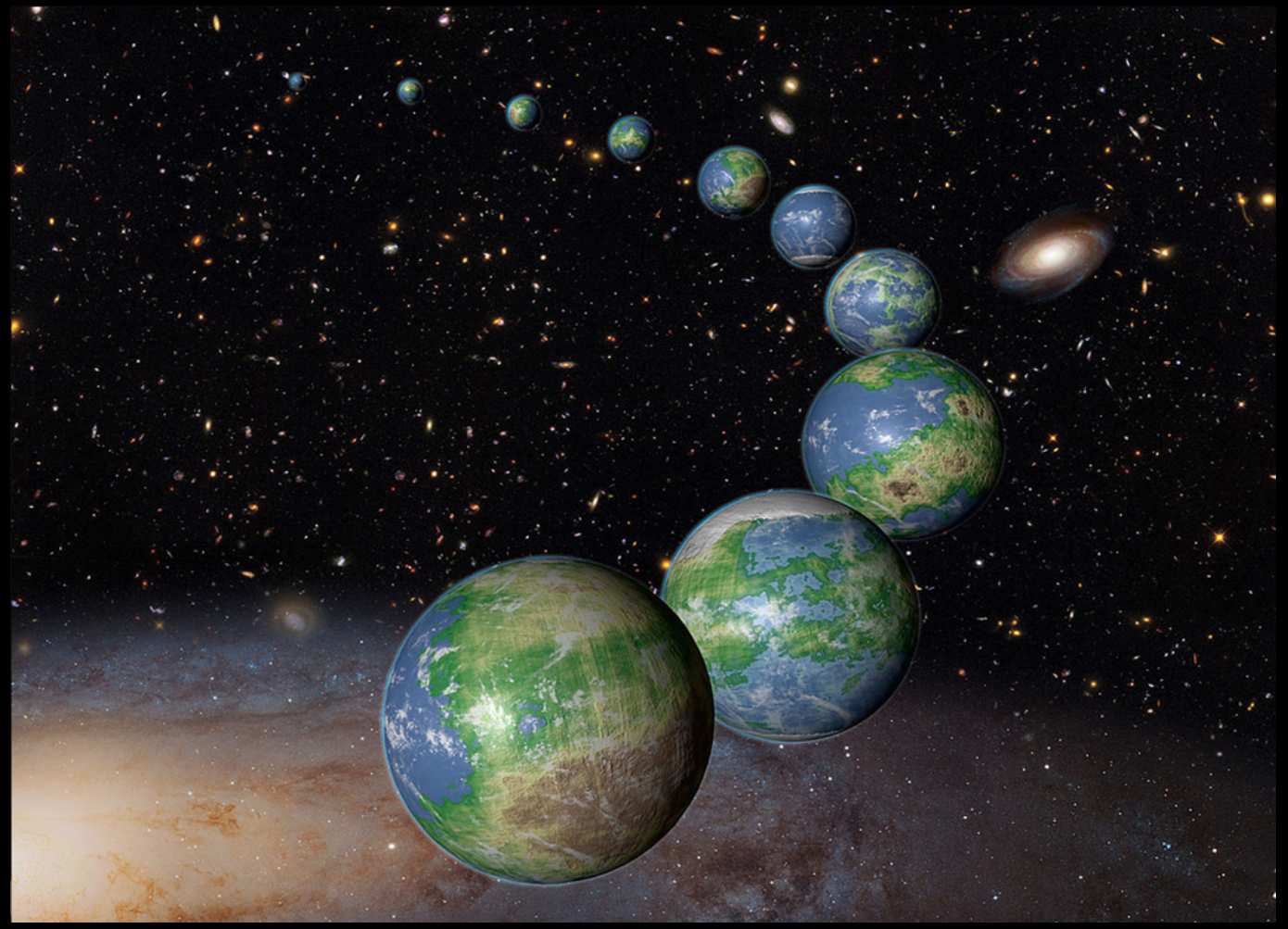
# Towards Multiverse Databases

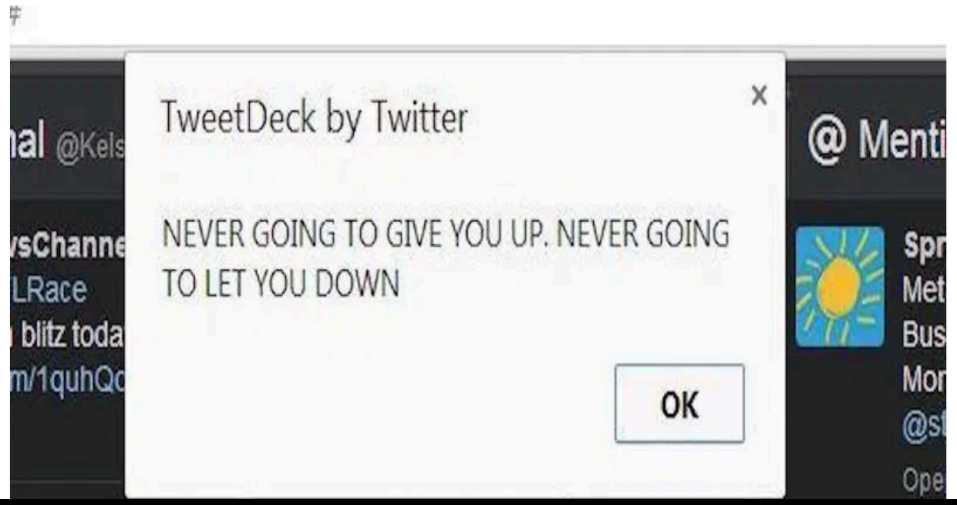


# Plan:

- 1 Context
- 2 Questions-driven interactive presentation
  - The multiverse approach
  - How to make multiverse databases practical
- 3 Proof of Concept
- 4 Research Directions

