Deletion Framework

How Facebook upholds its commitments towards data deletion

Benoît Reitz - 2021

Introduction

Introduction

- Deletion Framework is responsible for graph deletions at Facebook
- Deletion Framework in numbers:
  - 20B new graph deletions per day
  - ~500B individual objects deleted per day
  - Handles a dozen of interconnected datastores
- A birds eye view of many systems rather than a deep dive

Outline

I. Design Overview
II. Guarantees
III. Monitoring Guarantees
IV. Conclusion

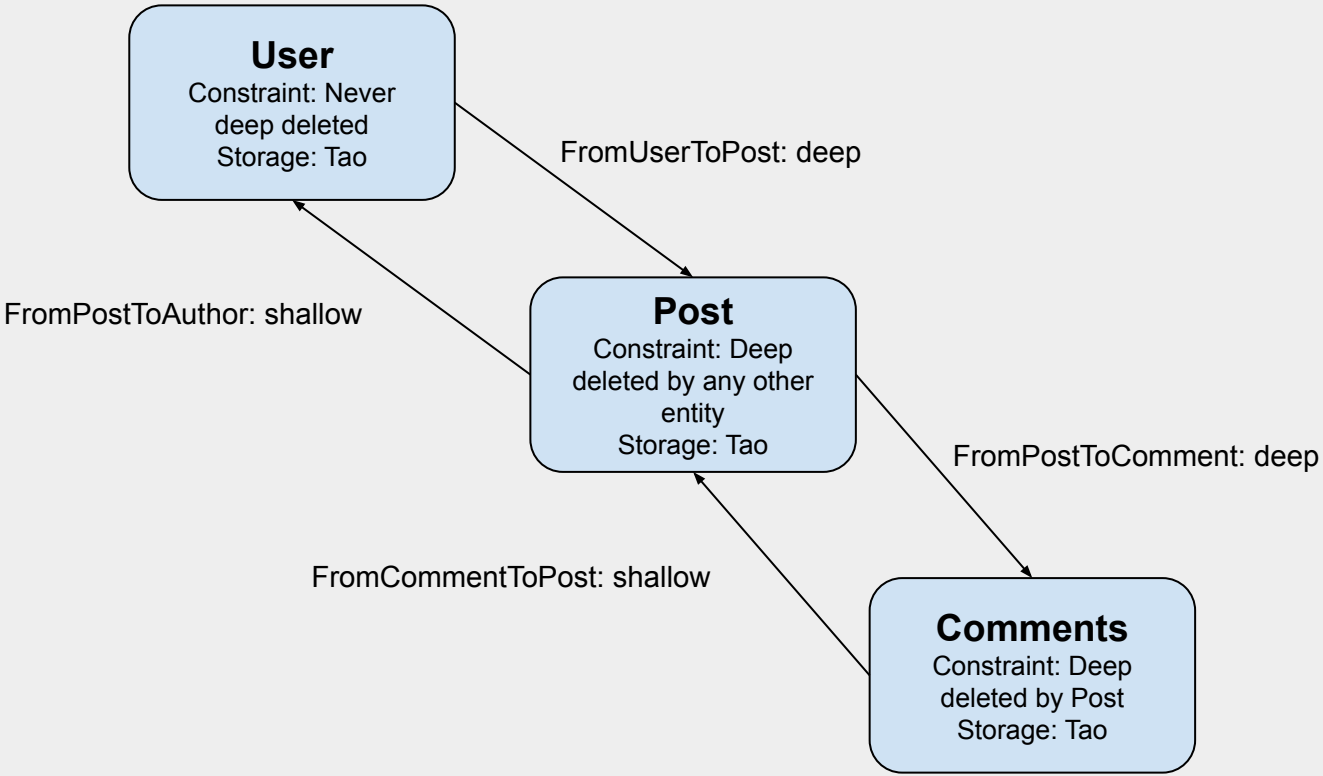
Deletion Framework

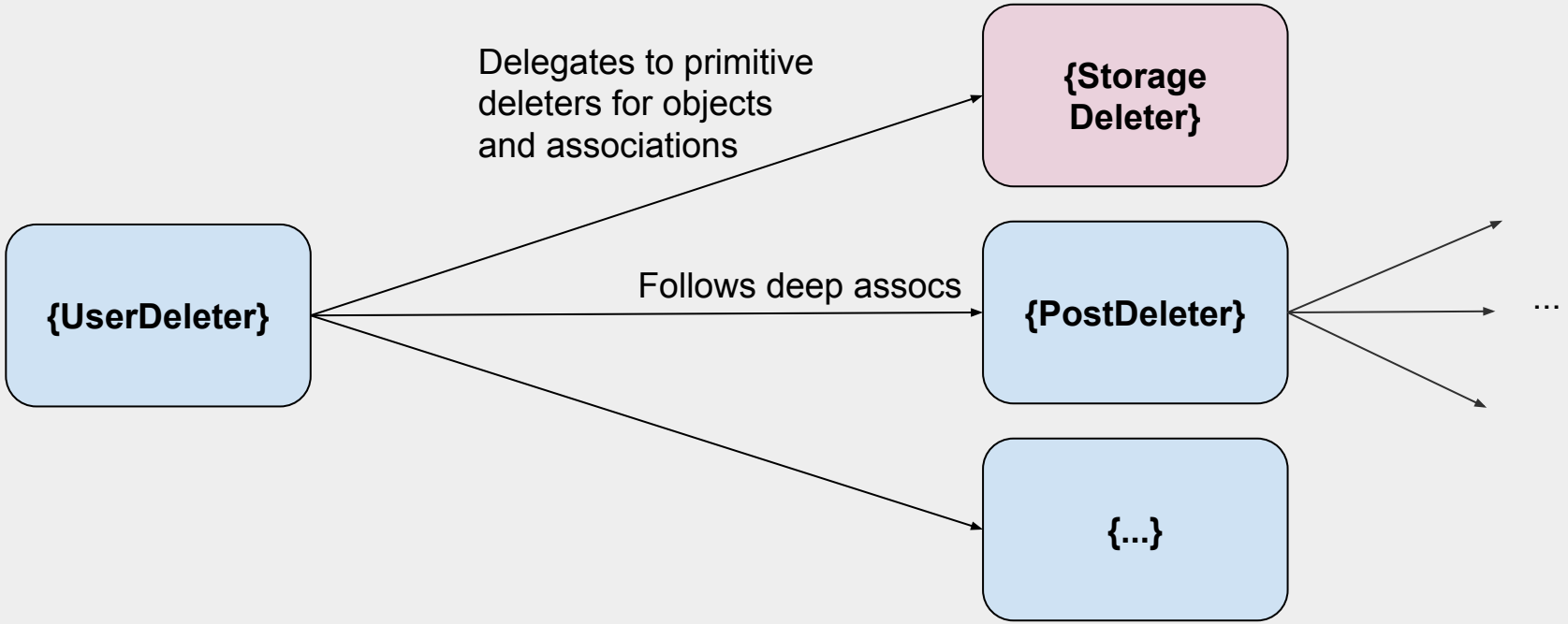
Outline

I. Design Overview
   A. Relying on a Data Definition Language
   B. Walking the graph
   C. Batching
   D. Scheduling deletions in the future
II. Guarantees
III. Monitoring Guarantees
IV. Conclusion

Relying on a Data Definition Language

- Code generation is a critical component
- Deletion logic is generated from annotations present on the schema
  - Storage configuration
  - Deletion Edge annotations
    - Shallow / Deep / Refcounted
  - Deletion Constraints


User
Constraint: Never deep deleted
Storage: Tao
FromUserToPost: deep
FromPostToAuthor: shallow
Post
Constraint: Deep deleted by any other entity
Storage: Tao
FromPostToComment: deep
FromCommentToPost: shallow
Comments
Constraint: Deep deleted by Post
Storage: Tao

