXT OWNED_BY users(name),
  receiver TEXT OWNED_BY users(name)
);
```



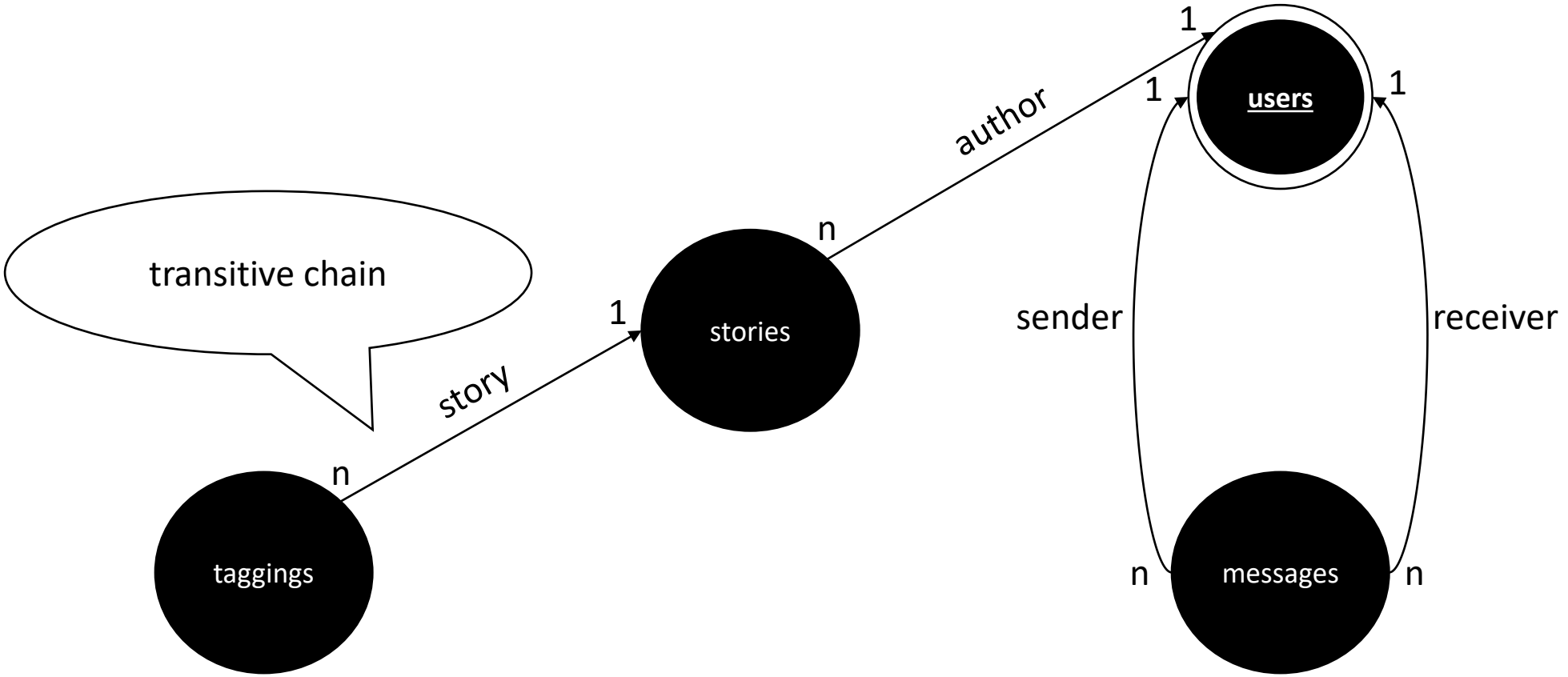
Data Ownership Graph (DOG)

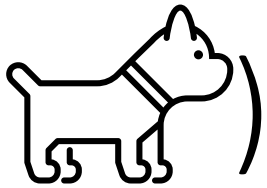


Data Ownership Graph (DOG)

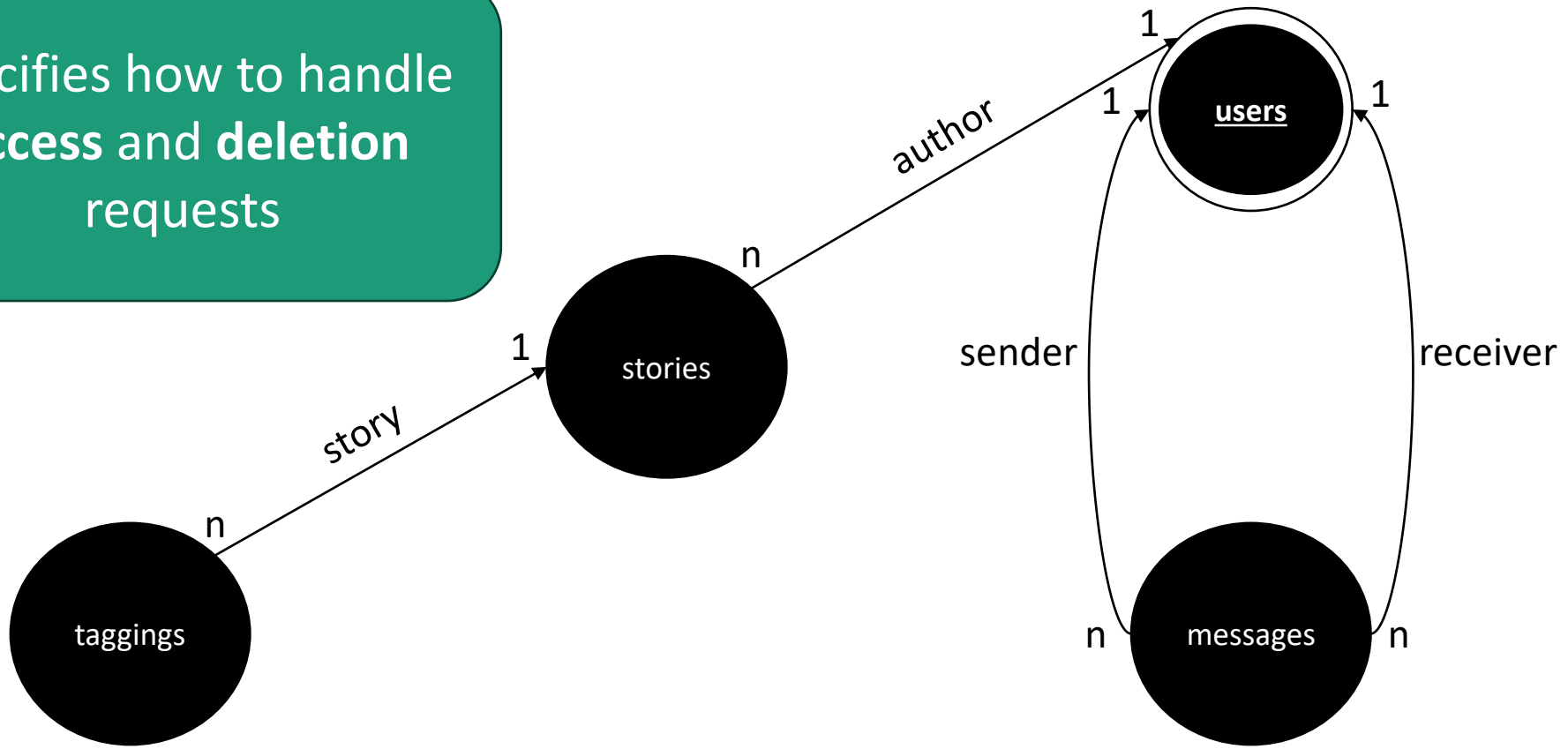


Data Ownership Graph (DOG)

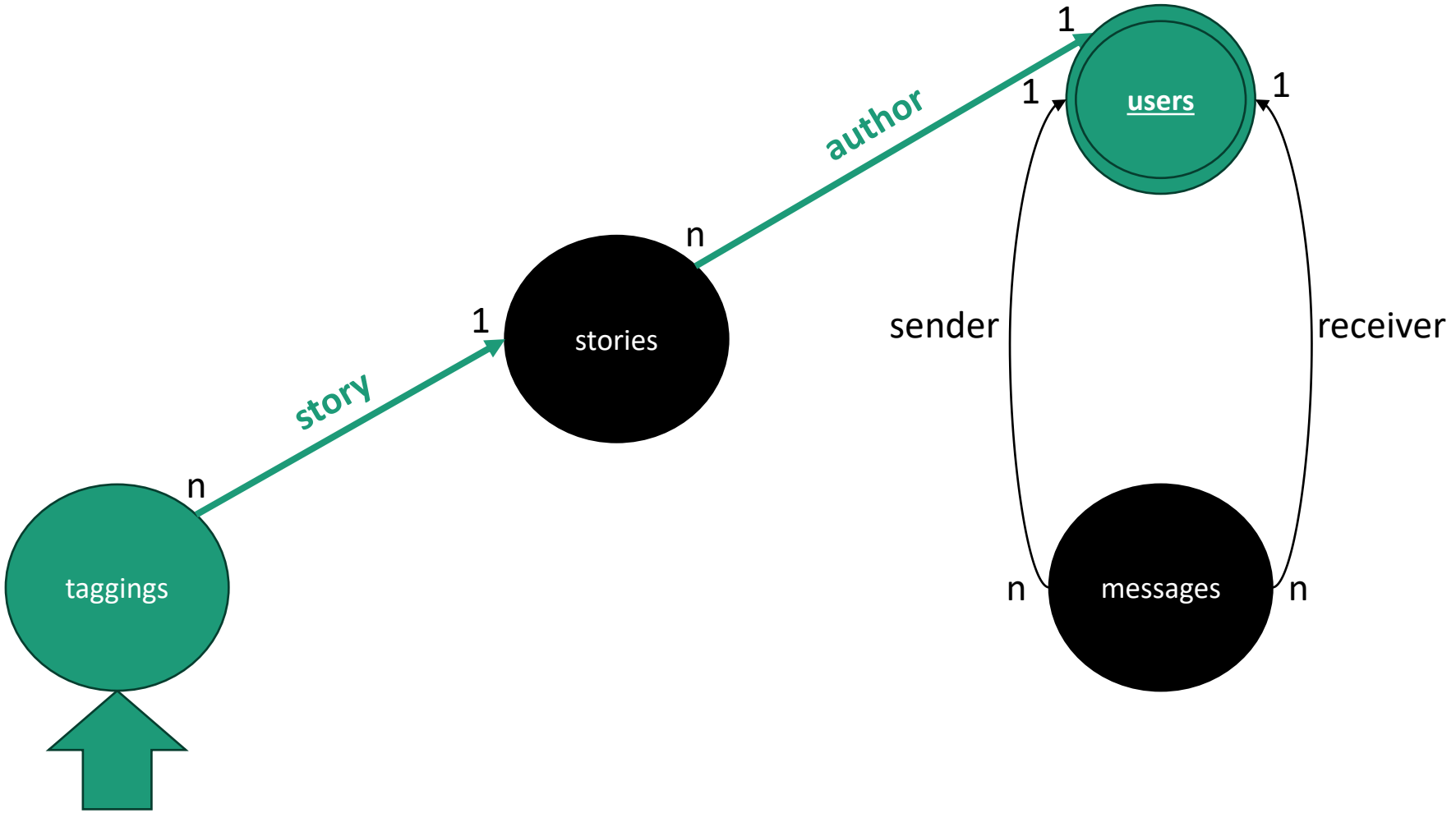


Data Ownership Graph (DOG )

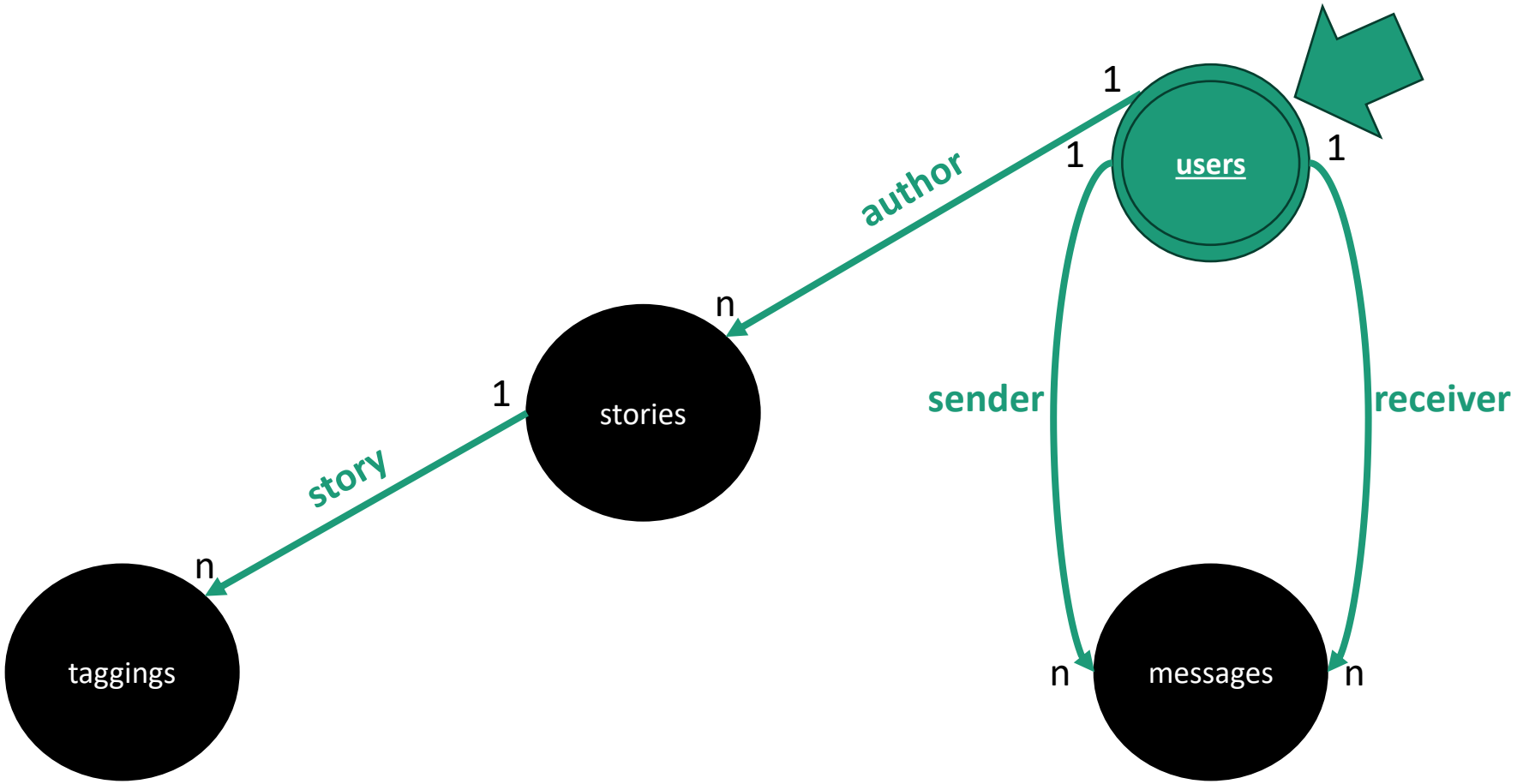
Specifies how to handle **access** and **deletion** requests



DOG identifies the owners of every row

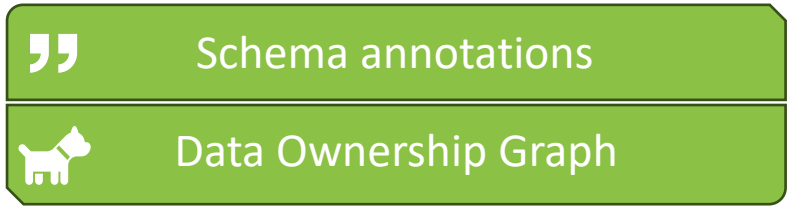


DOG identifies data owned by data subject



Challenges

Capture application-specific compliance policy

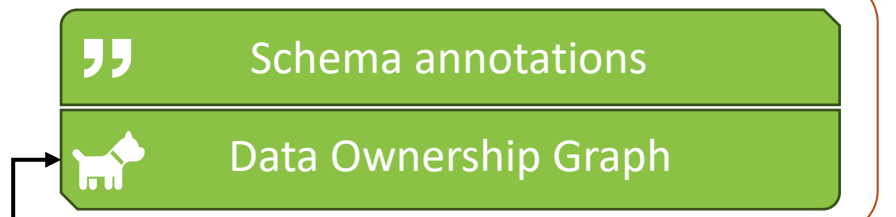


Correctly handle access/deletion requests and enforce compliance invariants

Maintain good performance

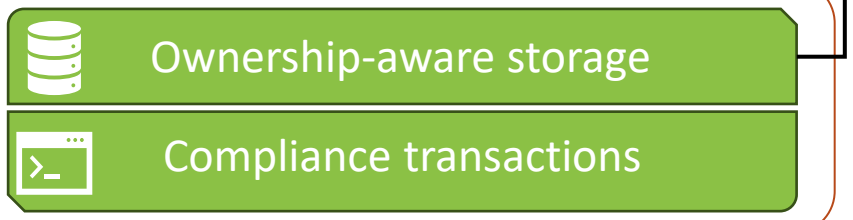
Challenges

Capture application-specific compliance policy