# TweetDeck Taken Down To Assess XSS Vulnerability

Kyle Russell @kylebrussell/ / 1:06 pm EDT • June 11, 2014




# Facebook admits bug allowed apps to see hidden photos

**Bug let developers access pictures people had uploaded but chosen not to post**

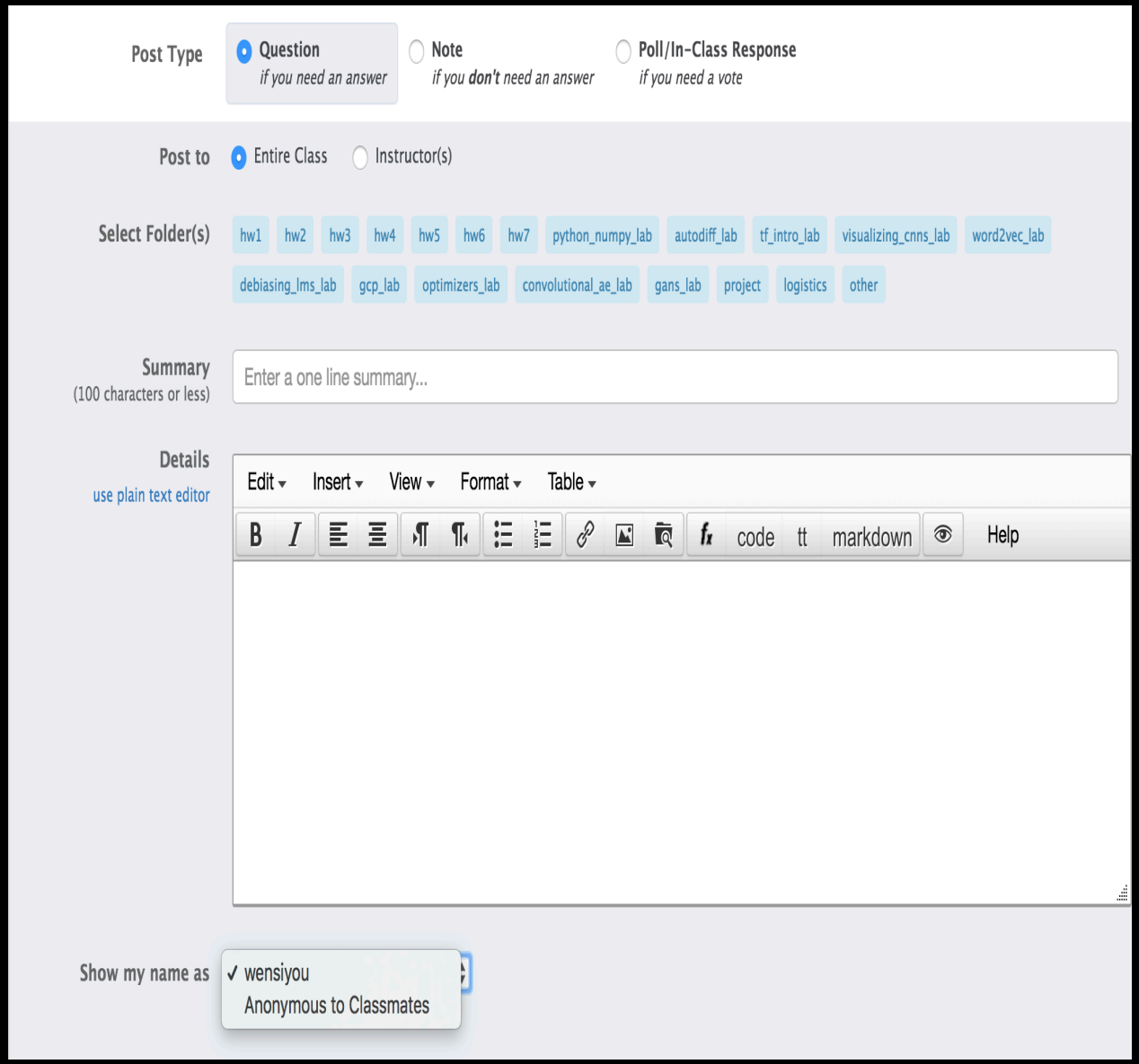

▲ Facebook says up to 6.8 million users and up to 1,500 apps have been affected. Photograph: Noah Berger/AP

# British Airways faces record-breaking GDPR fine after data breach

*The ICO wants to fine the airline £183 million*

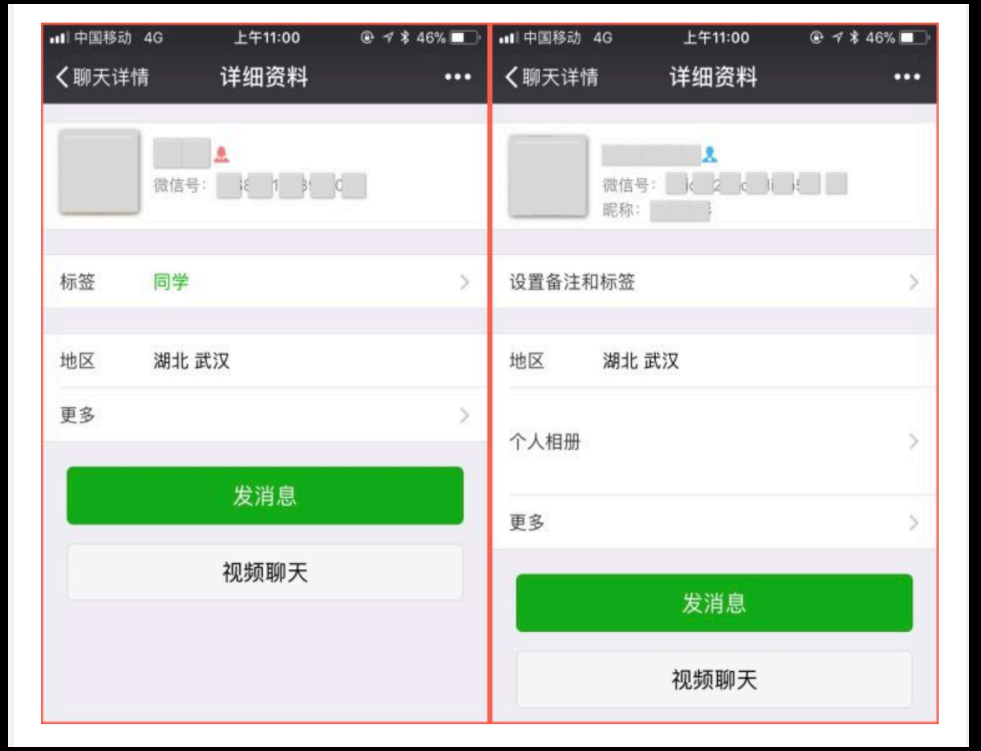
By Jon Porter | @JonPorty | Jul 8, 2019, 4:15am EDT

SHARE




```
SELECT * FROM Post
WHERE user = ? AND
Post.anonymous = 0 OR
Post.author = 'alice';
```


Matthew Green
@matthew_d_green
Follow
Piazza offers anonymous posting, but does not hide each user's total number of posts. Discuss.
10:37 AM - 30 Oct 2017
11 Retweets 52 Likes

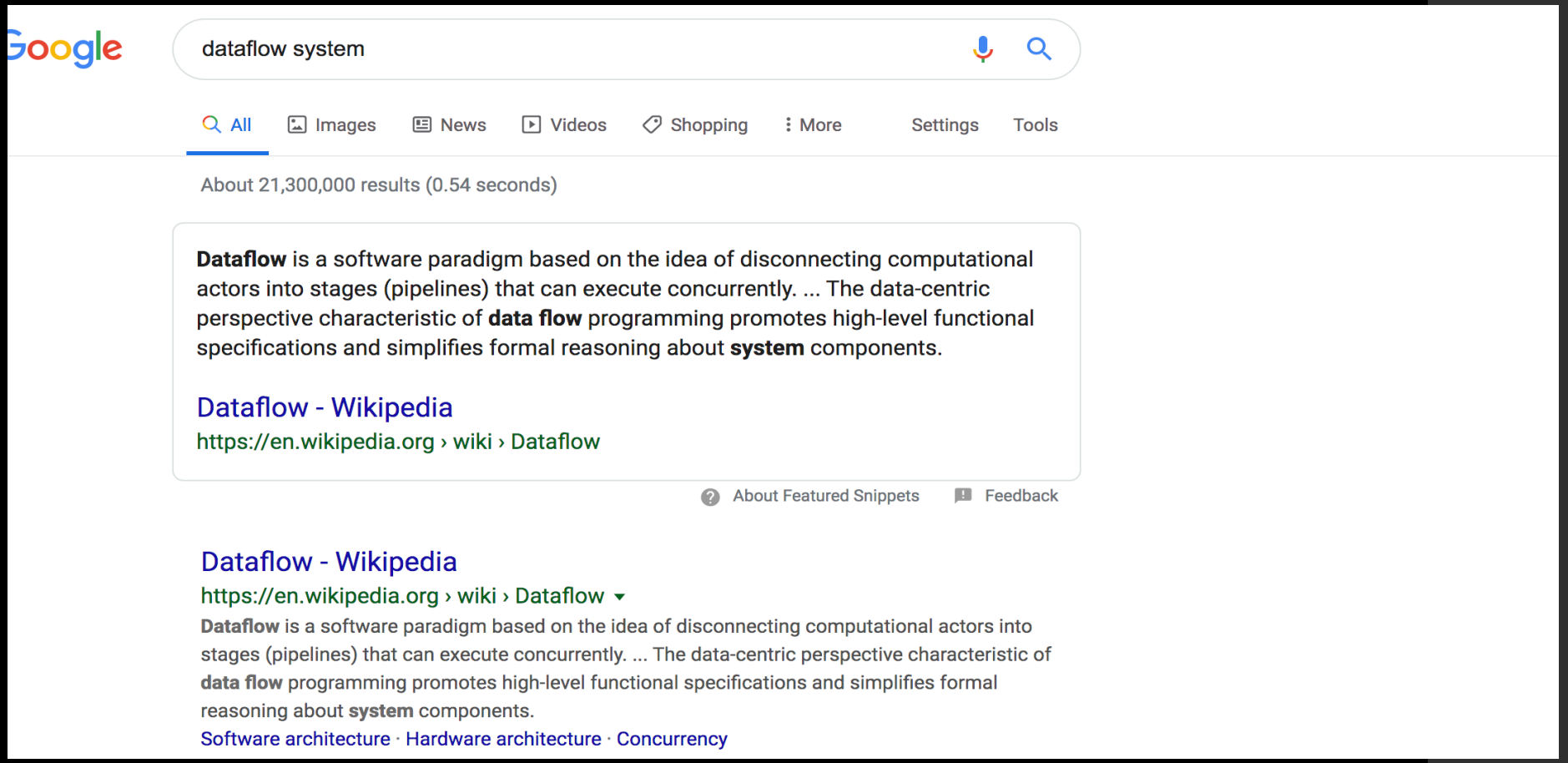

中国移动 4G 上午11:00 46%
聊天详情
详细资料
微信号:
标签 同学
地区 湖北 武汉
更多
发消息
视频聊天
中国移动 4G 上午11:00 46%
聊天详情
详细资料
微信号:
昵称:
设置备注和标签
地区 湖北 武汉
个人相册
更多
发消息
视频聊天