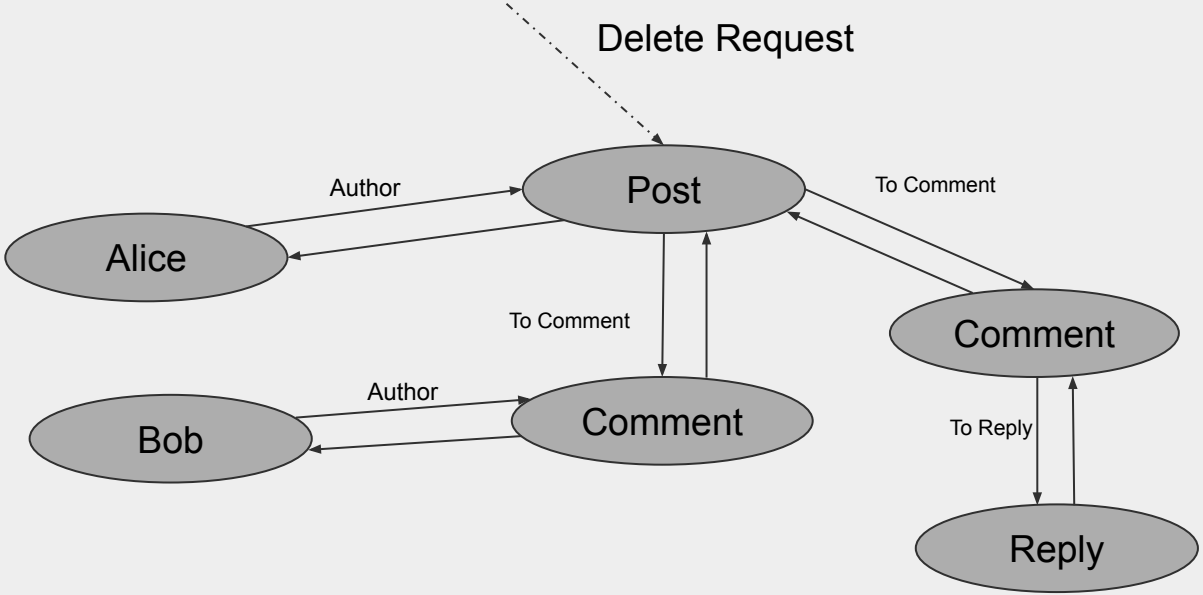

Delegates to primitive deleters for objects and associations
{Storage Deleter}
{UserDeleter}
Follows deep assocs
{PostDeleter}
...
{...}

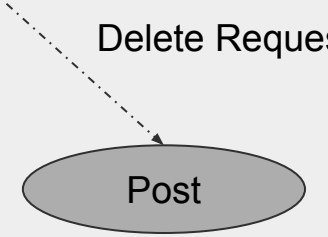
Outline

I. Design Overview
   A. Relying on a Data Definition Language
   B. Walking the graph
   C. Batching
   D. Scheduling deletions in the future
II. Guarantees
III. Monitoring Guarantees
IV. Conclusion

Walking the graph

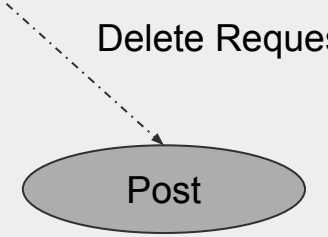
- We use a DFS
- We need to be re-entrant
- At least once semantics


Delete Request
Post
Author
Alice
To Comment
Comment
To Comment
Author
Bob
Comment
To Reply
Reply



Actions:

1. Read

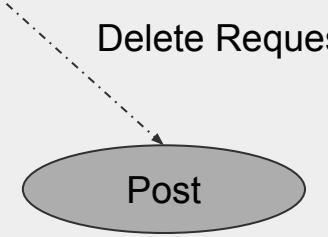


Actions:

1. Read
2. Checkpoint on stack

Persistent Stack:

- Post



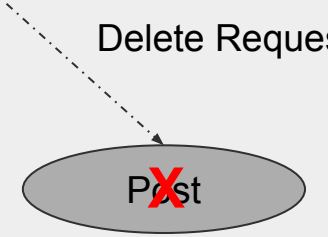
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs

Persistent Stack:

- Post

Restoration Logs:
Post



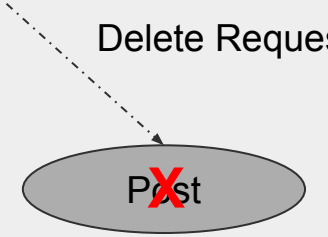
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Self Delete

Persistent Stack:

- Post

Restoration Logs:
Post



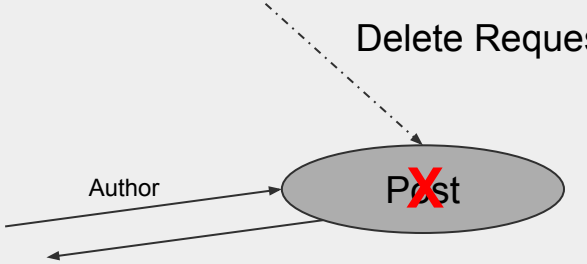
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Self Delete
5. Graph Delete

Persistent Stack:

- Post

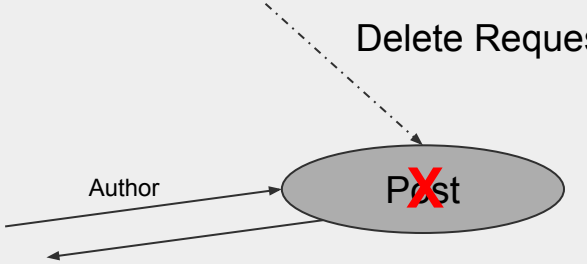
Restoration Logs:
Post



Actions:
1. Read

Persistent Stack:
- Post

Restoration Logs:
Post



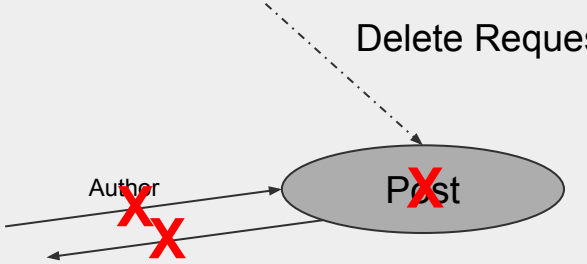
Actions:

1. Read
2. Log restoration logs

Persistent Stack:

- Post

Restoration Logs:
Post, Assoc From author



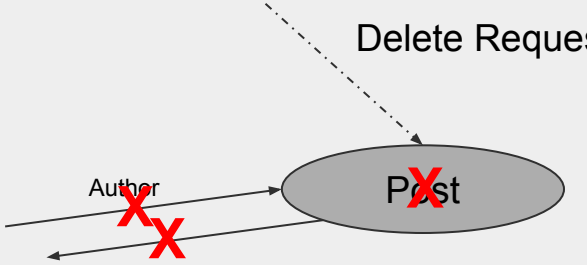
Actions:

1. Read
2. Log restoration logs
3. Delete

Restoration Logs:
Post, Assoc From author

Persistent Stack:

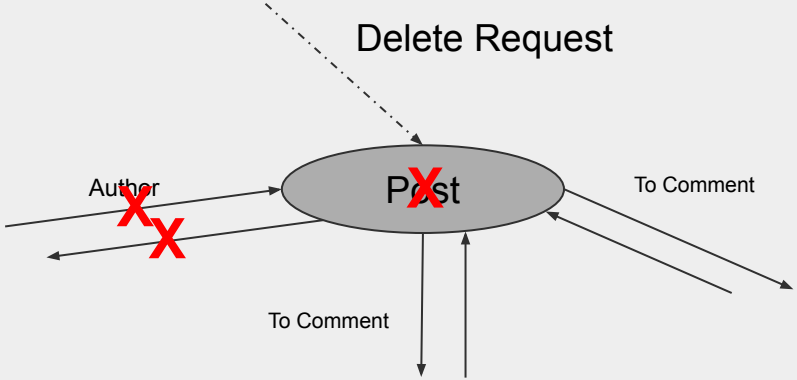
- Post



Actions:
1. Read

Restoration Logs:
Post, Assoc From author

Persistent Stack:
- Post



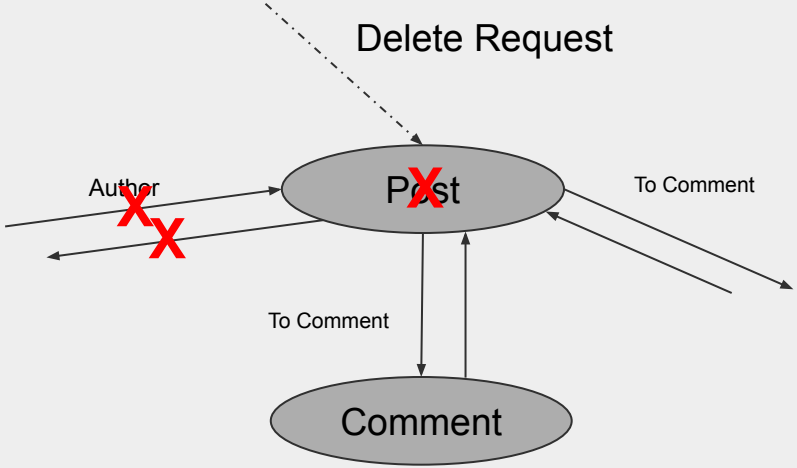
Actions:

1. Read
2. Traverse

Restoration Logs:
Post, Assoc From author

Persistent Stack:

- Post



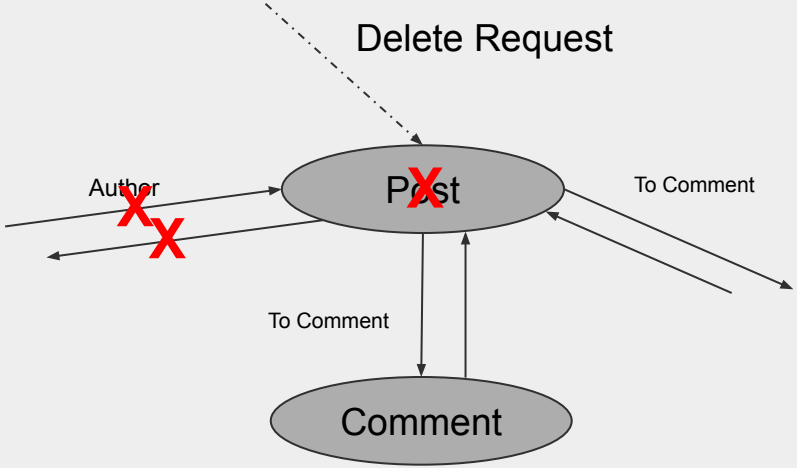
Actions:

1. Read

Restoration Logs:
Post, Assoc From author

Persistent Stack:

- Post



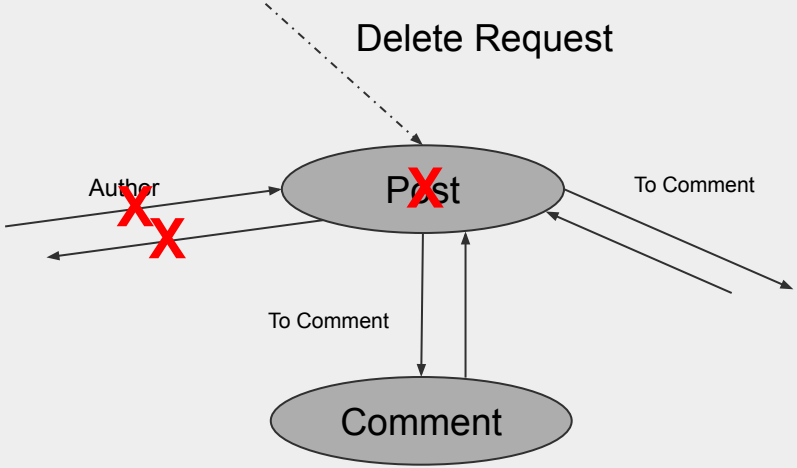
Actions:

1. Read
2. Checkpoint on stack

Restoration Logs:
Post, Assoc From author

Persistent Stack:

- Post
- Comment



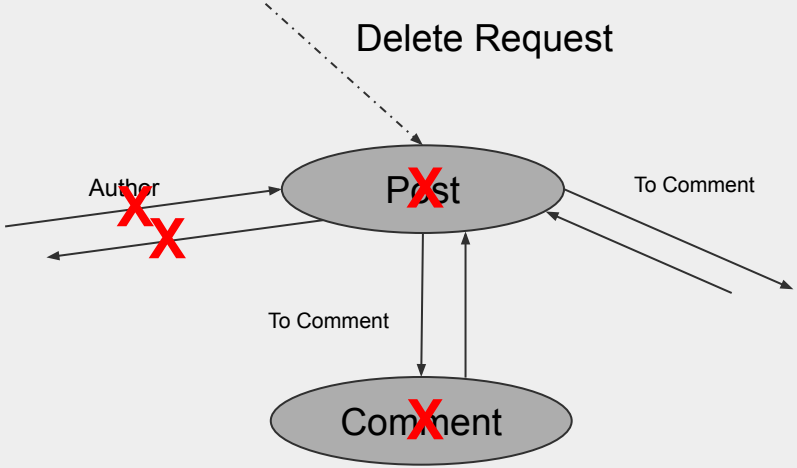
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs

Restoration Logs:
Post, Assoc From author, Comment

Persistent Stack:

- Post
- Comment



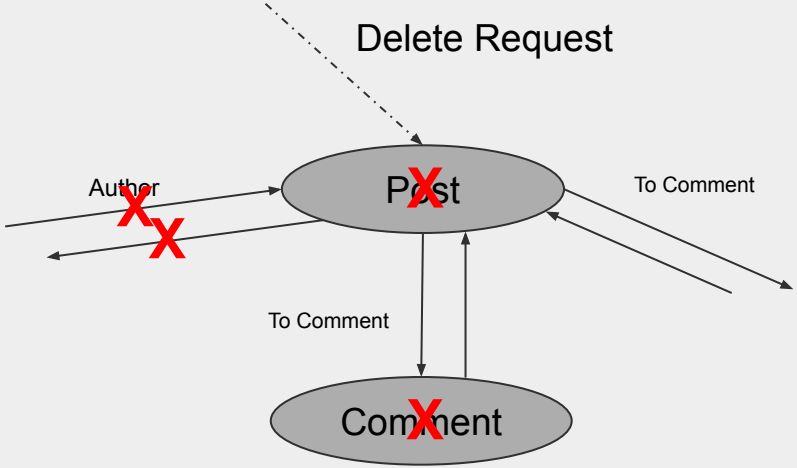
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Self Delete

Restoration Logs:
Post, Assoc From author, Comment

Persistent Stack:

- Post
- Comment



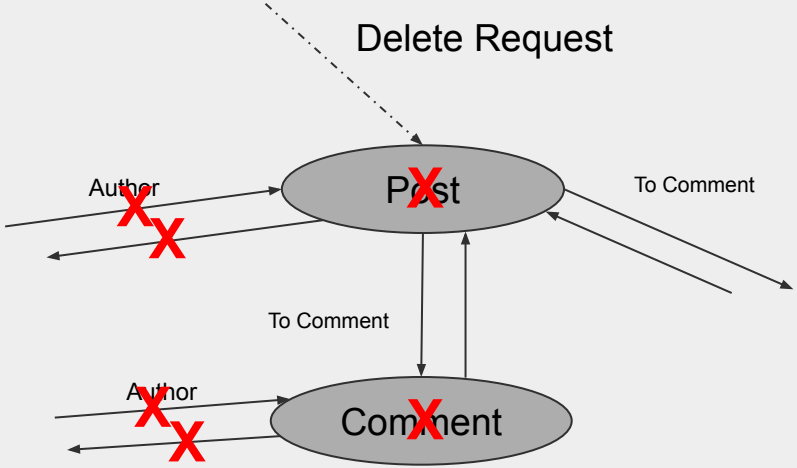
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Self Delete
5. Graph Delete

Persistent Stack:

- Post
- Comment

Restoration Logs:
Post, Assoc From author, Comment



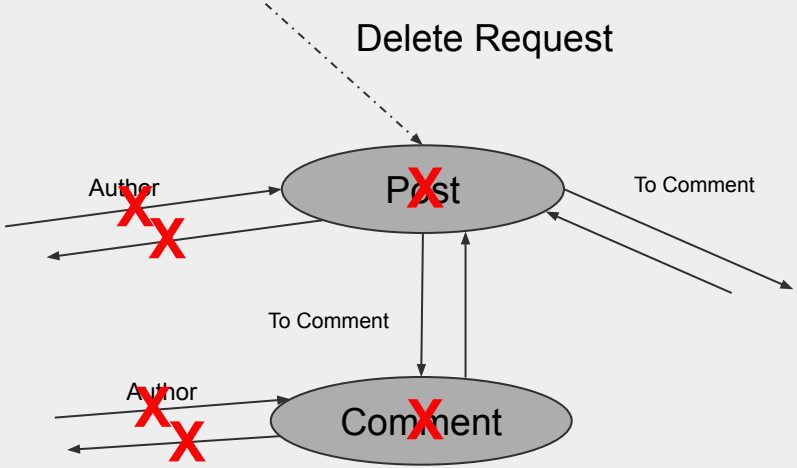
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Self Delete
5. Graph Delete