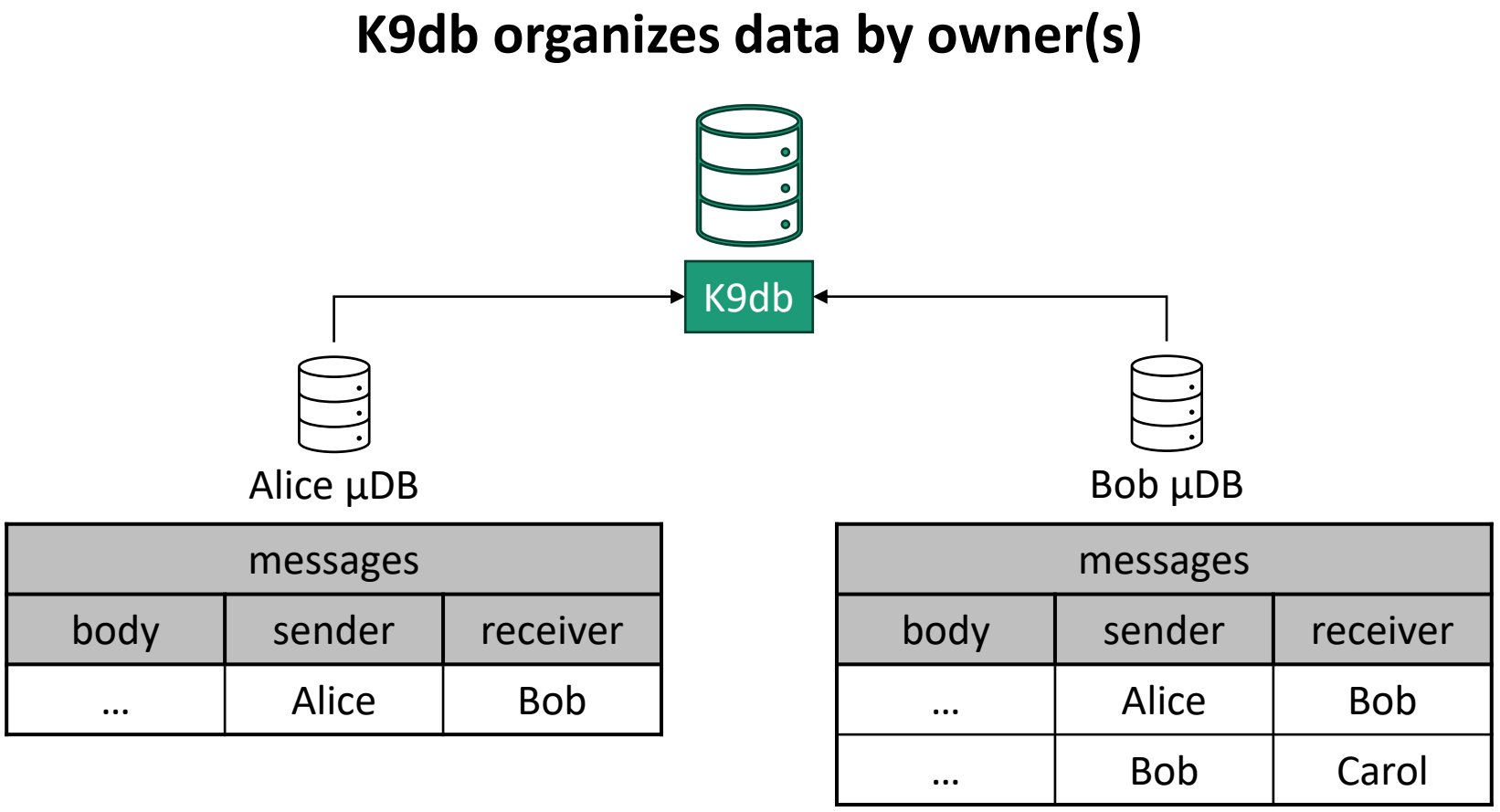
Organized according to

Correctly handle access/deletion requests and enforce compliance invariants

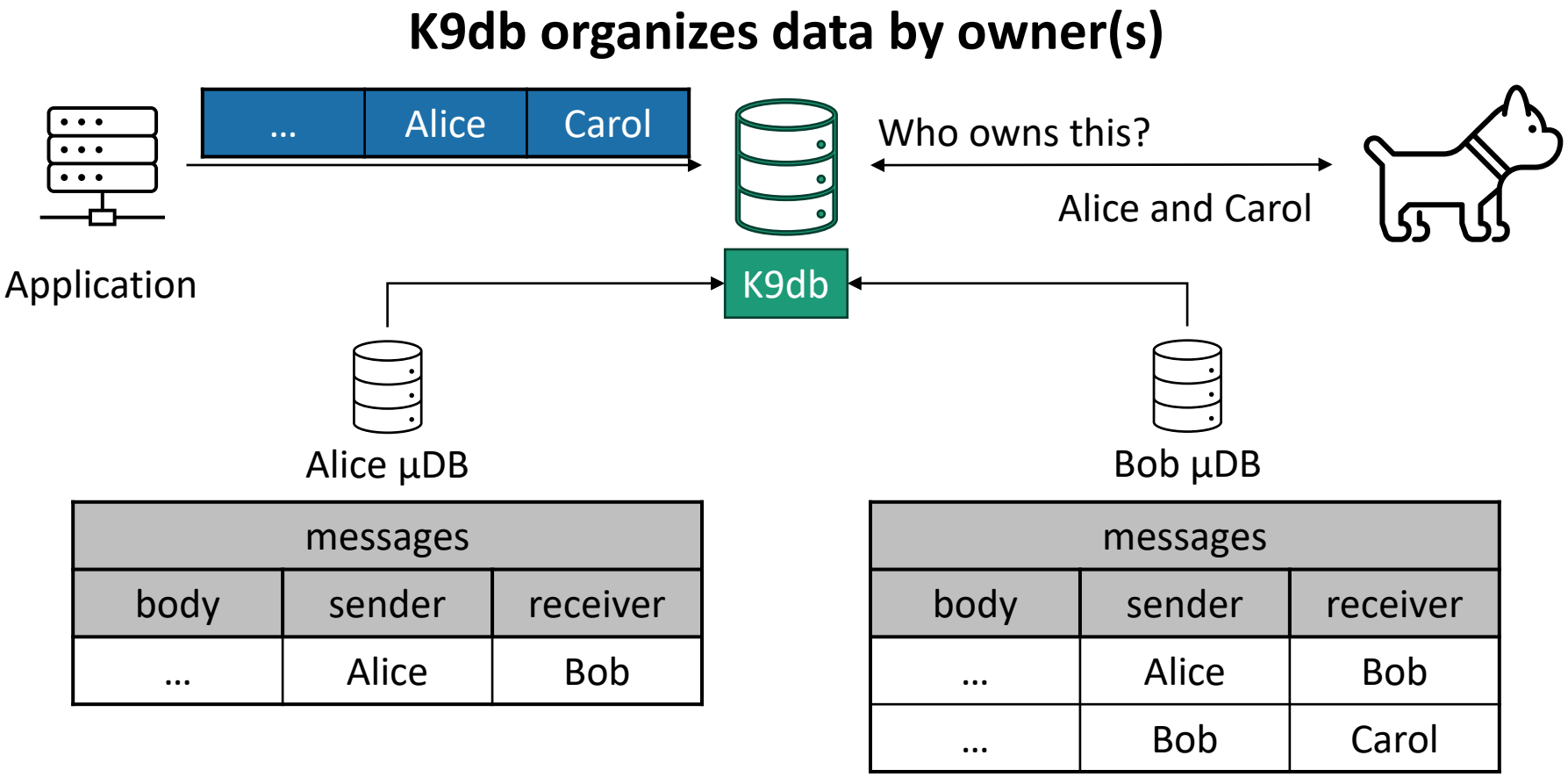


Maintain good performance

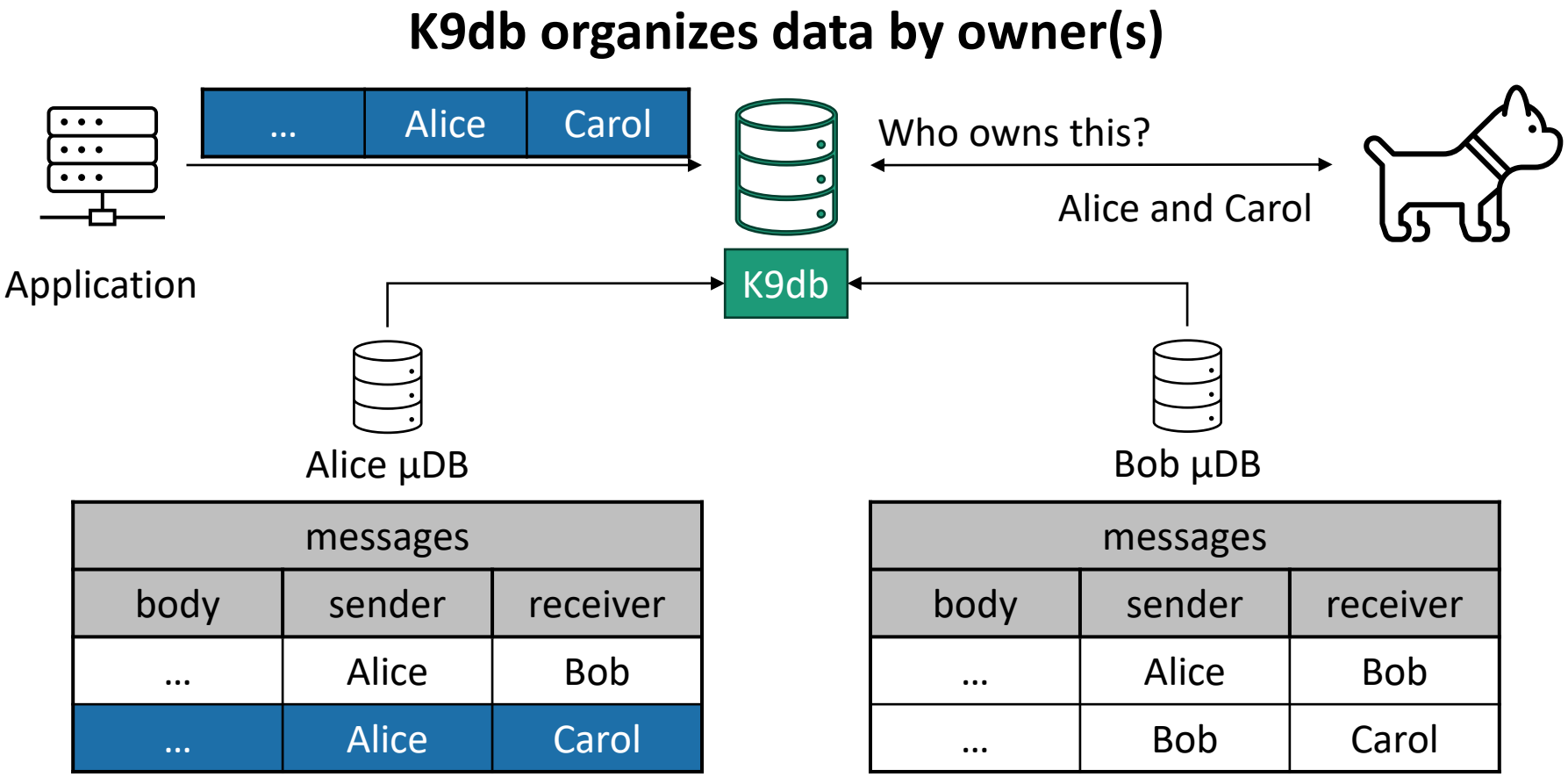
Ownership-aware storage: per-user µDBs



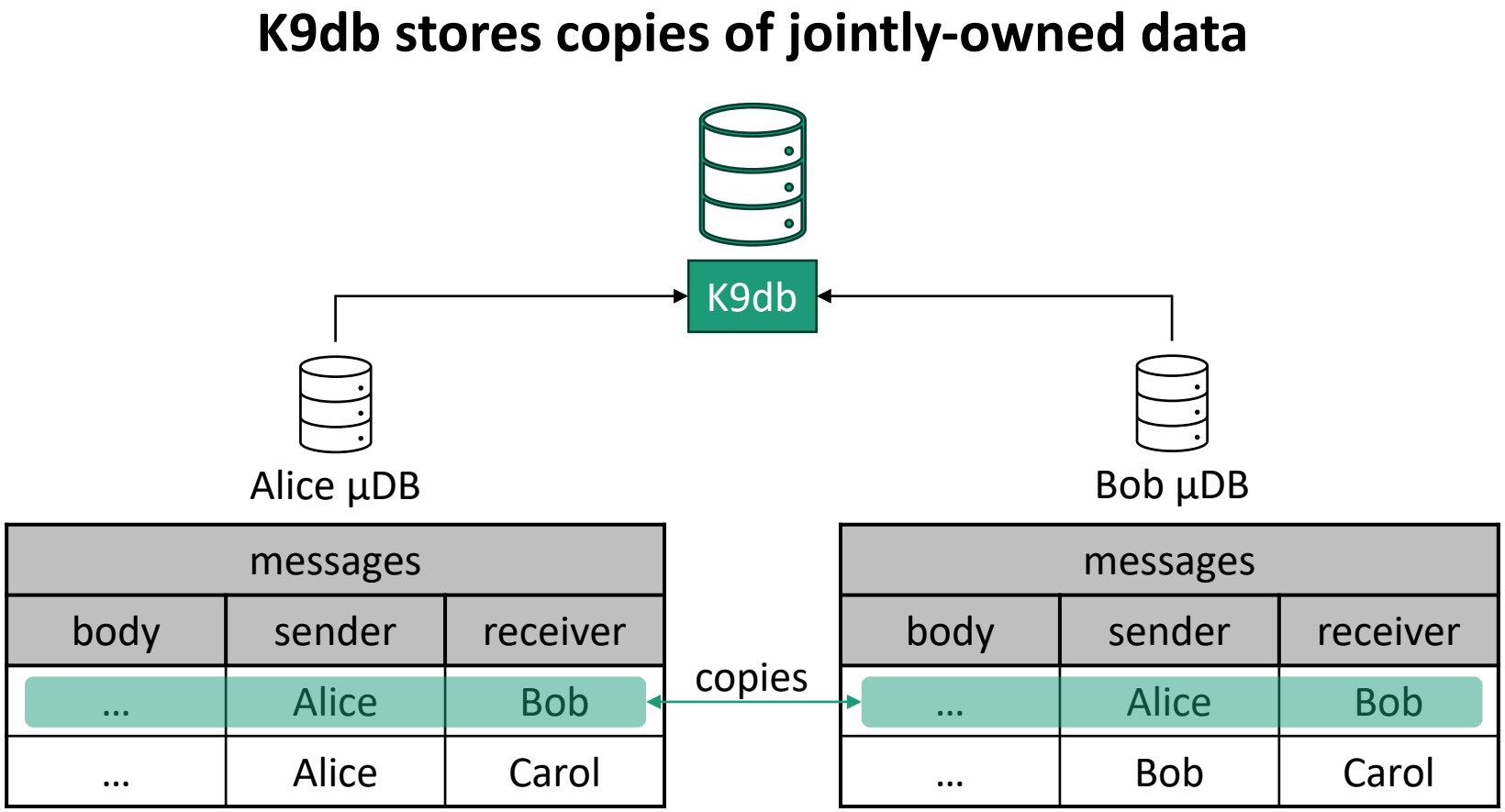
Ownership-aware storage: per-user µDBs



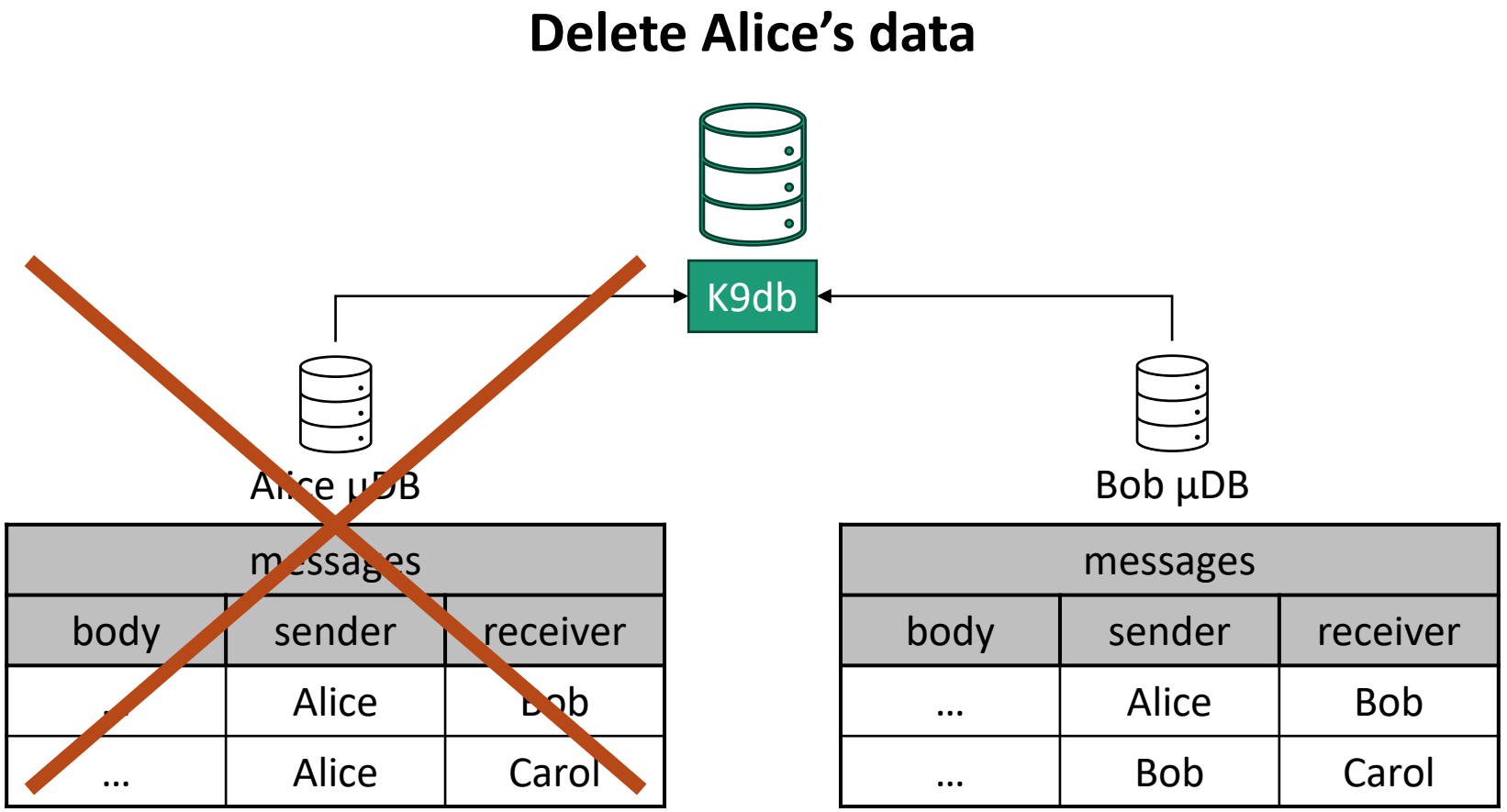
Ownership-aware storage: per-user µDBs



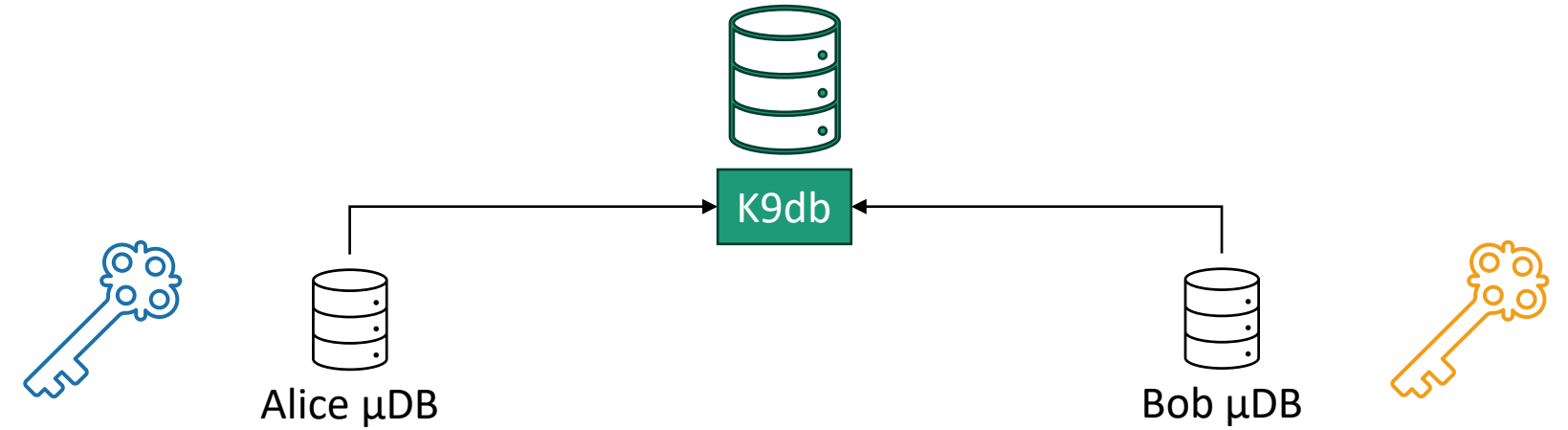
Ownership-aware storage: per-user µDBs



Advantage 1: µDBs provide correct deletion by construction



Advantage 2: µDBs allow encryption with user-specific keys

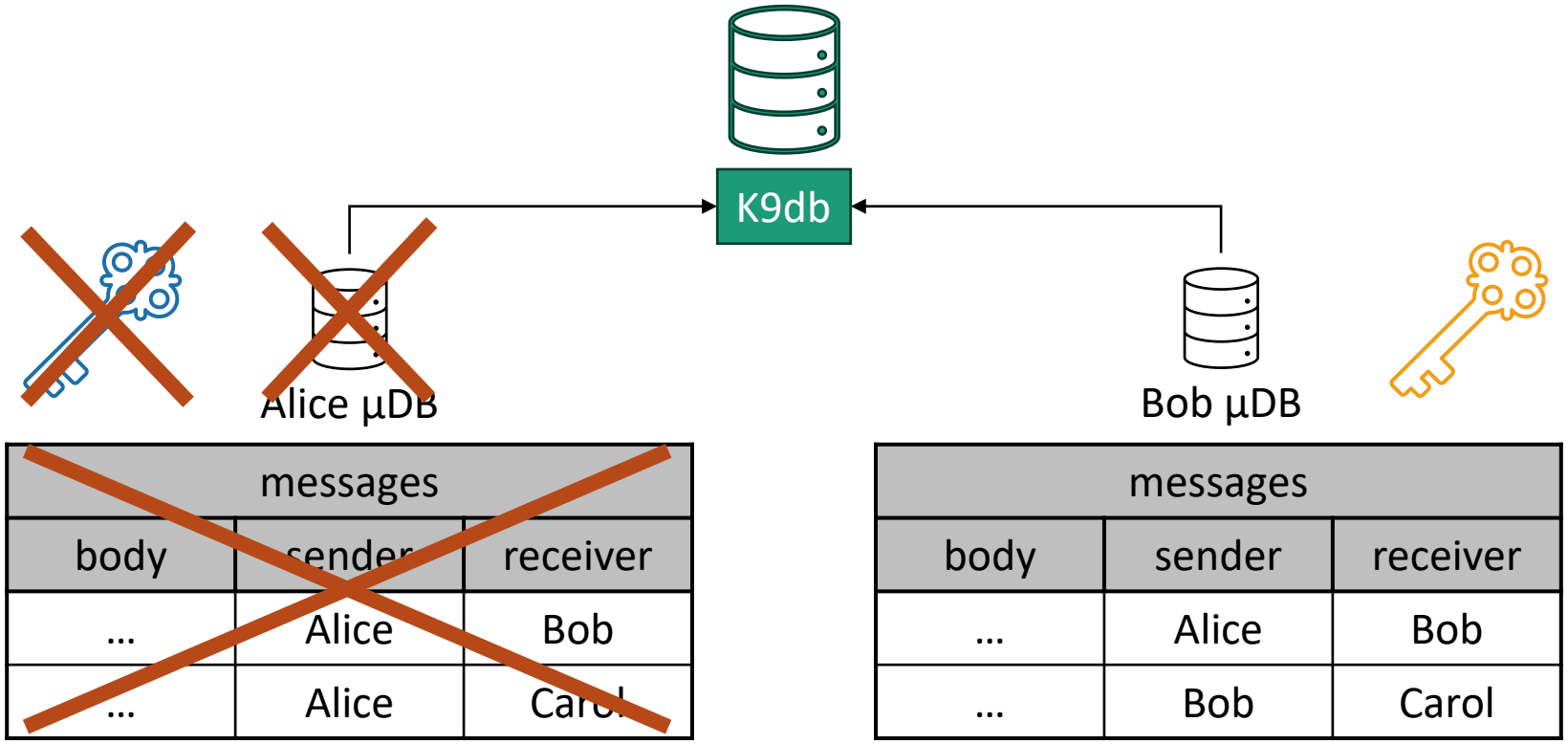


messages		
body	sender	receiver
…	Alice	Bob
…	Alice	Carol

messages		
body	sender	receiver
…	Alice	Bob
…	Bob	Carol

Advantage 2: µDBs allow encryption with user-specific keys