# “Why Cryptosystems Fail”

-- by Ross Anderson

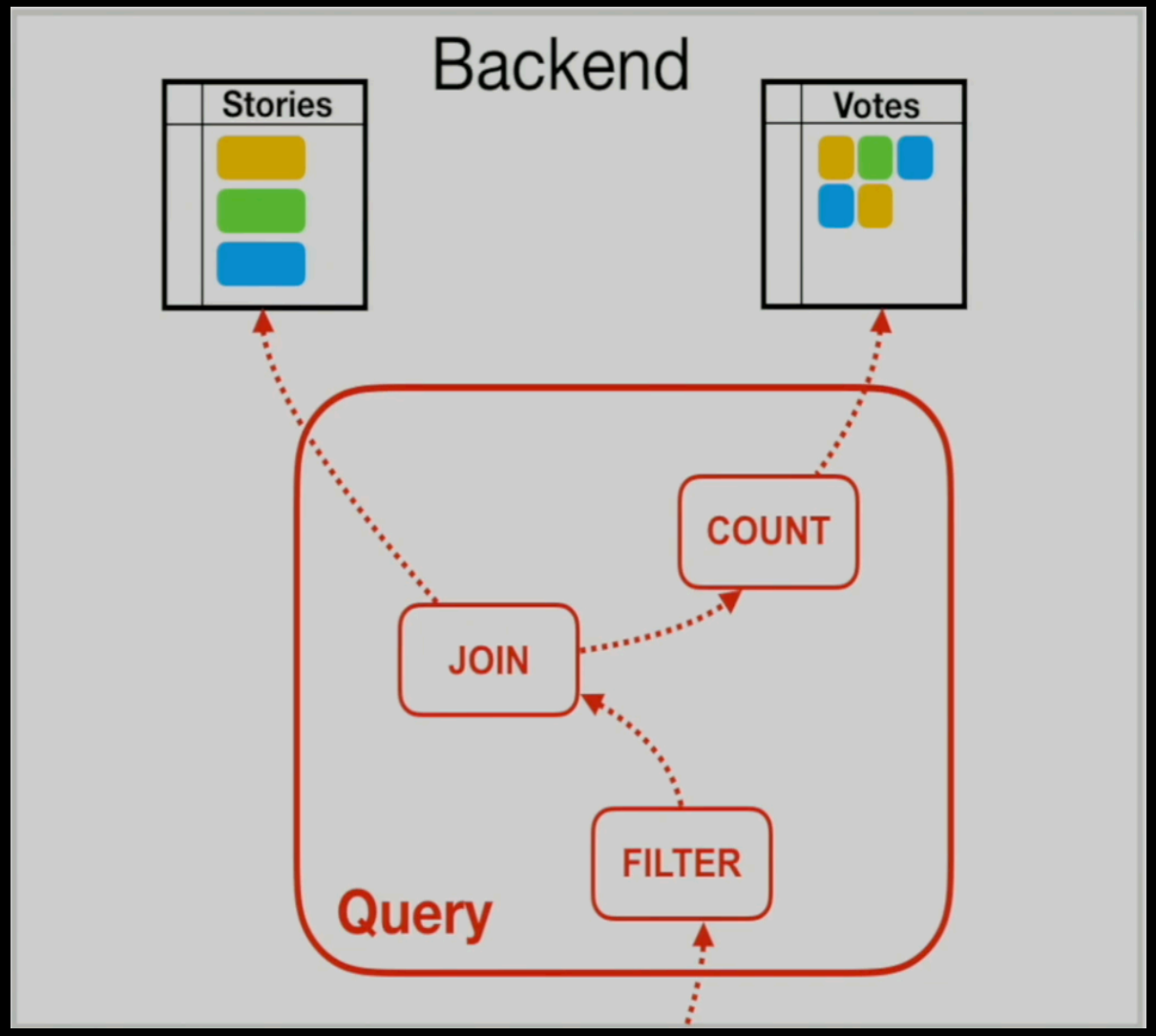
- *“It turns out that the threat model commonly used by cryptosystem designers was wrong: most frauds were not caused by cryptanalysis or other technical attacks, but by implementation errors and management failures.”*