Restoration Logs:
Post, Assoc From author, Comment, Assoc From author

Persistent Stack:

- Post
- Comment



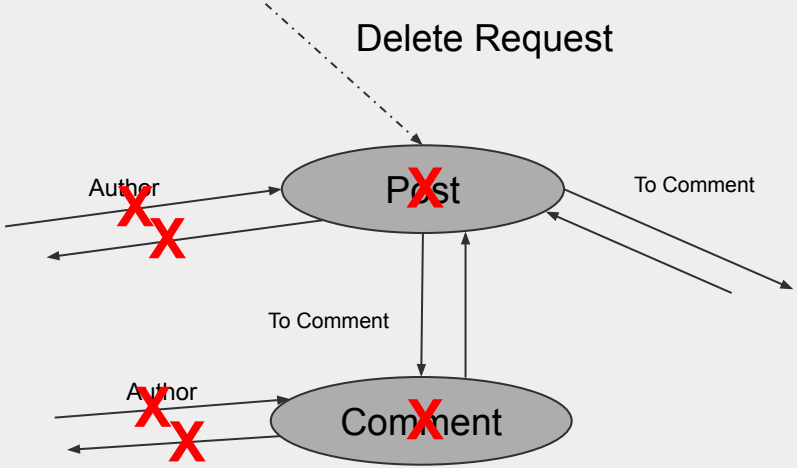
Actions:

1. Pop from stack

Restoration Logs:
Post, Assoc From author, Comment, Assoc From author

Persistent Stack:

- Post
- ~~Comment~~



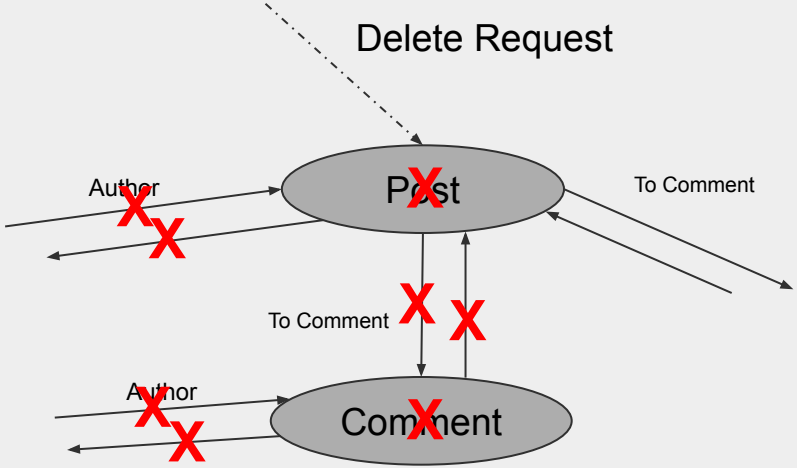
Actions:

1. Read
2. Traverse
3. Log Restoration Log

Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment

Persistent Stack:

- Post



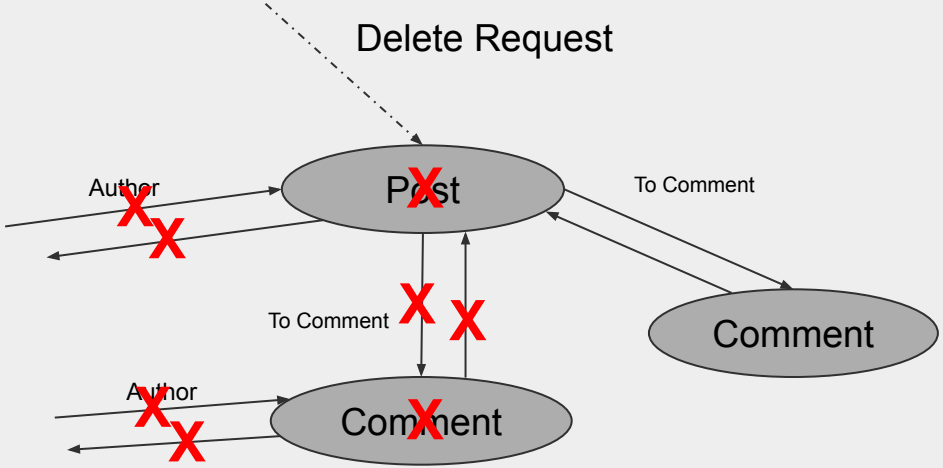
Actions:

1. Read
2. Traverse
3. Log Restoration Log
4. Delete

Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment

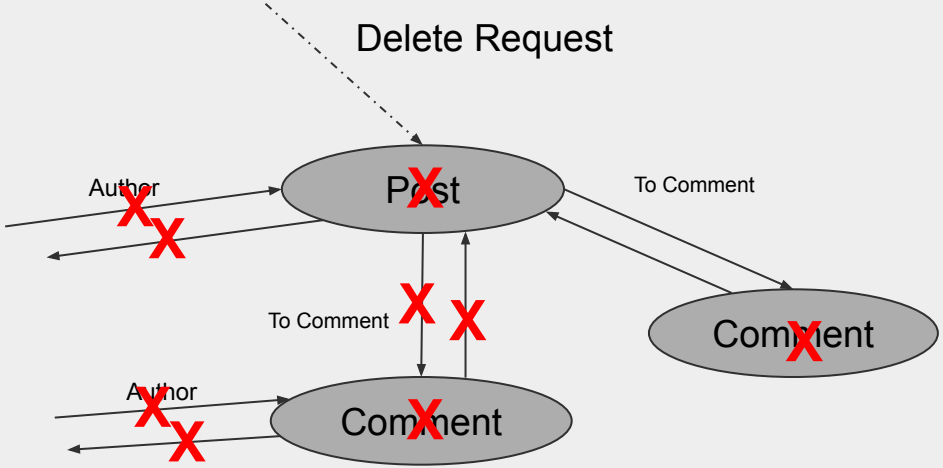
Persistent Stack:

- Post



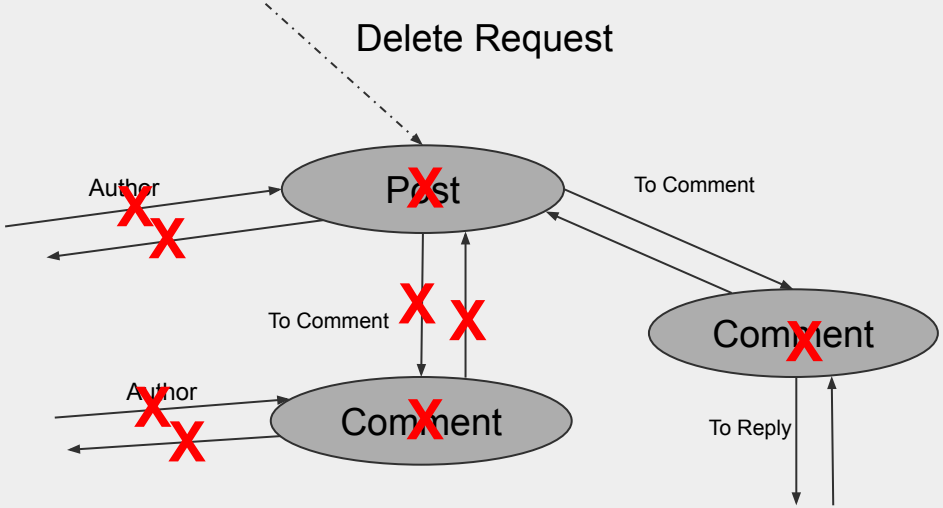
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment

Persistent Stack:
- Post
- Comment



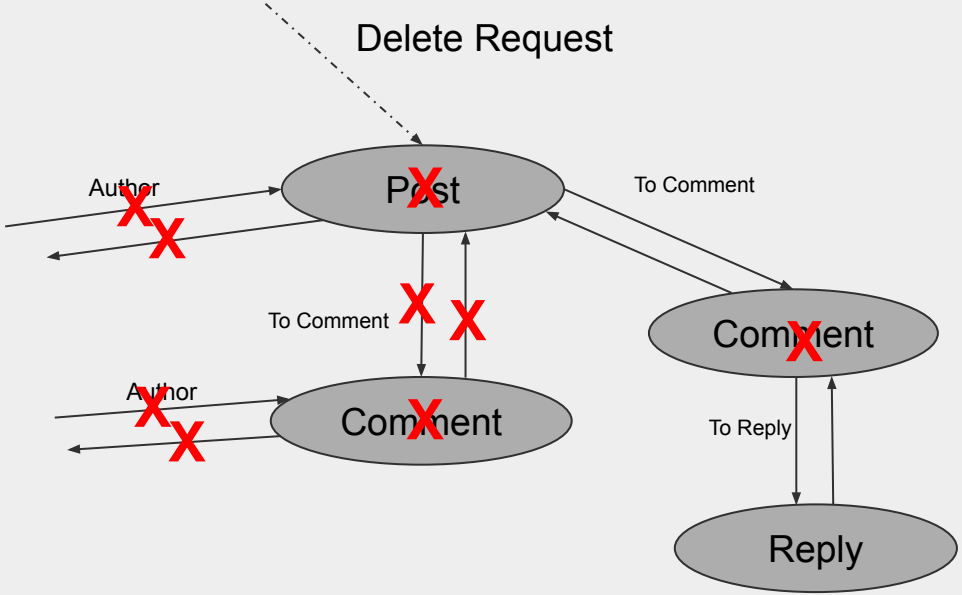
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment

Persistent Stack:
- Post
- Comment



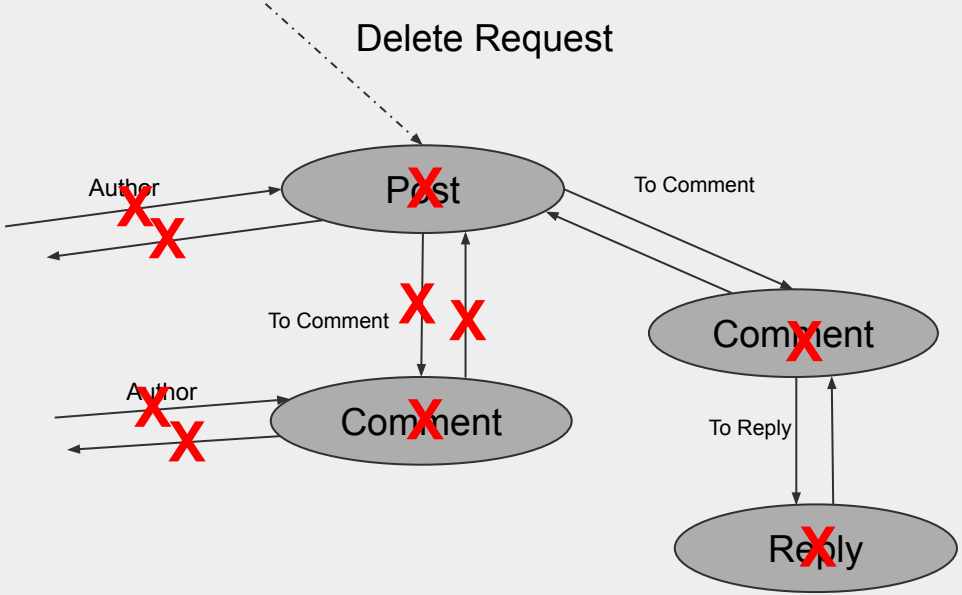
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment

Persistent Stack:
- Post
- Comment



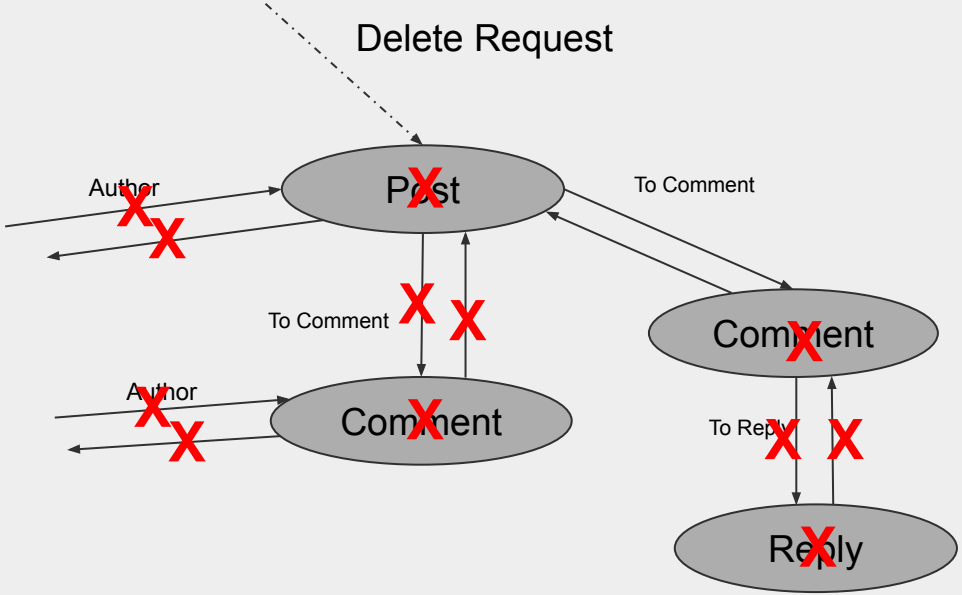
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment

Persistent Stack:
- Post
- Comment
- Reply



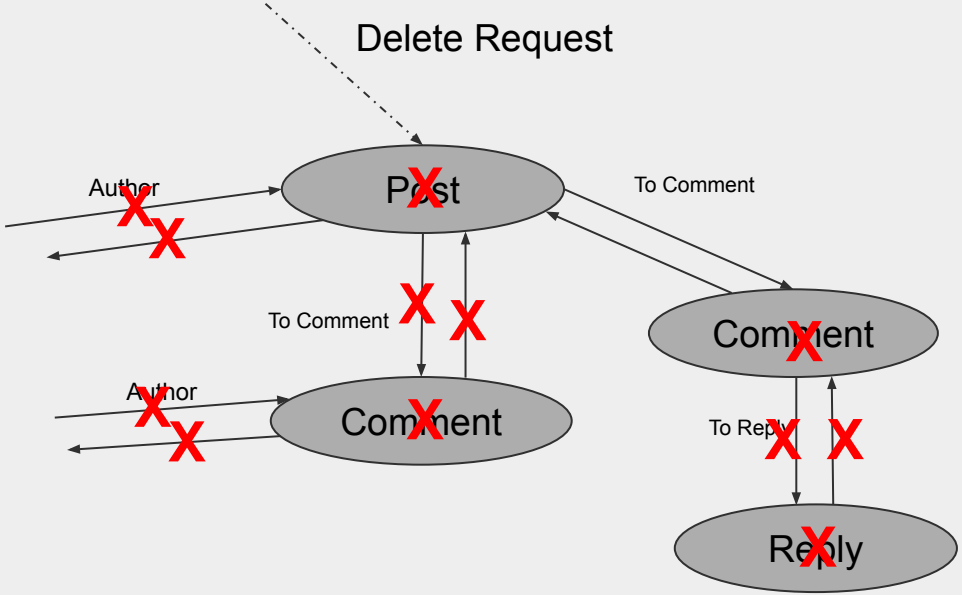
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment, Reply