Delete Alice's data ➔ Delete Alice's key ➔ Backups of Alice's data are "deleted"



Advantage 3: µDBs ensure applications maintain compliance throughout execution

- Some **regular application operations** may violate compliance
  - Cause data to be orphaned (*i.e.*, has no owners)

- K9db detects and rejects violating operations with the help of µDBs
  - *E.g.*, Orphaned data cannot be stored in any user µDB

Advantage 3: µDBs ensure applications maintain compliance throughout execution

- Some **regular application operations** may violate compliance
  - Cause data to be orphaned (*i.e.*, has no owners)

- K9db detects and rejects violating operations with the help of µDBs
  - *E.g.*, Orphaned data cannot be stored in any user µDB

- Common pattern: temporarily create orphaned data then delete it
  - K9db safely supports such sequences using **compliance transactions (CTX)**
  - K9db commits CTX only if compliance is restored

Challenges

Capture application-specific compliance policy

Schema annotations

Data Ownership Graph

Correctly handle access/deletion requests and enforce compliance invariants

Ownership-aware storage

Compliance transactions

Maintain good performance

K9db Implementation

- K9db realizes µDBs over a single datastore (RocksDB)
  - K9db secondary indexes identify all the µDBs where a row is stored
  - RocksDB iterators, snapshots, transactions …

K9db Implementation

- K9db realizes µDBs over a single datastore (RocksDB)
  - K9db secondary indexes identify all the µDBs where a row is stored
  - RocksDB iterators, snapshots, transactions …

- K9db serves complex queries from an **integrated in-memory cache**
  - based on materialized views maintained via dataflow processing
  - K9db updates cache in response to deletion requests
    → cache is always compliant

Evaluation Questions

Evaluation Questions