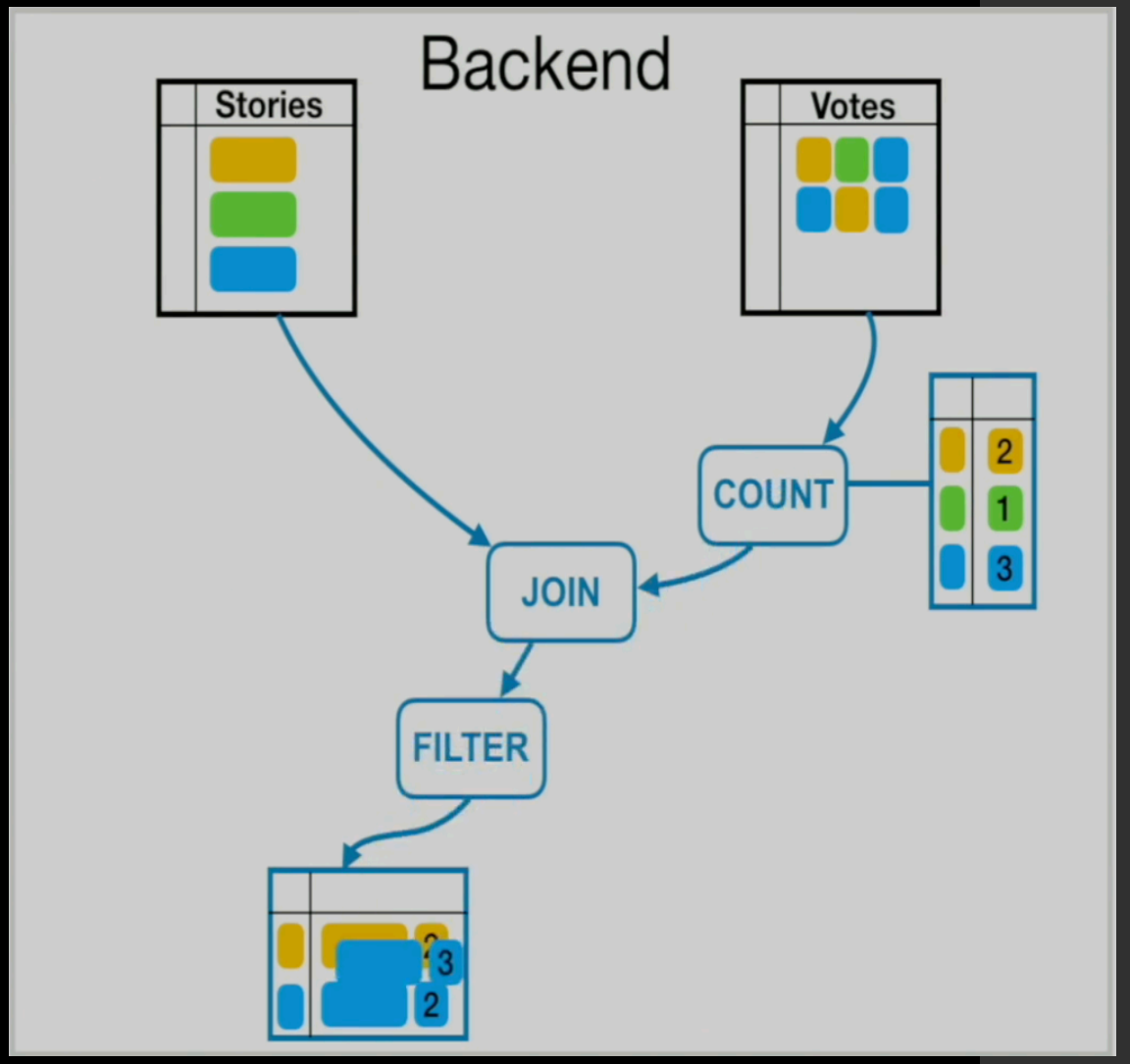
# Terms



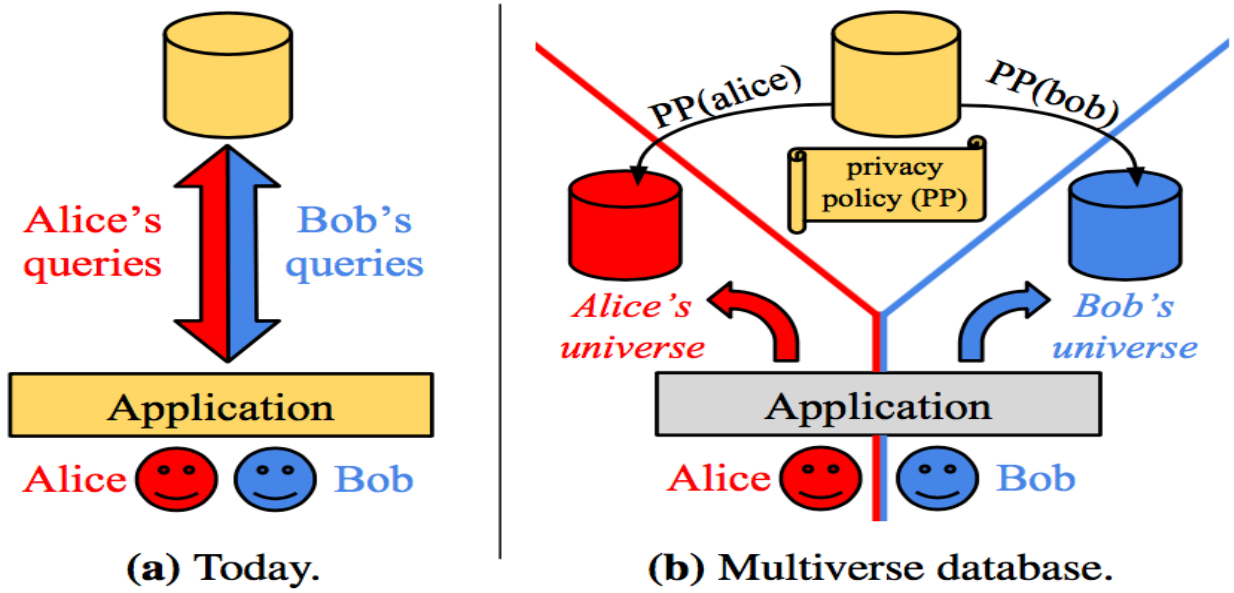

- Dataflow system (streaming computation system )
- Read-driven computation vs Write-driven computation
- computing in response to writes => dataflow computation

# Read-driven computation vs Write-driven computation



# Part 1 Basics of Design




**(a)** Today. **(b)** Multiverse database.

**Figure 1.** Currently, web applications' entire frontend code is trusted (amber) and can query any backend data *(left, (a))*. A multiverse database applies a privacy policy (PP) to generate transformed, logical per-user databases, in which the untrusted applications' queries can only see permissible data *(right, (b))*.

# Privacy Policy

```
table: Post,
-- user sees public posts and her own anonymous posts in full
-- (`ctx`, a universe-specific context, holds the user's ID)
allow: [ WHERE Post.anon = 0,
         WHERE Post.anon = 1 AND Post.author = ctx.UID ],
-- hide author of anonymous posts unless user is class staff
rewrite: [
  { predicate: WHERE Post.anon = 1 AND Post.class
      NOT IN (SELECT class FROM Enrollment
              WHERE role = "instructor" AND uid = ctx.UID),
    column: Post.author,
    replacement: "Anonymous" } ],
```

# Q1

- What is TCB (Trusted computing base) for multiverse databases?

- A. database
- B. database + privacy policy
- C. application code + database
- D. application code

# Q2

- Where do privacy policies get set?

- A. client side code
- B. backend server code
- C. database (specifically, the backend store interface)

# Q3

- What are the logical components of the multiverse database model?

- A. One base database
- B. Many base databases (one for each user)
- C. One base database + many user universes
- D. None of the above

# Q4

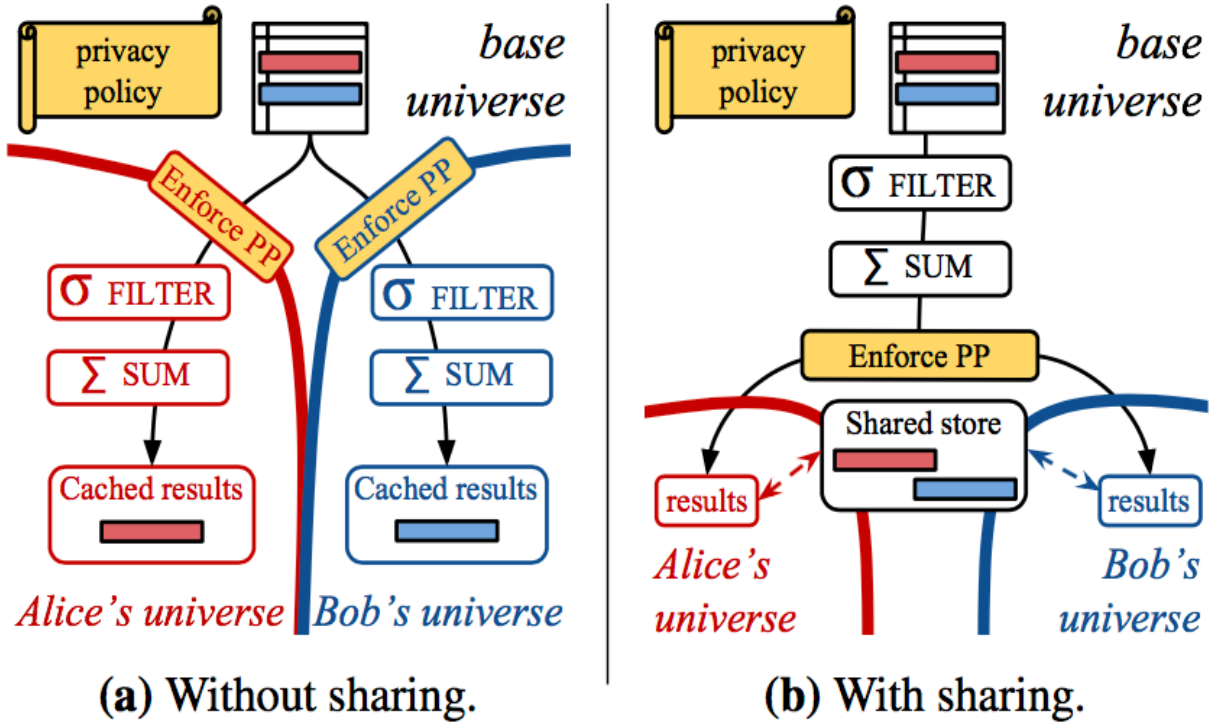
- where is the base universe, where are the user universes?

- A. both in the cache (memory)
- B. both in the disk
- C. disk, cache, respectively
- D. cache, disk, respectively

# Q 5

- What is Enforcement Operator?

- A. just another name for "**Relational Set Operators**", such as Union, Join…
- B. just another name for "**Dataflow Operator**"
- C. special dataflow vertices, that compute and apply the **privacy policy**'s effect (e.g., pass, discard, transform) for each record that flows through them




**(a)** Without sharing. **(b)** With sharing.

**Figure 2.** A multiverse database realized as a joint dataflow. For efficiency, universes can share computation and state (§4.2), as (b) shows for an identical query issued by Alice and Bob (here, without any group universes).

# Q 6

- What if we compute each user universe only during read query execution?

- A. exploding disk space
- B. exploding memory space
- C. high latency

# Q 7

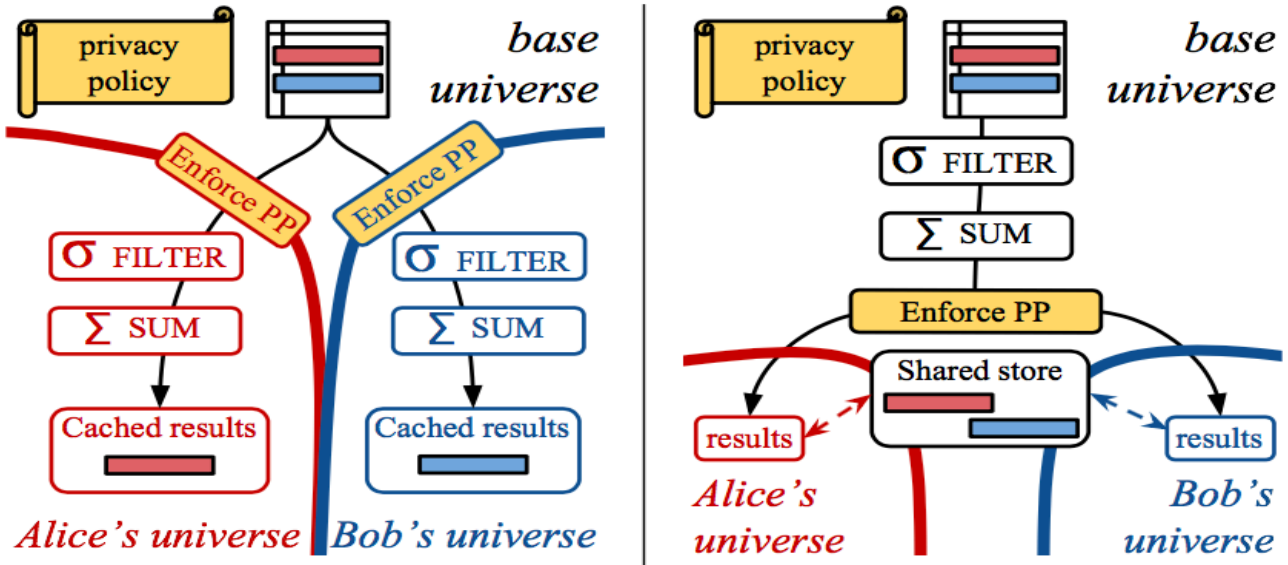
- What if we precompute ***all*** user universes in their ***entirety***?