Persistent Stack:
- Post
- Comment
- Reply



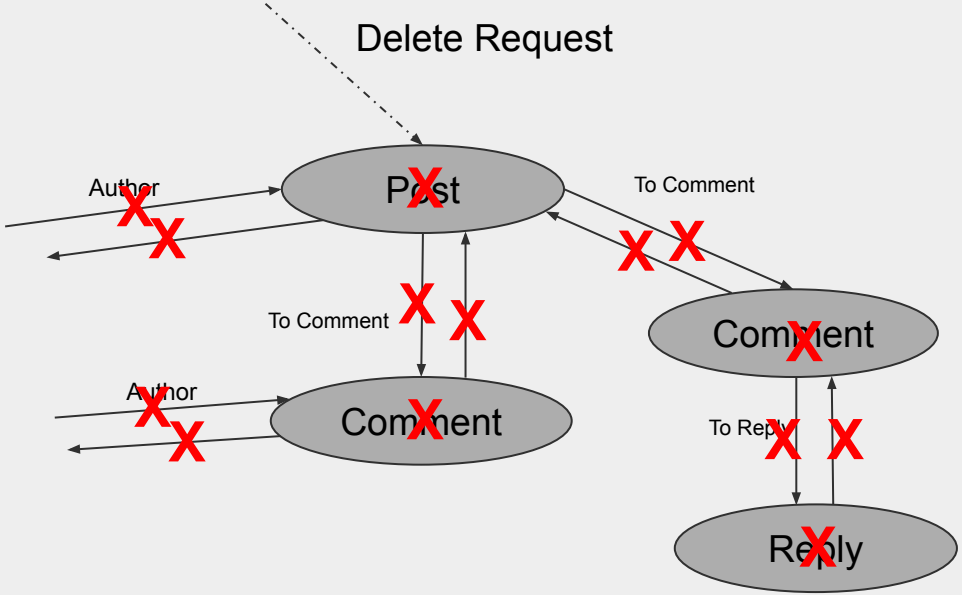
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment, Reply, ...

Persistent Stack:
- Post
- Comment
- ~~Reply~~



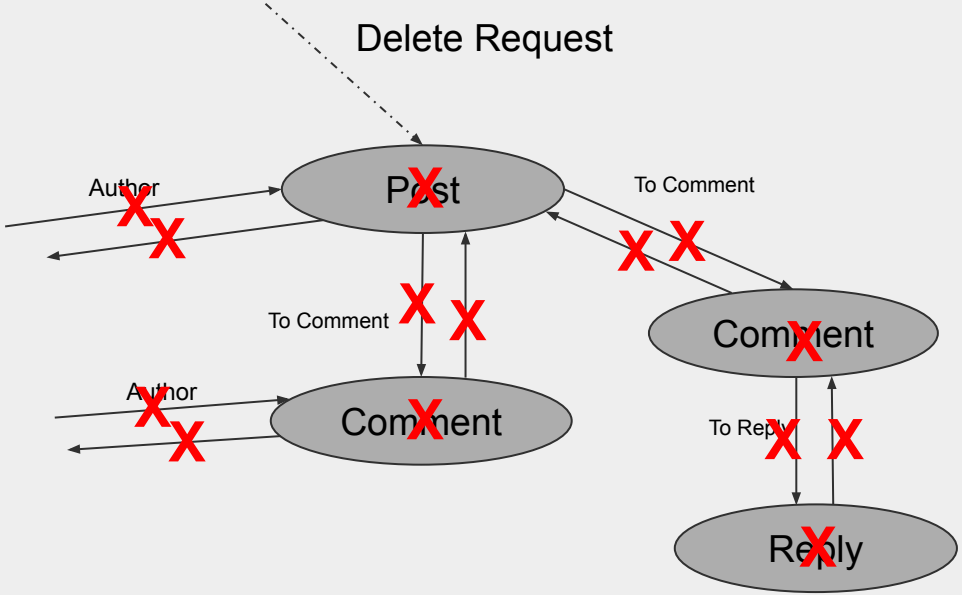
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment, Reply, ...

Persistent Stack:
- Post
- ~~Comment~~
- ~~Reply~~



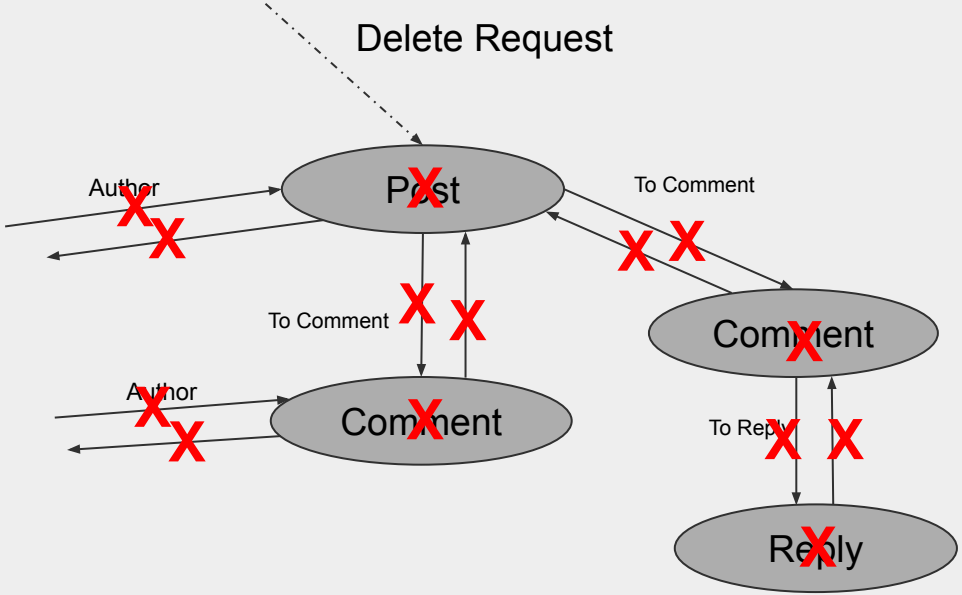
Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment, Reply, ...

Persistent Stack:
- Post
- ~~Comment~~
- ~~Reply~~



Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment, Reply, ...

Persistent Stack:
~~Post~~

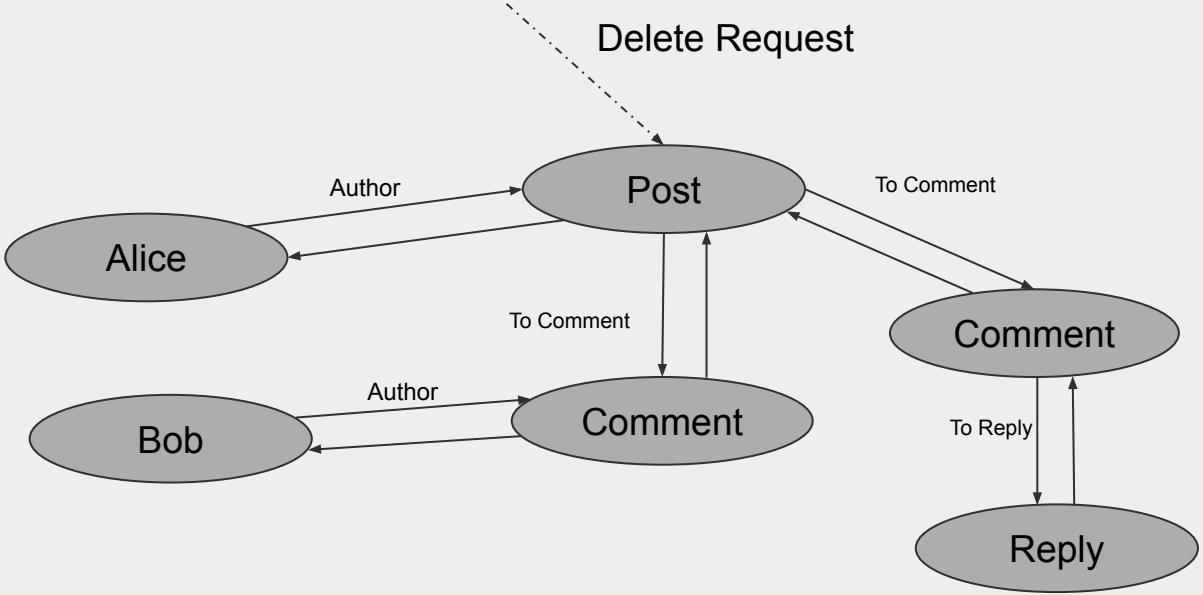


Done!

Restoration Logs:
Post, Assoc From author, Comment, Assoc From author, Assoc to Comment, Comment, Reply, ...