1. What is the impact of using K9db on E2E web application performance?

Evaluation Questions

1. What is the impact of using K9db on E2E web application performance?

2. What effect do K9db's design features and optimizations have on performance?

Evaluation Questions

1. What is the impact of using K9db on E2E web application performance?

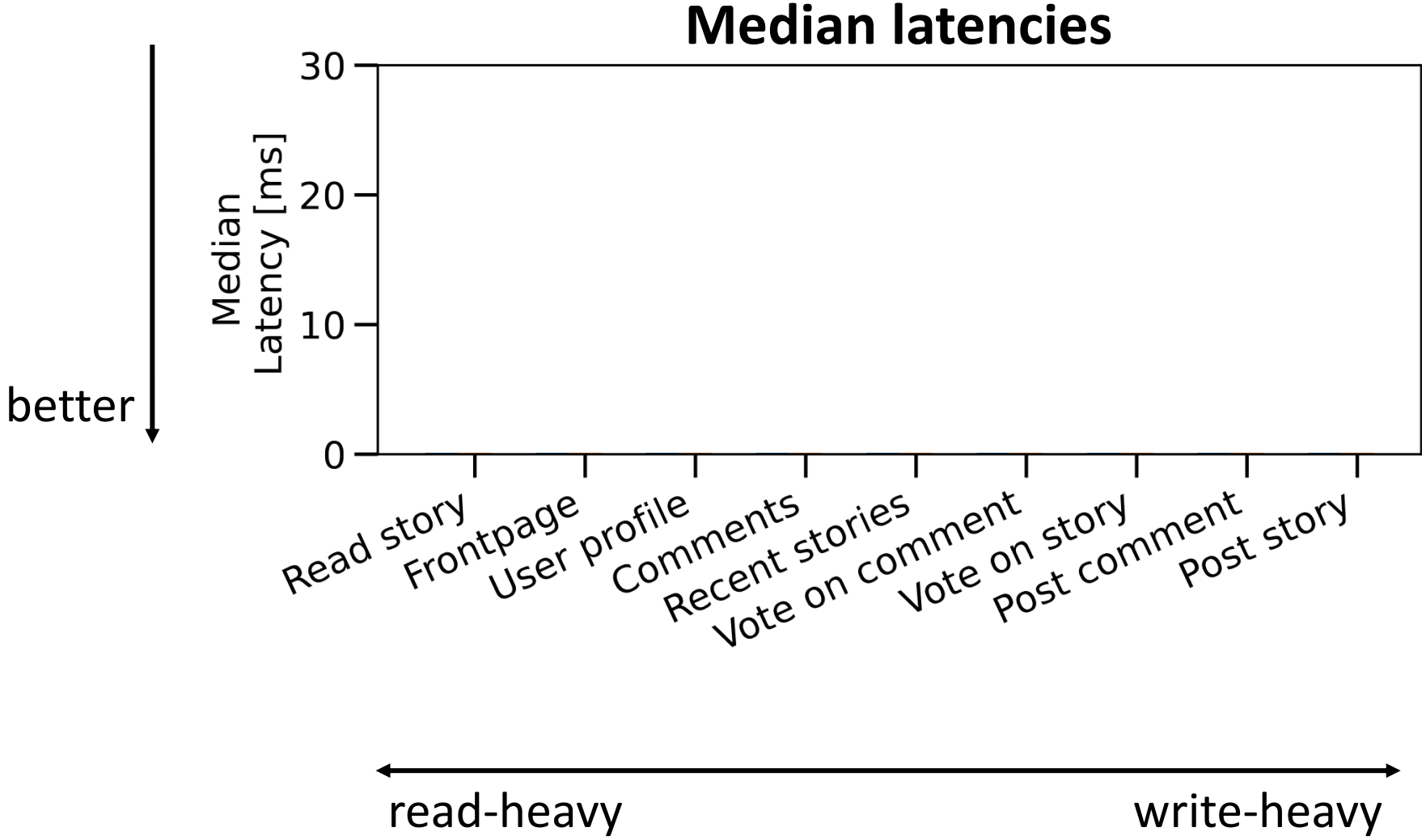
2. What effect do K9db's design features and optimizations have on performance?

3. How much application developer effort is required to use K9db?

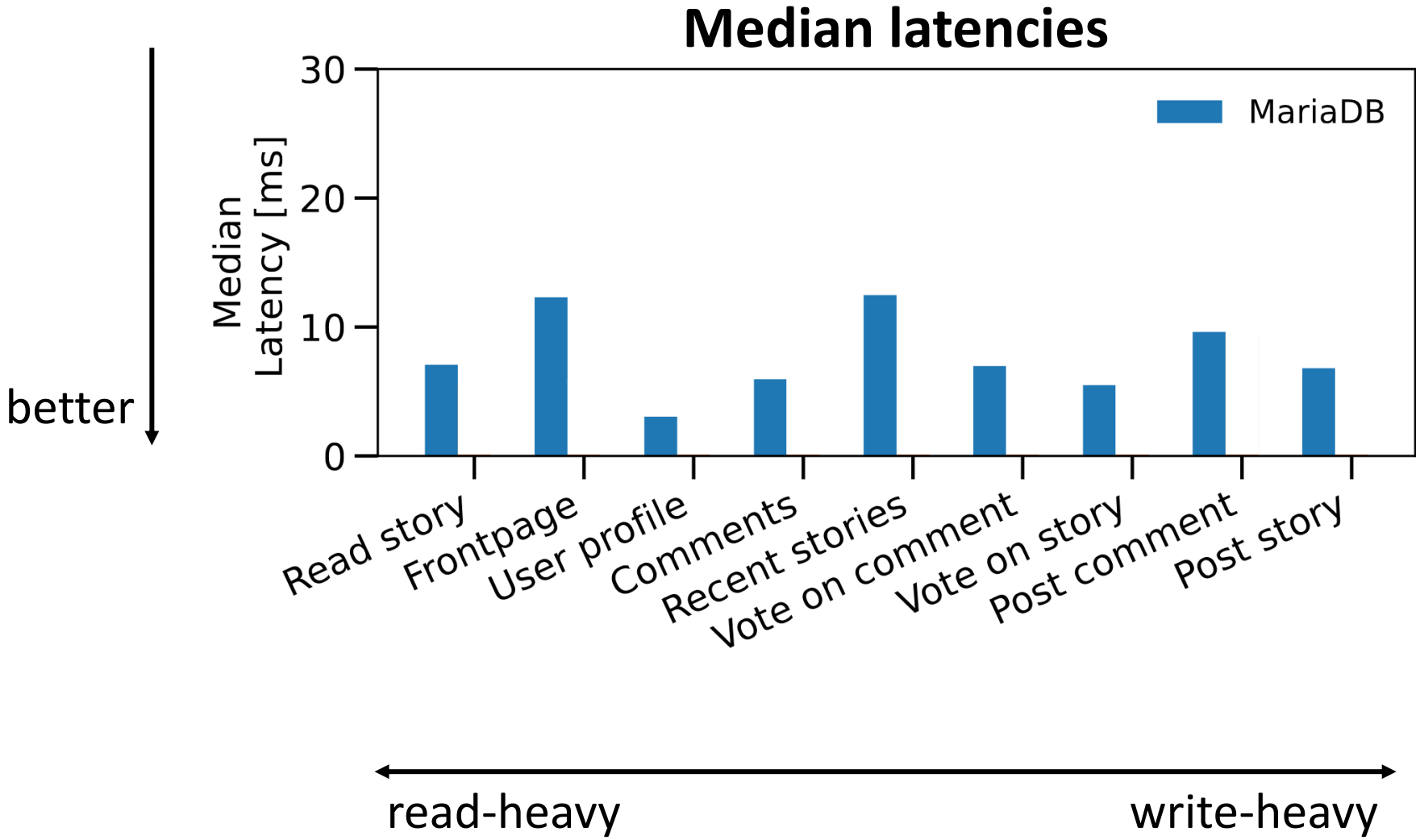
Evaluation Questions