- A. prohibitive disk space
- B. prohibitive memory space
- C. high latency
- D. prohibitive memory space and update costs

# Summary && Open question

- Q8 What might be the major challenges for the multiverse database to be practical?

# Part 2 Make Multiverse Practical




**(a)** Without sharing. **(b)** With sharing.

**Figure 2.** A multiverse database realized as a joint dataflow. For efficiency, universes can share computation and state (§4.2), as (b) shows for an identical query issued by Alice and Bob (here, without any group universes).

# 2.1 Group policies

- Q. Can "user group" such as "TAs for CS1470" have its own privacy policies?
- A.Yes
- B.No

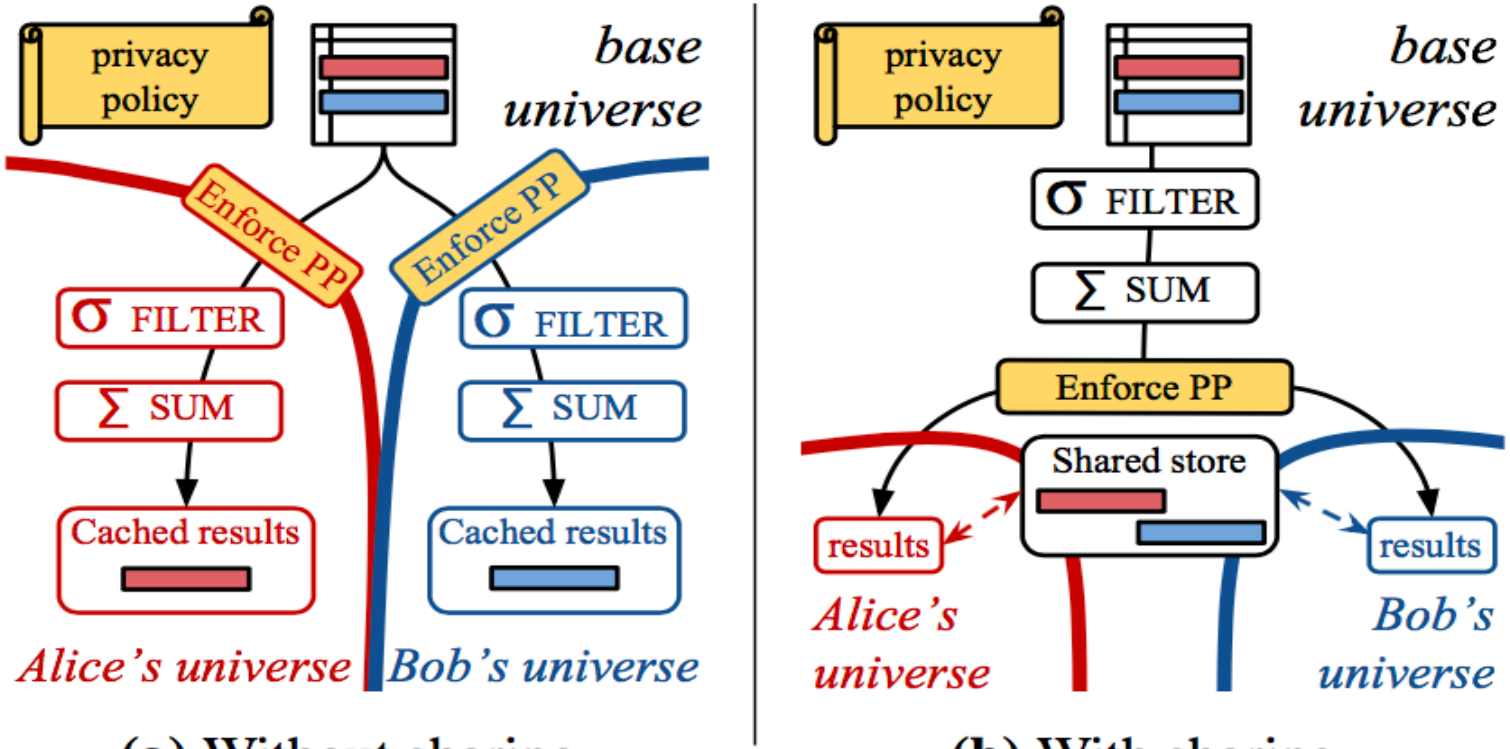
- Q True or False: Multiverse computes the privacy policy for "TA group for CS1470" *once on the boundary to each group member's user universe.*
- A. True
- B. False

# 2.1 Group Policy

```
group: "TAs",
-- define TA group for each class
membership: SELECT uid, class_id AS GID FROM Enrollment
            WHERE role = "TA",
policies: [
-- show anonymous posts to TAs
-- (`ctx` is a group universe context, holds the group ID)
  { table: Post,
    allow: WHERE Post.anonymous = 1 AND ctx.GID = Post.class } ],
```

# 2.2 Sharing between queries

- Q. Could many queries possibly share some privacy policies and results?
- A. Yes
- B. No
- Q. Can the system detect such sharing and merge identical dataflow paths?
- A. yes
- B. no




**(a)** Without sharing. **(b)** With sharing.

**Figure 2.** A multiverse database realized as a joint dataflow. For efficiency, universes can share computation and state (§4.2), as (b) shows for an identical query issued by Alice and Bob (here, without any group universes).

# 2.3 Sharing across universes

- Identical queries on behalf of many users
- E.g., 10 most recent posts to the class
- Results overlap in part:  public posts
- "Key Idea: prevent storing copies of identical records in many universes!"
- Question:  How to achieve this?

# 2.4 Partial Materialization

- Q The data-flow operator state is:
- A. windowed (can you explain what windowed mean?)
- B. partial (can you explain what partial mean?)

# “Partially-statetful”

- The partially-stateful dataflow model allows the system to choose dynamically what results to precommpute and cache, and how much computation to perform during read query execution.

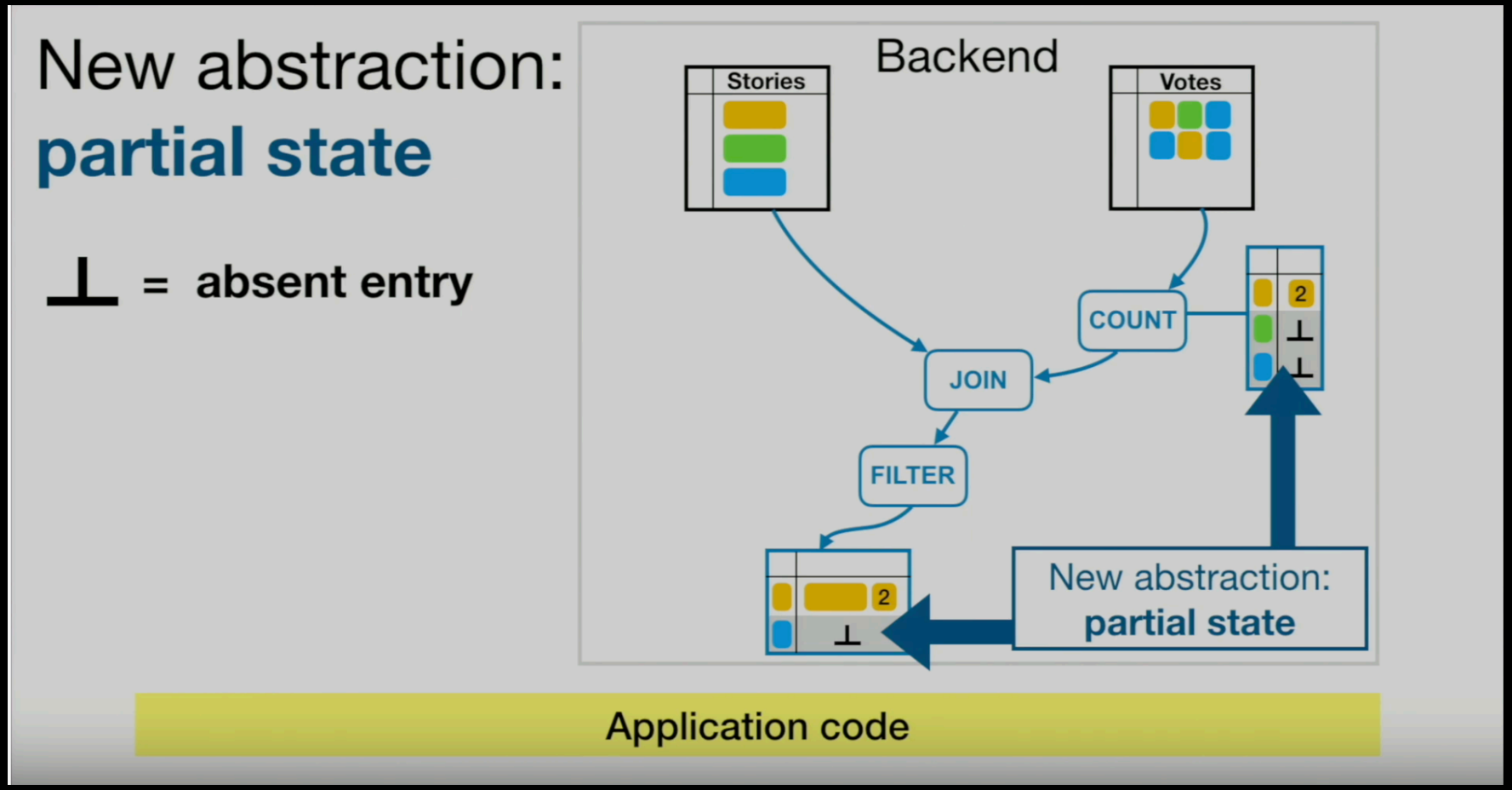
- → “Materialization Frontier”