1. What is the impact of using K9db on E2E web application performance?
   - Two real world applications: **Lobsters**, ownCloud
2. What effect do K9db's design features and optimizations have on performance?
3. How much application developer effort is required to use K9db?

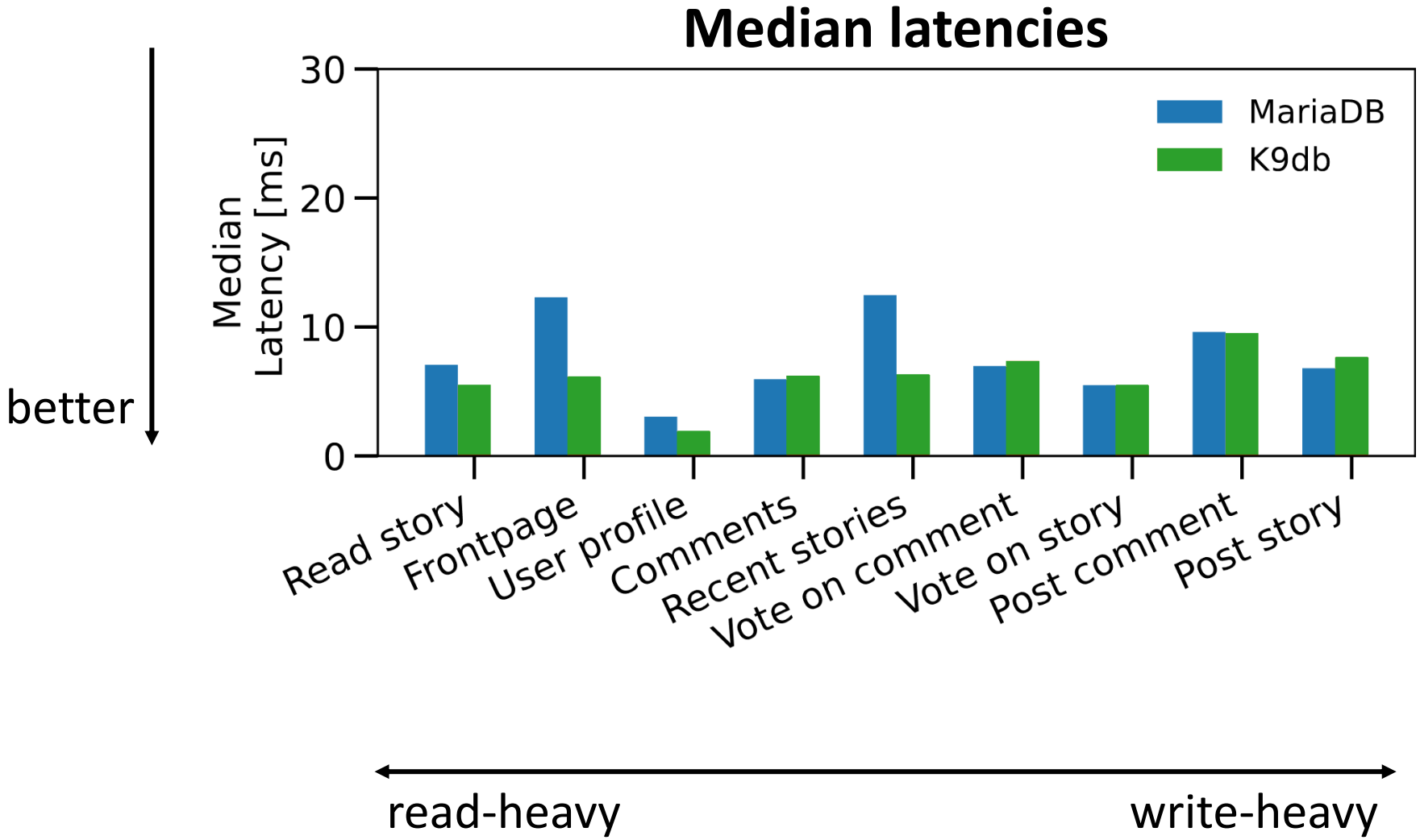
What is the impact of using K9db on E2E web application performance?



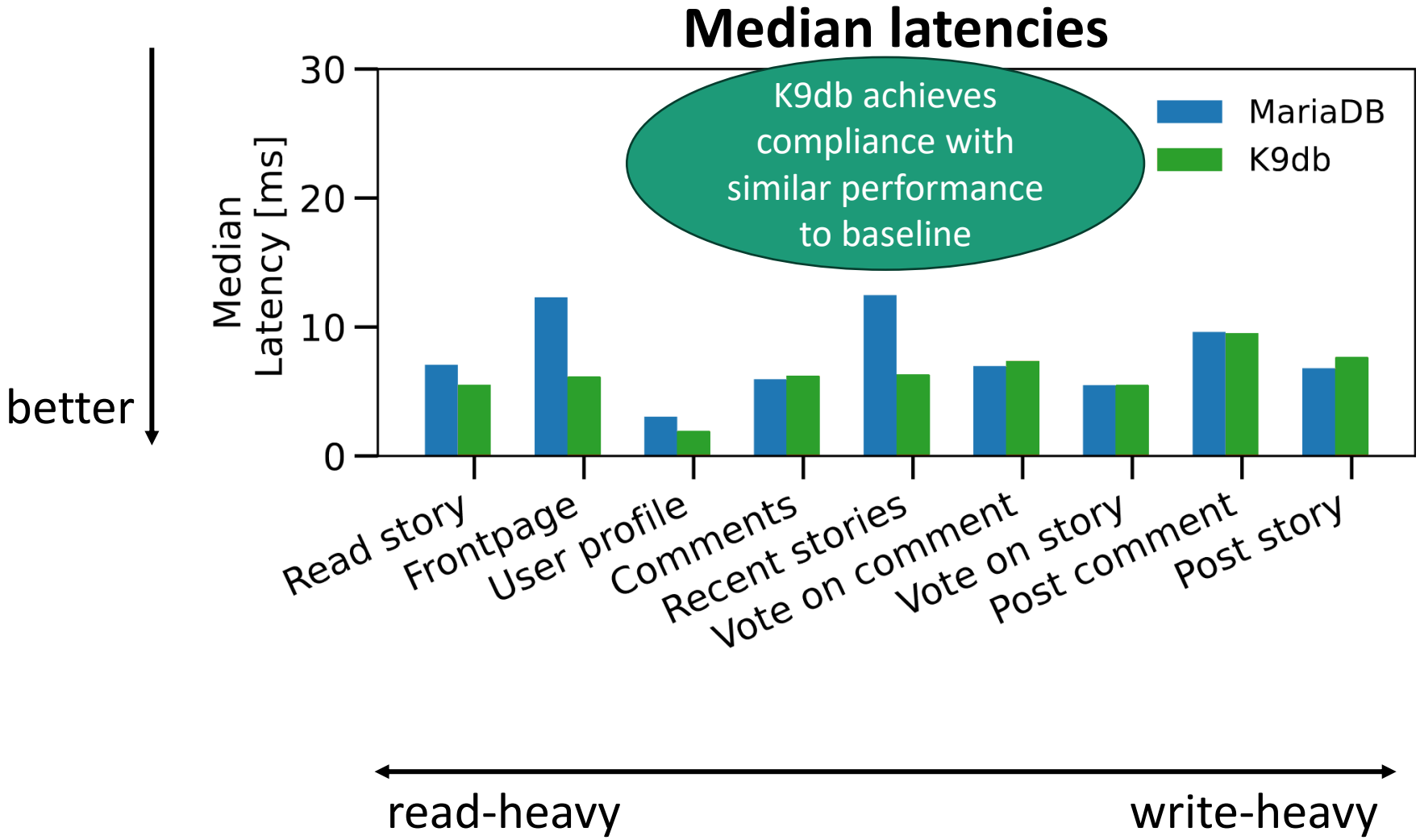
What is the impact of using K9db on E2E web application performance?



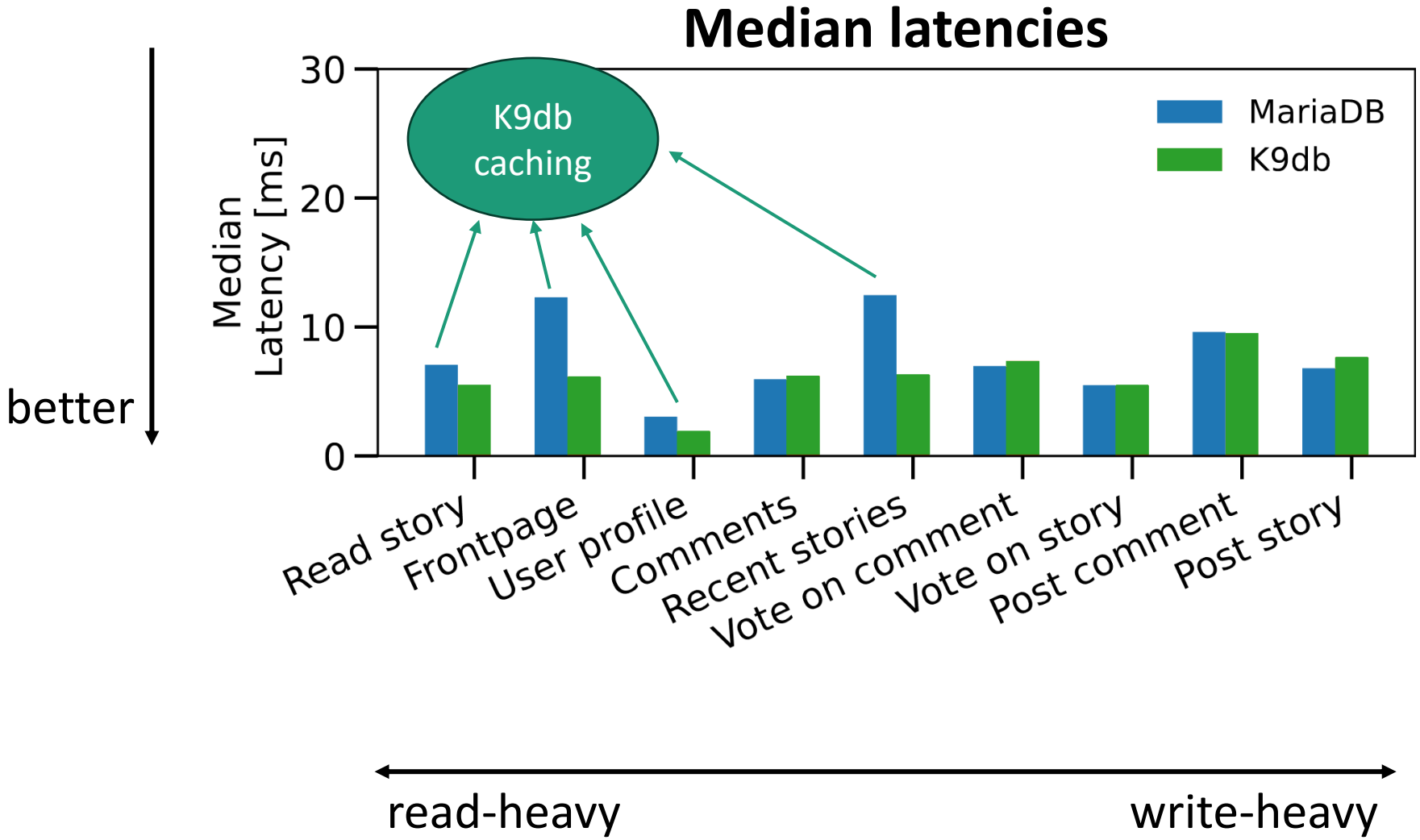
What is the impact of using K9db on E2E web application performance?



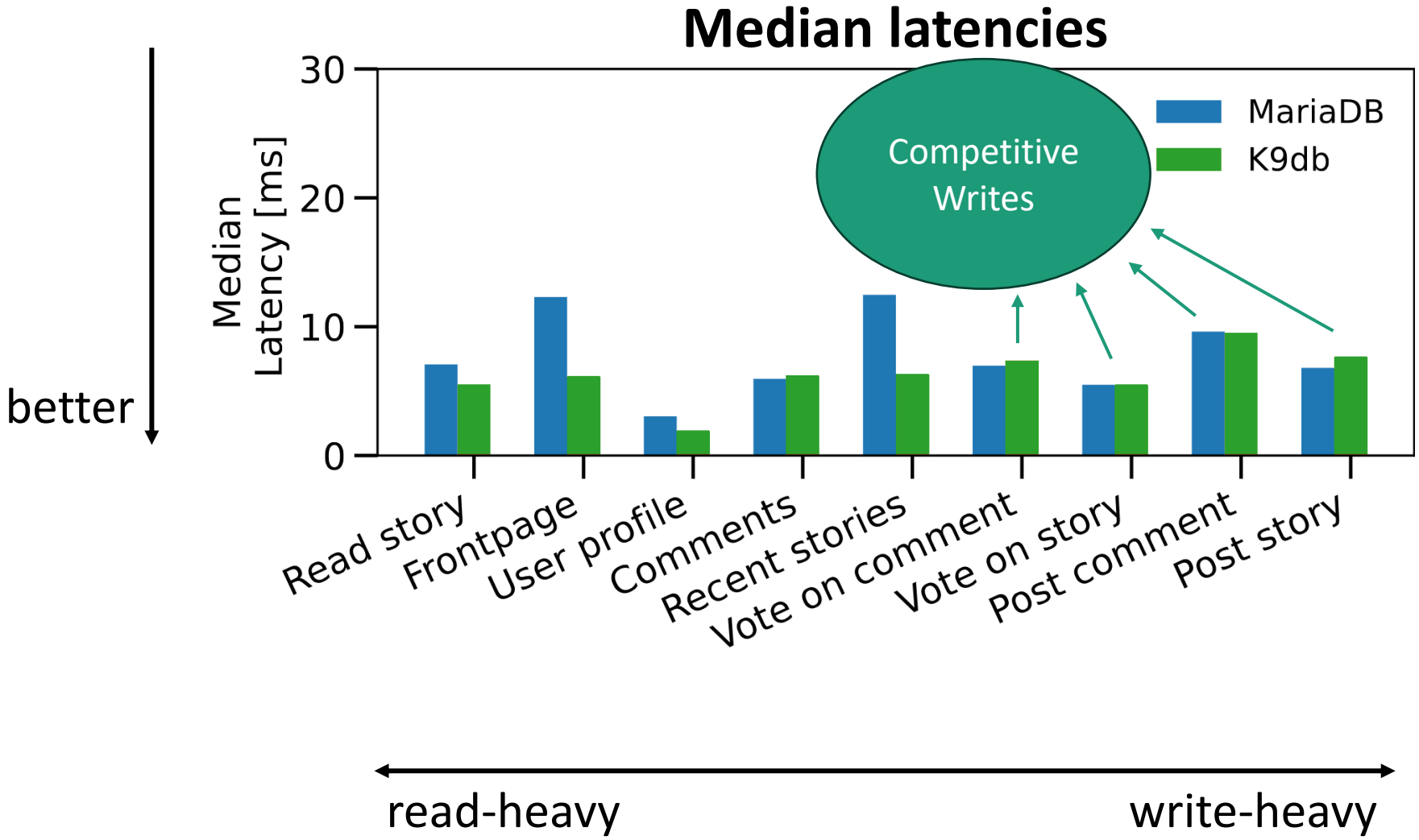
What is the impact of using K9db on E2E web application performance?



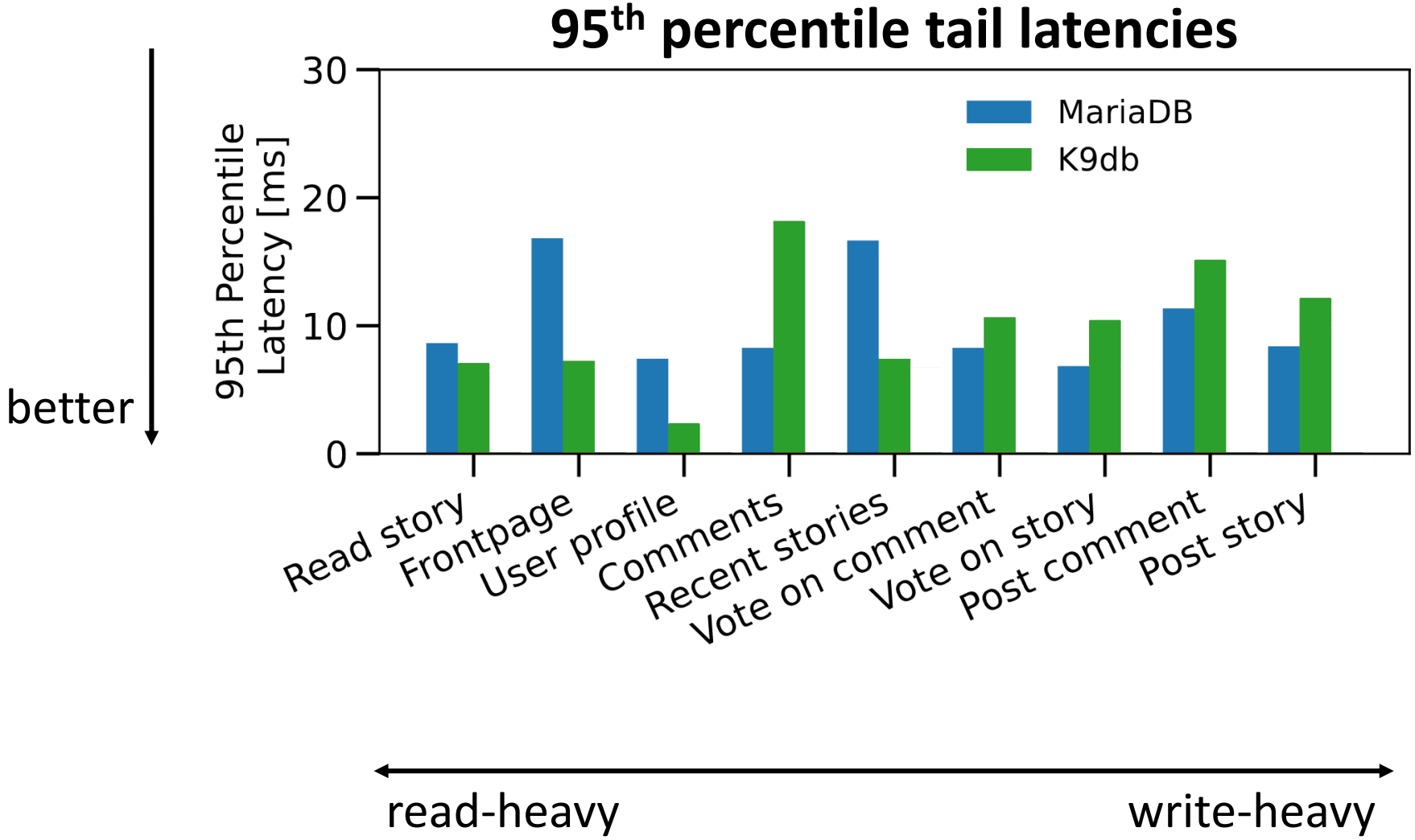
What is the impact of using K9db on E2E web application performance?