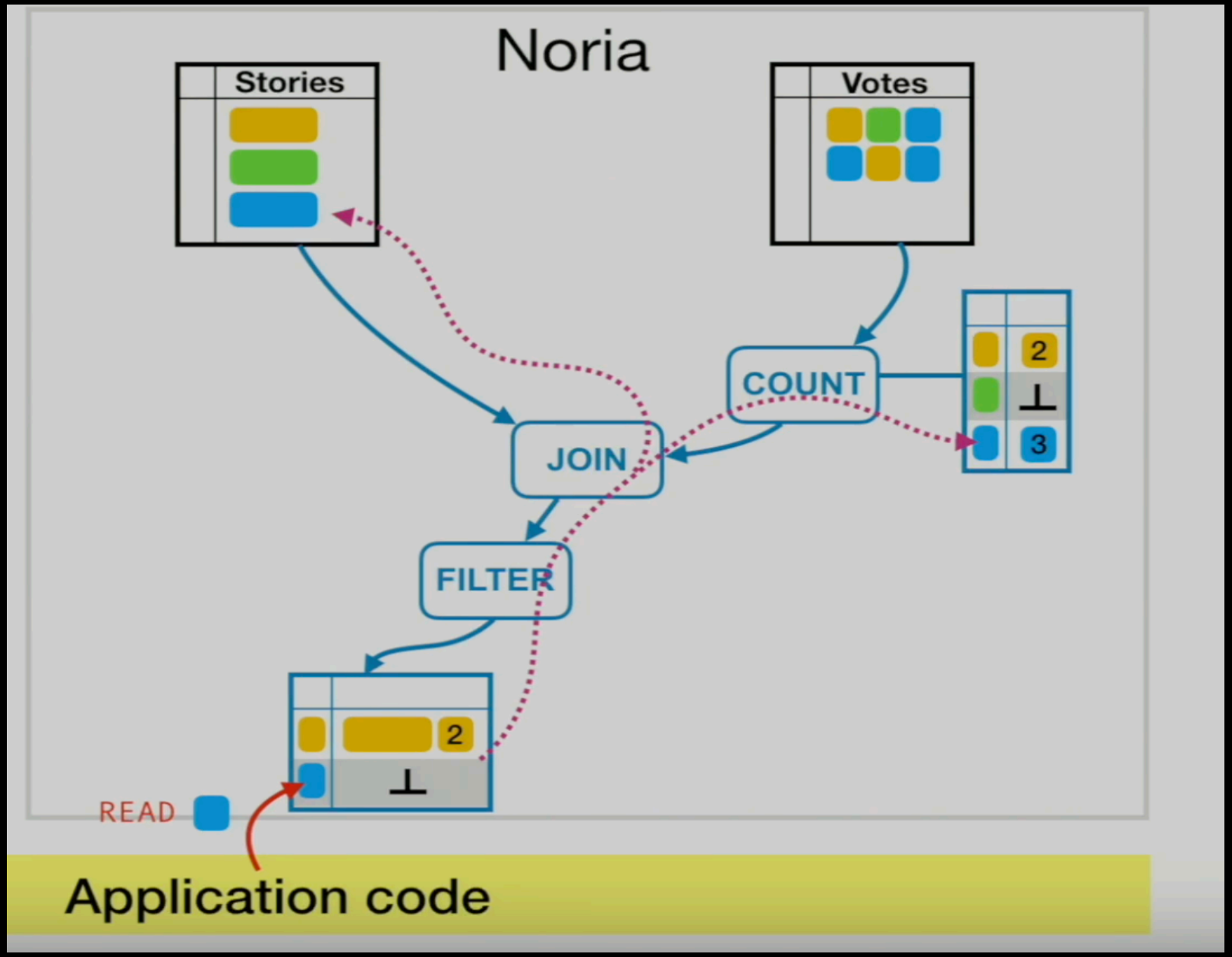
- →Benefits of partial state?

# Absent states

- Q When querying, if there are some missing states for an dataflow operator, what would happen?
- A. operator directly requests from the disk for the missing states
- B. operator requests an upquery

# Partially-stateful dataflow model



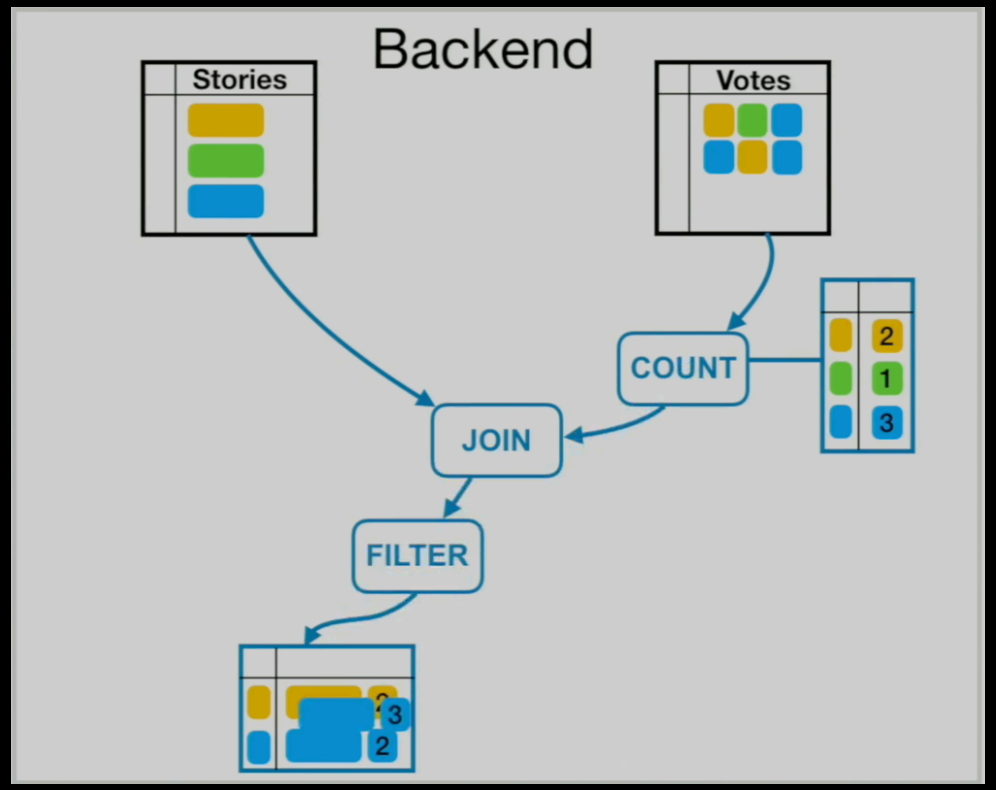

Noria
Stories
Votes
COUNT
2
⊥
3
JOIN
FILTER
2
⊥
READ
Application code

# Write

- Q When writing data into database:

- A. write to the base universe, then update user universes
- B. write to the user universe , then update base universe and other user universes
- C. merely write to the base universe

# Open question: When writing, how do we update the user universes?



# Key idea:

- compute/update only what's actually needed!

# Universe Creation

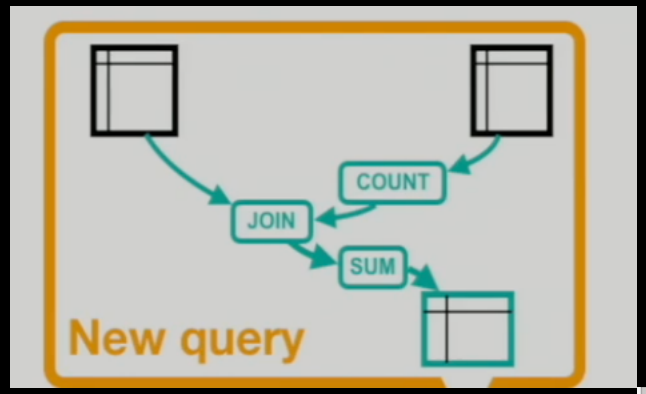
- Q1. when do we create a user universe?
- A. when user account is created
- B. when user log in
- Q2. when do you destroy a user universe?
- A. when user logs out
- B. when user deletes his/her account
- Q3. when we start a universe (but before query comes for that user), is the user universe partially full, entire full, or empty?
- A. partially full  B. entire full  C. empty

# New query

- Q. What happens when query changes (or, a new query comes)?

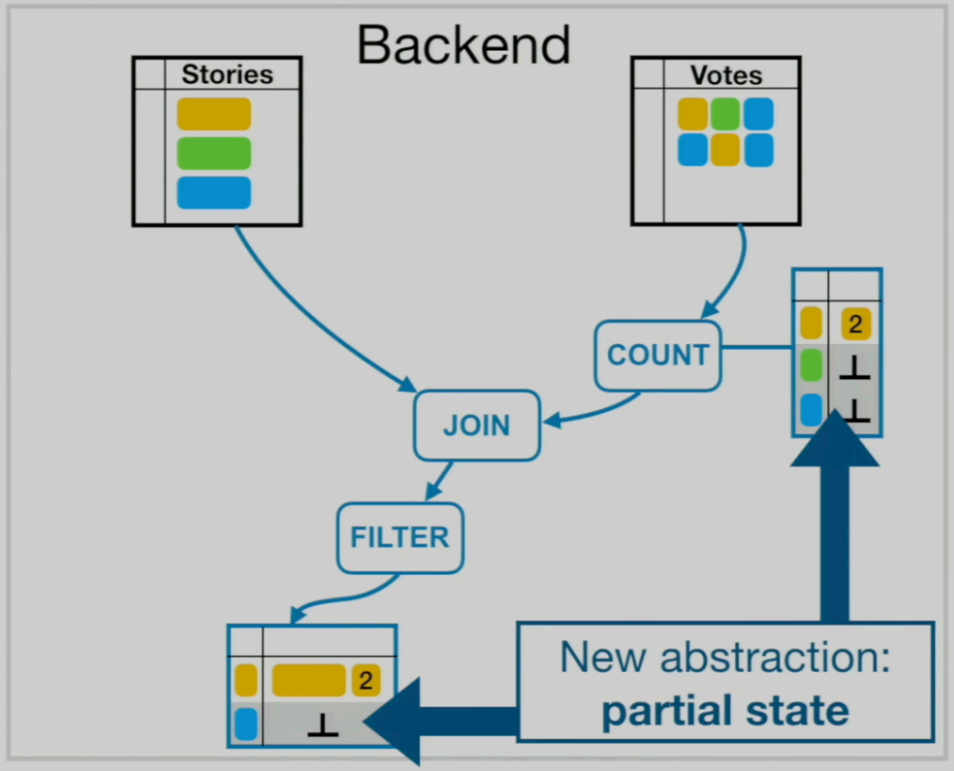
- A. perform the SQL query on base database and store the result in corresponding user universe
- B. detecting overlapping queries, reuse state and dataflow, add new operators, start with new state absent, use **upqueries** to fill new state lazily over time