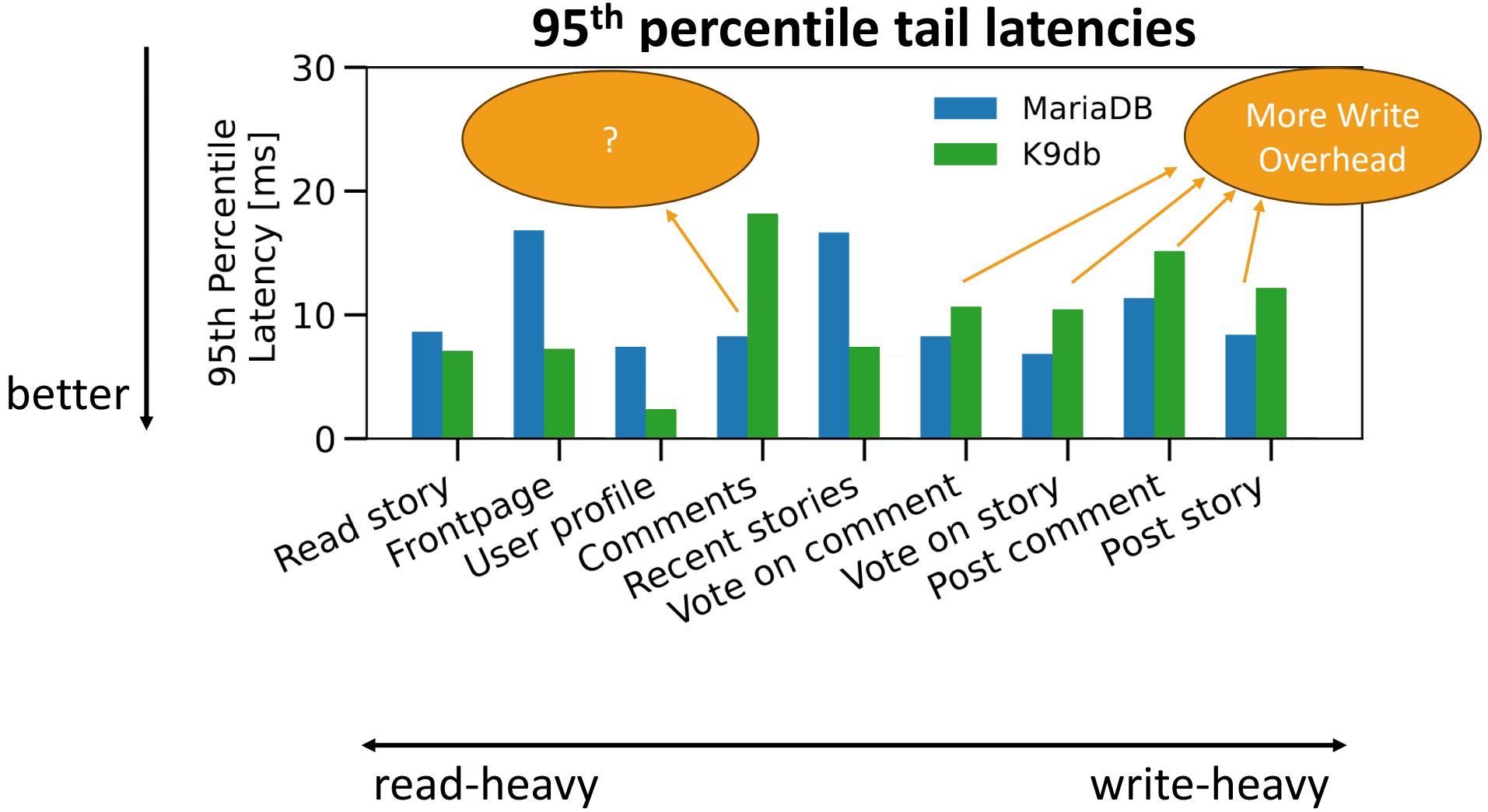
What is the impact of using K9db on E2E web application performance?



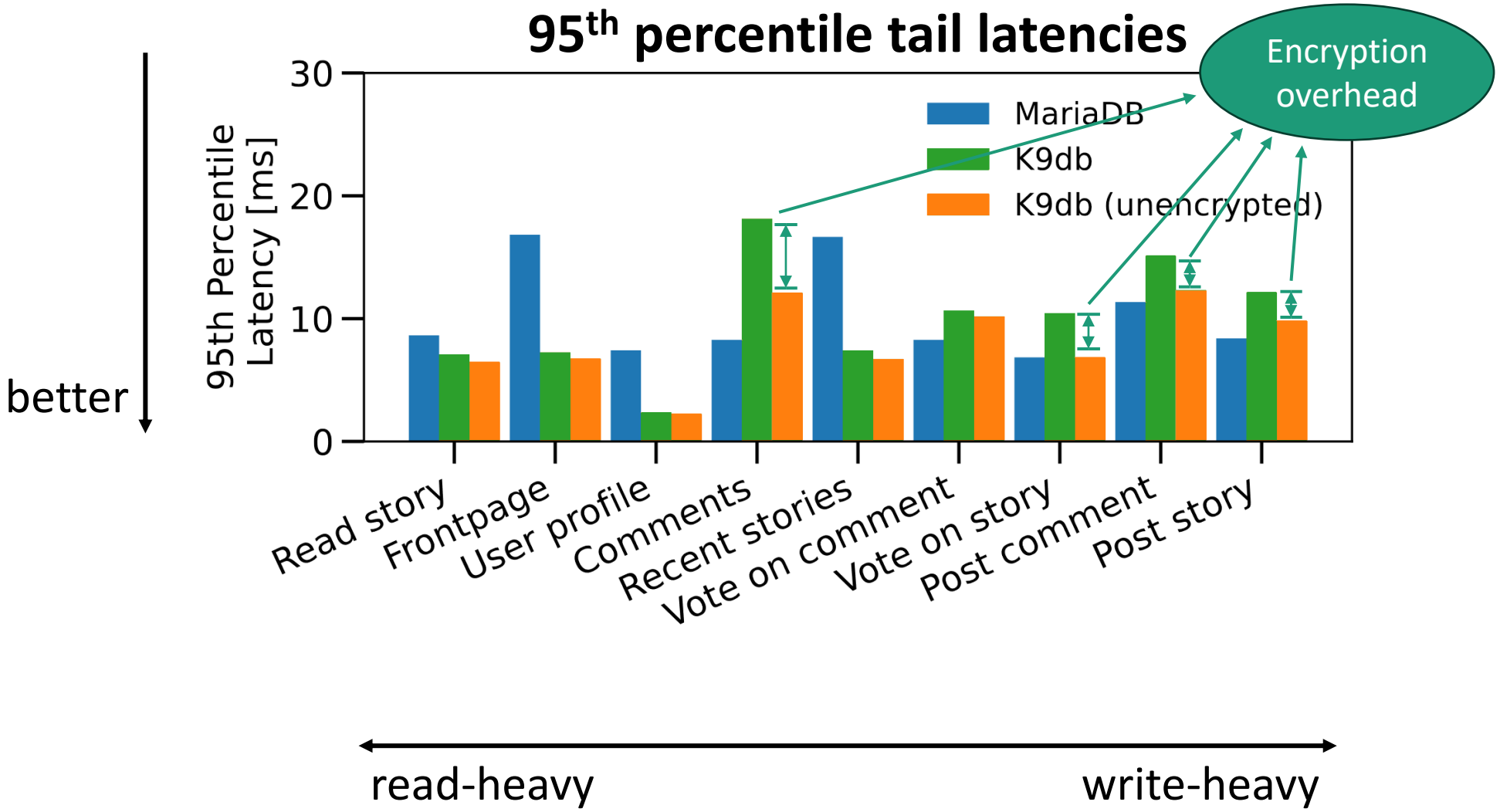
What is the impact of using K9db on E2E web application performance?



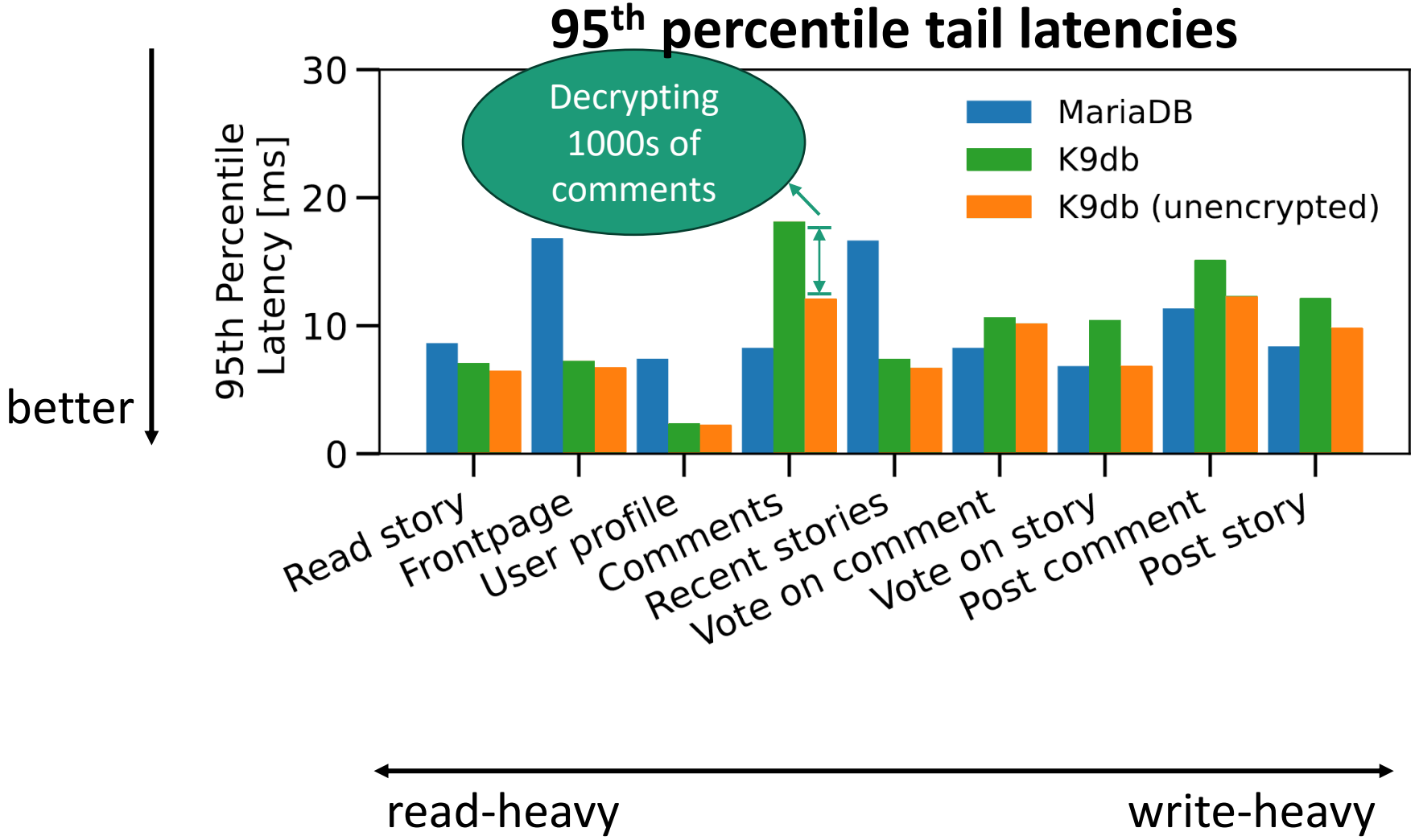
What is the impact of using K9db on E2E web application performance?



What is the impact of using K9db on E2E web application performance?



What is the impact of using K9db on E2E web application performance?



Evaluation – more in the paper

- Comparable performance to on-demand caching (Memcached), reasonable memory overhead
- Storage layout optimizations critical for good performance
- K9db's schema annotations can express policies for 10 real web applications

Conclusion: K9db is a database that helps developers get compliance with GDPR right!

- **Key abstraction: data ownership graph (DOG)** captures compliance policies of real applications
- Ownership aware storage layer with per-user µDBs
- K9db achieves performance comparable to SQL databases

https://github.com/brownsys/K9db

babman@brown.edu

ETOS

BROWN