- *Follow-up: Same as "Multi-query optimization problem"?*

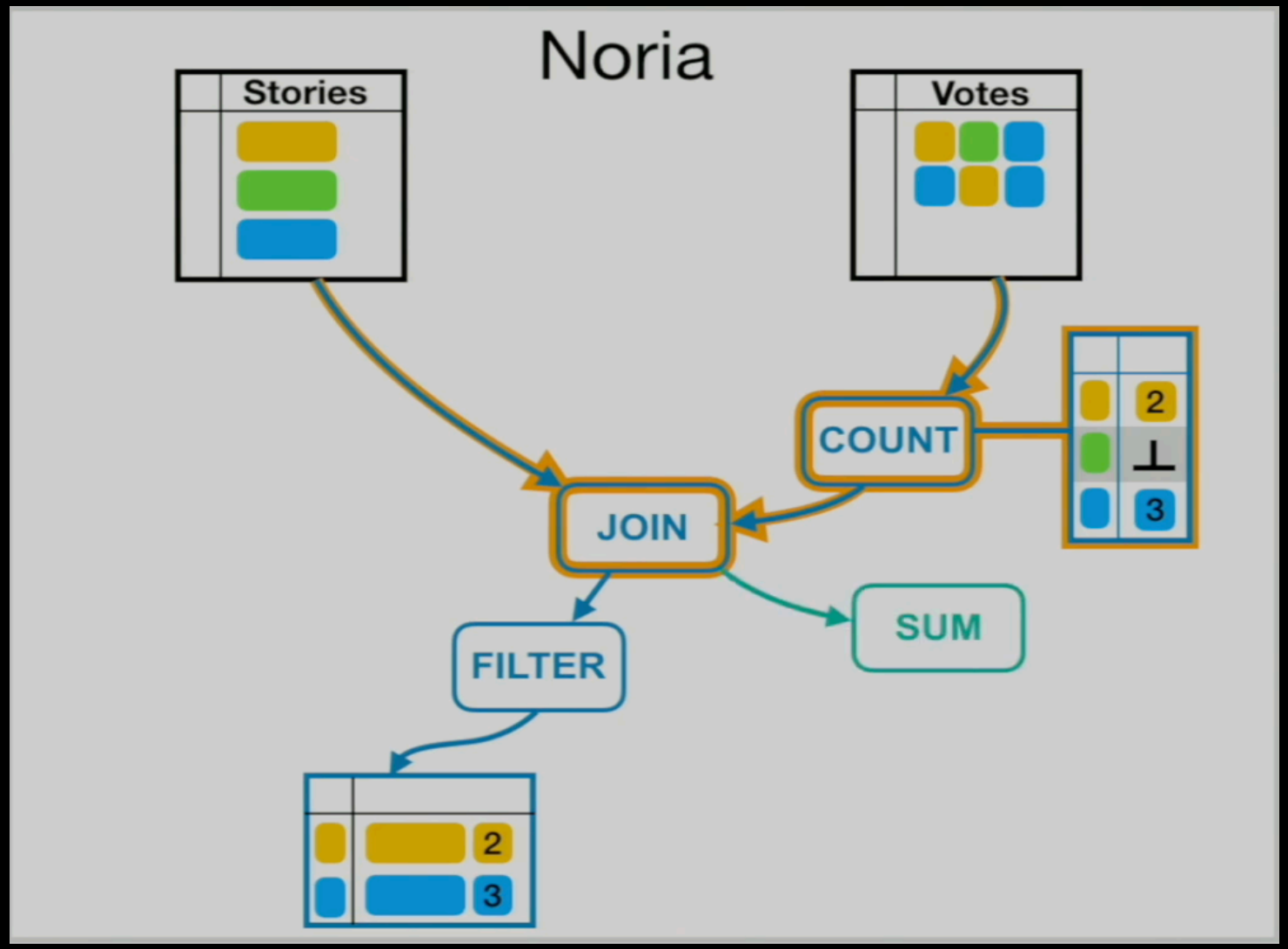


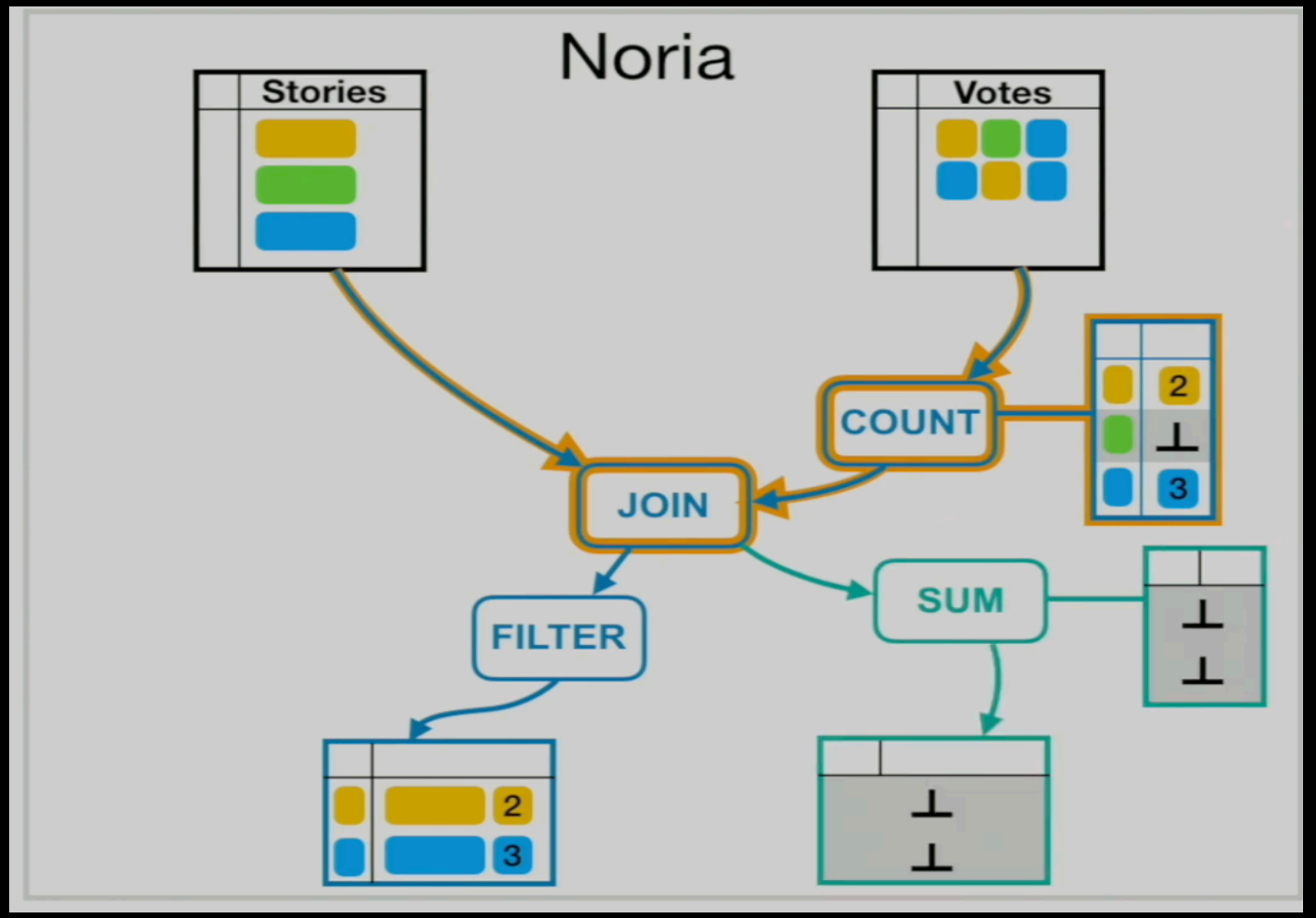

# New abstraction: partial state

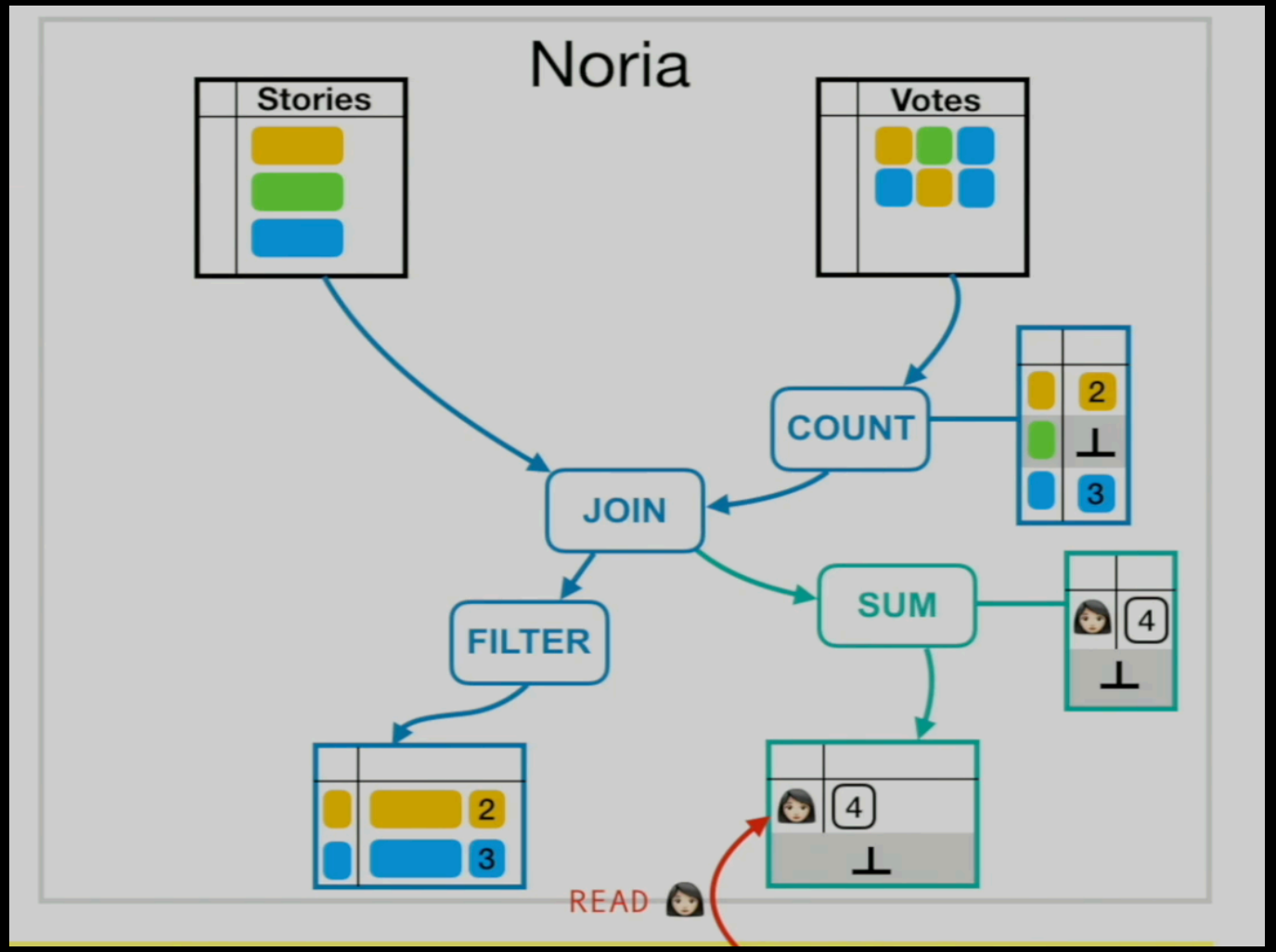
⊥ = absent entry




Application code


Noria
Stories
Votes
COUNT
2
⊥
3
JOIN
FILTER
SUM
2
3


Noria
Stories
Votes
COUNT
2
⊥
3
JOIN
FILTER
SUM
⊥
⊥
2
3
⊥
⊥


Noria
Stories
Votes
COUNT
2
⊥
3
JOIN
FILTER
SUM
4
⊥
2
3
4
⊥
READ

# State Eviction

# Consistency

- Q. Is it possible to multiverse databases to support **strong consistency**?

- A. Yes
- B. No

# Proof of Concept

- Extension to Noria
- Yes: Row suppression, rewrite, group policies, sharing
- No: Shared record store
- Piazza-style class forum
- Privacy policy: allows TA to see anonymous posts on a database containing 1M posts and 1000 classes.
- 5000 active user universes
- MySQL with privacy policies inlined in the query
- MySQL without any privacy policies

# Proof of concept

| | reads/sec | writes/sec |
|---|---|---|
| Multiverse database | 129.7k | 3.7k |
| MySQL (with AP) | 1.1k | 8.8k |
| MySQL (without AP) | 10.6k | 8.8k |

**Figure 3.** Our prototype achieves high read throughput compared to MySQL queries that execute privacy policies inline. Write throughput is lower than MySQL's as the multiverse database does more work on writes.

- Evaluating the privacy policy as part of the query slows down MySQL reads by 9.6X
- Write output: half of what MySQL supports
- Process memory use: from 0.5 GB (one universe) to 1.1GB (5000 universes)
- This 600MB footprint is about half of the 1.2GB needed without group universes
- (a separate benchmark showed that using a shared record store for identical queries reduces their space footprint by 94%).

# Future Research Directions

- Write authorization policies
- User-defined policy operators
- Policy Correctness
- Verified policy compilation

# Conclusion && Discussion

- 1 Easy to use
- 2 High performance
- 3 Acceptable overhead

- What do you think?