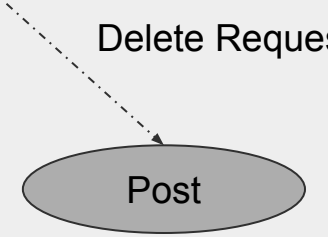
Outline

I. Design Overview
   A. Relying on a Data Definition Language
   B. Walking the graph
   C. Batching
   D. Scheduling deletions in the future
II. Guarantees
III. Monitoring Guarantees
IV. Conclusion



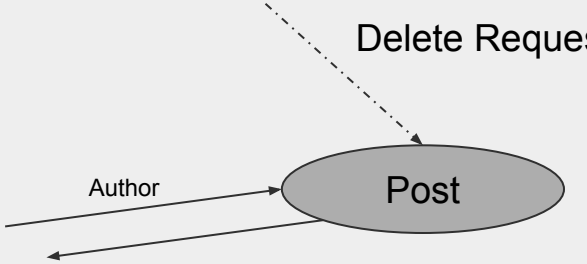
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Self Delete
5. Graph Delete



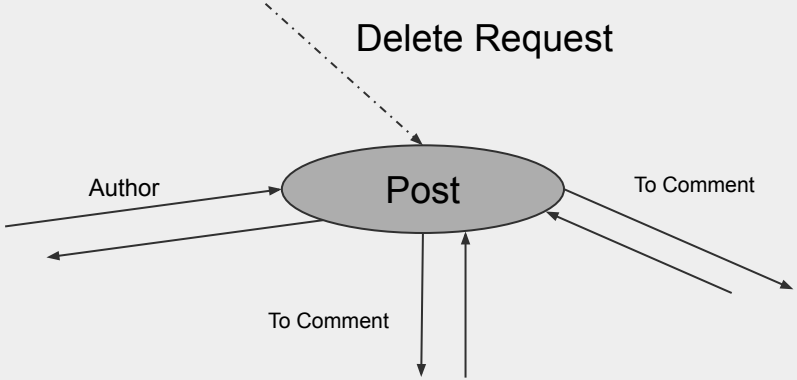
Actions:

1. Read



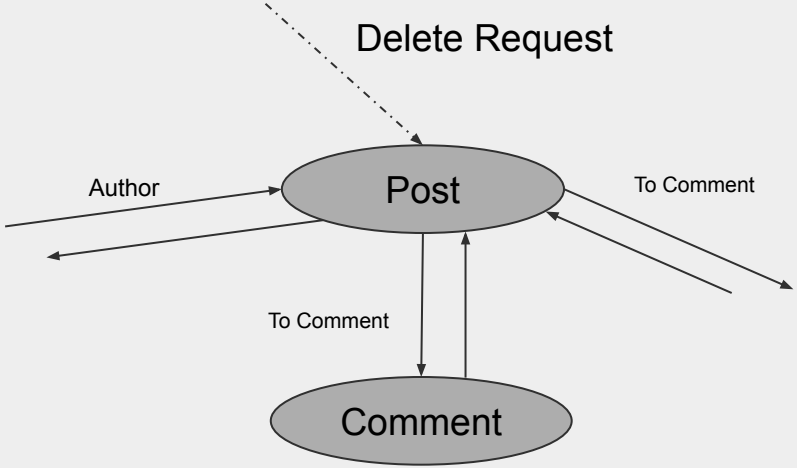
Actions:

1. Read



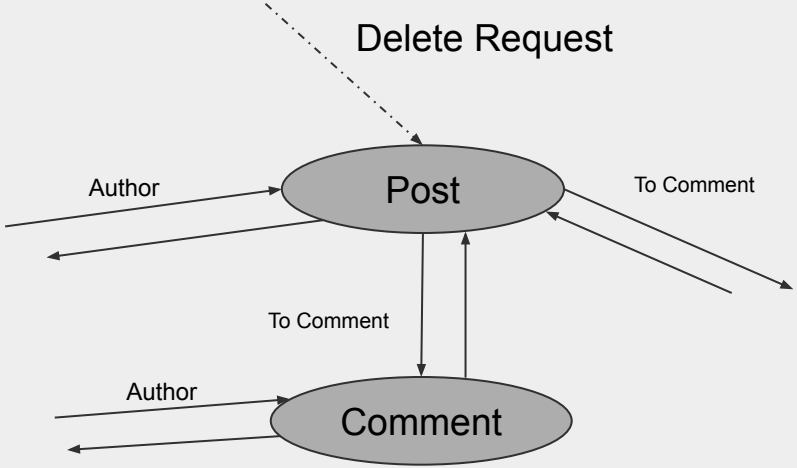
Actions:

1. Read



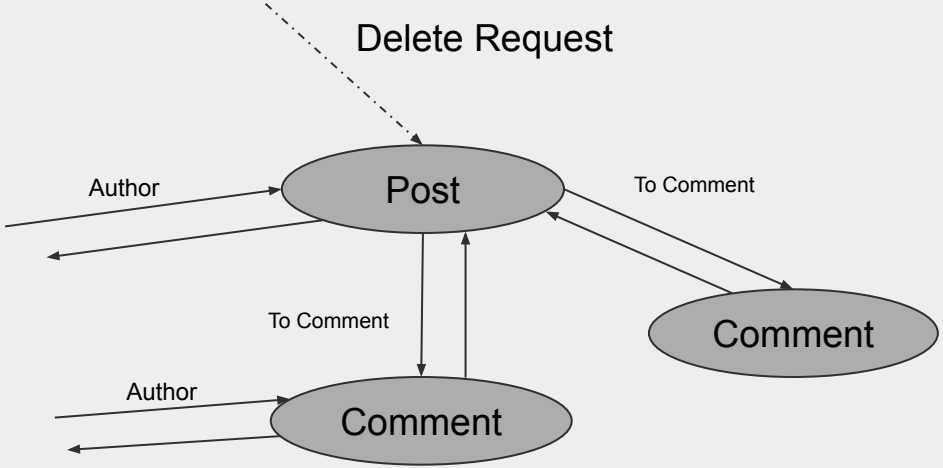
Actions:

1. Read



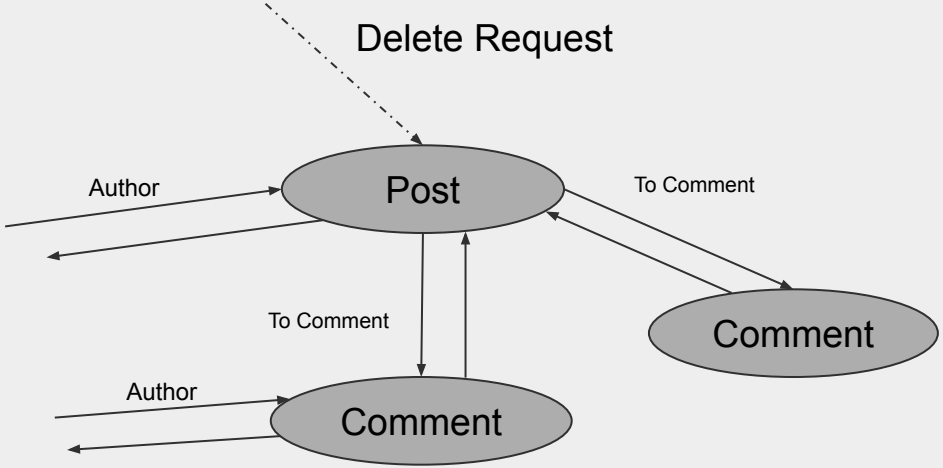
Actions:

1. Read



Actions:

1. Read

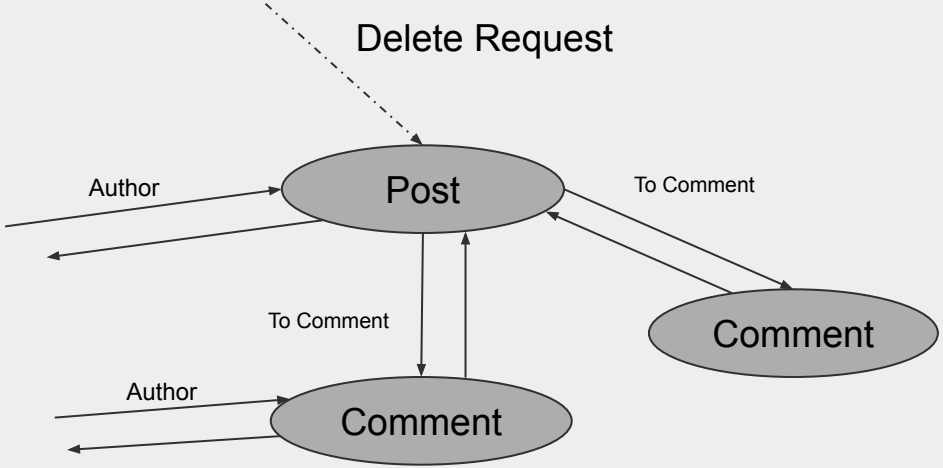


Actions:

1. Read
2. Checkpoint on stack

Persistent Stack:

- Post
- Comment
- Comment



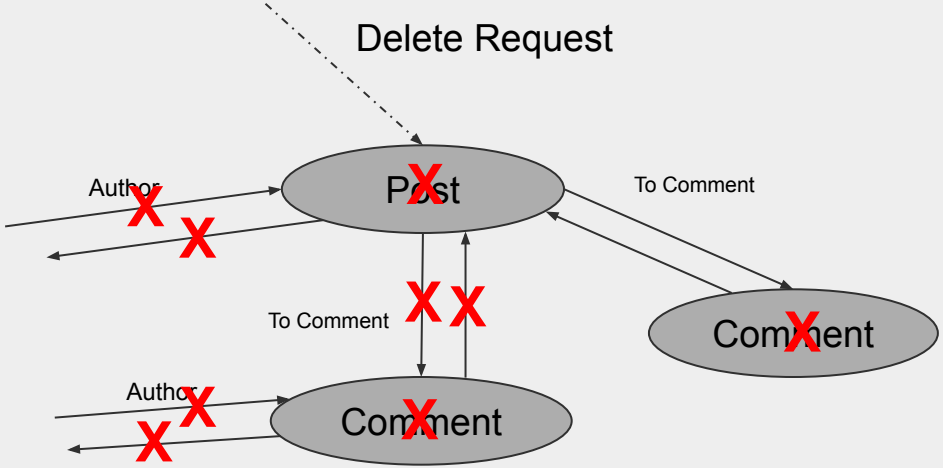
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs

Restoration Logs:
Batched Restoration Record

Persistent Stack:

- Post
- Comment
- Comment



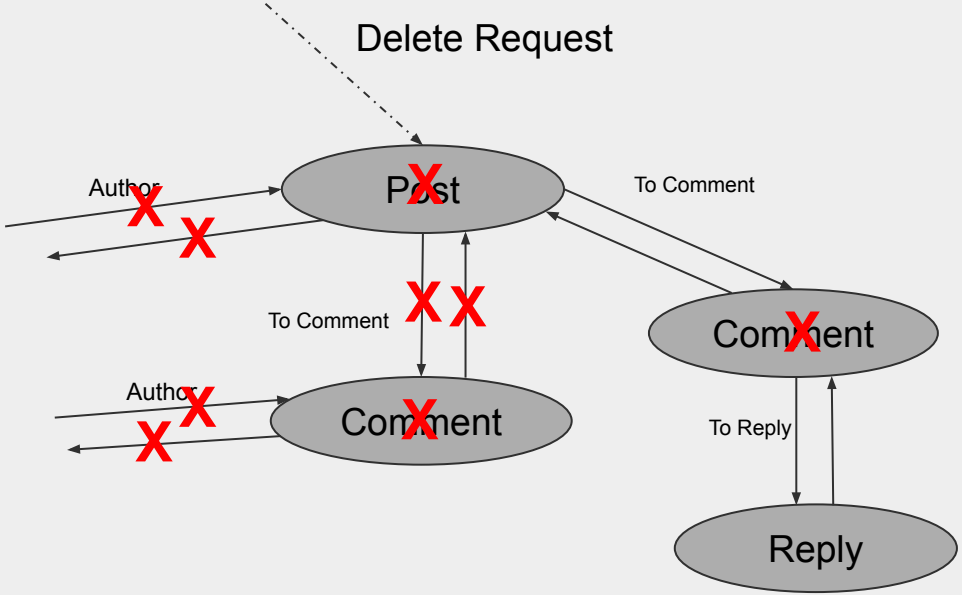
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Delete

Restoration Logs:
Batched Restoration Record

Persistent Stack:

- Post
- ~~Comment~~
- Comment



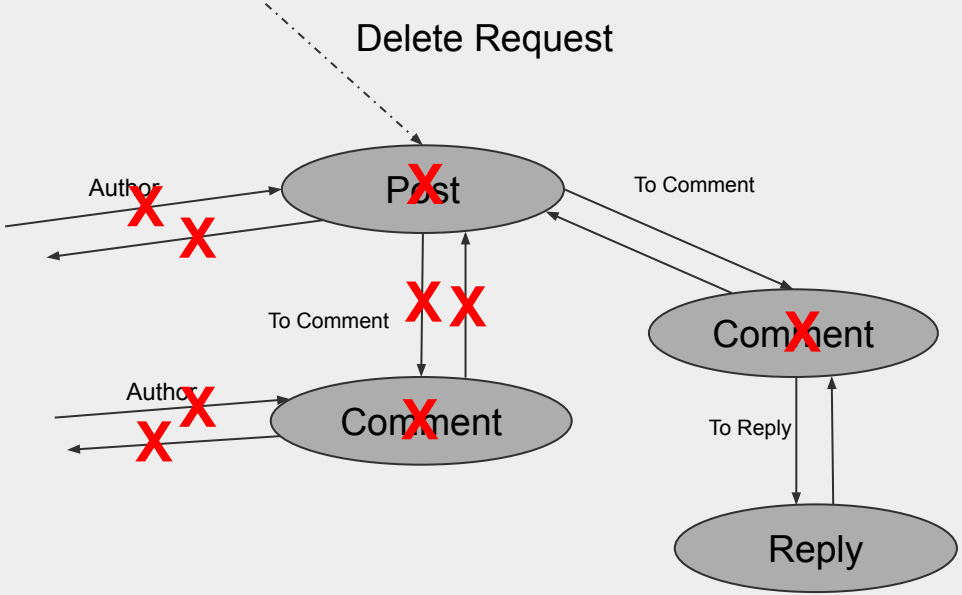
Actions:

1. Read
2. Checkpoint on stack

Restoration Logs:
Batched Restoration Record

Persistent Stack:

- Post
- Comment



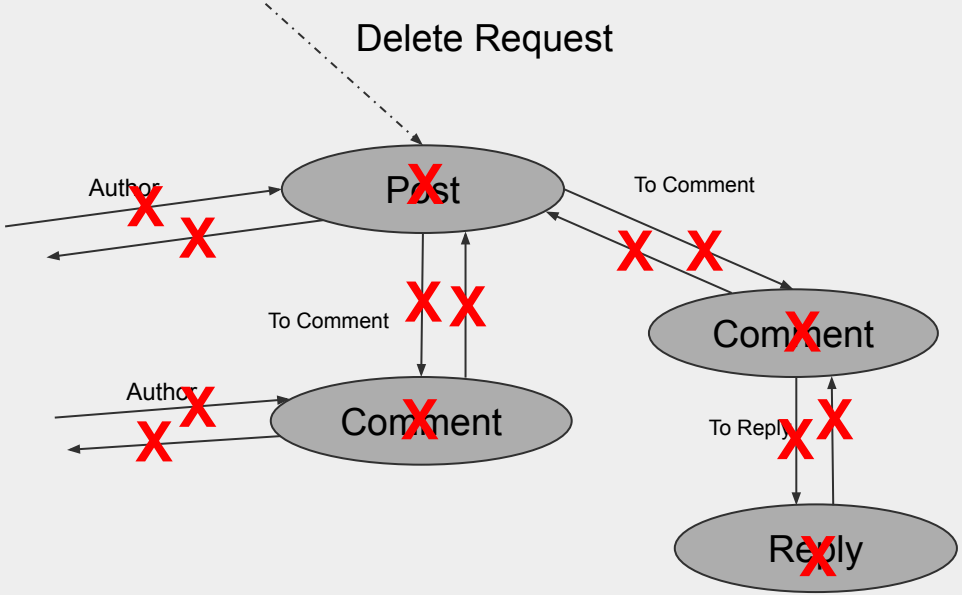
Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs

Restoration Logs:
Batched Restoration Record, other batched record

Persistent Stack:

- Post
- Comment



Actions:

1. Read
2. Checkpoint on stack
3. Log restoration logs
4. Delete

Restoration Logs:
Batched Restoration Record, other batched record

Persistent Stack:

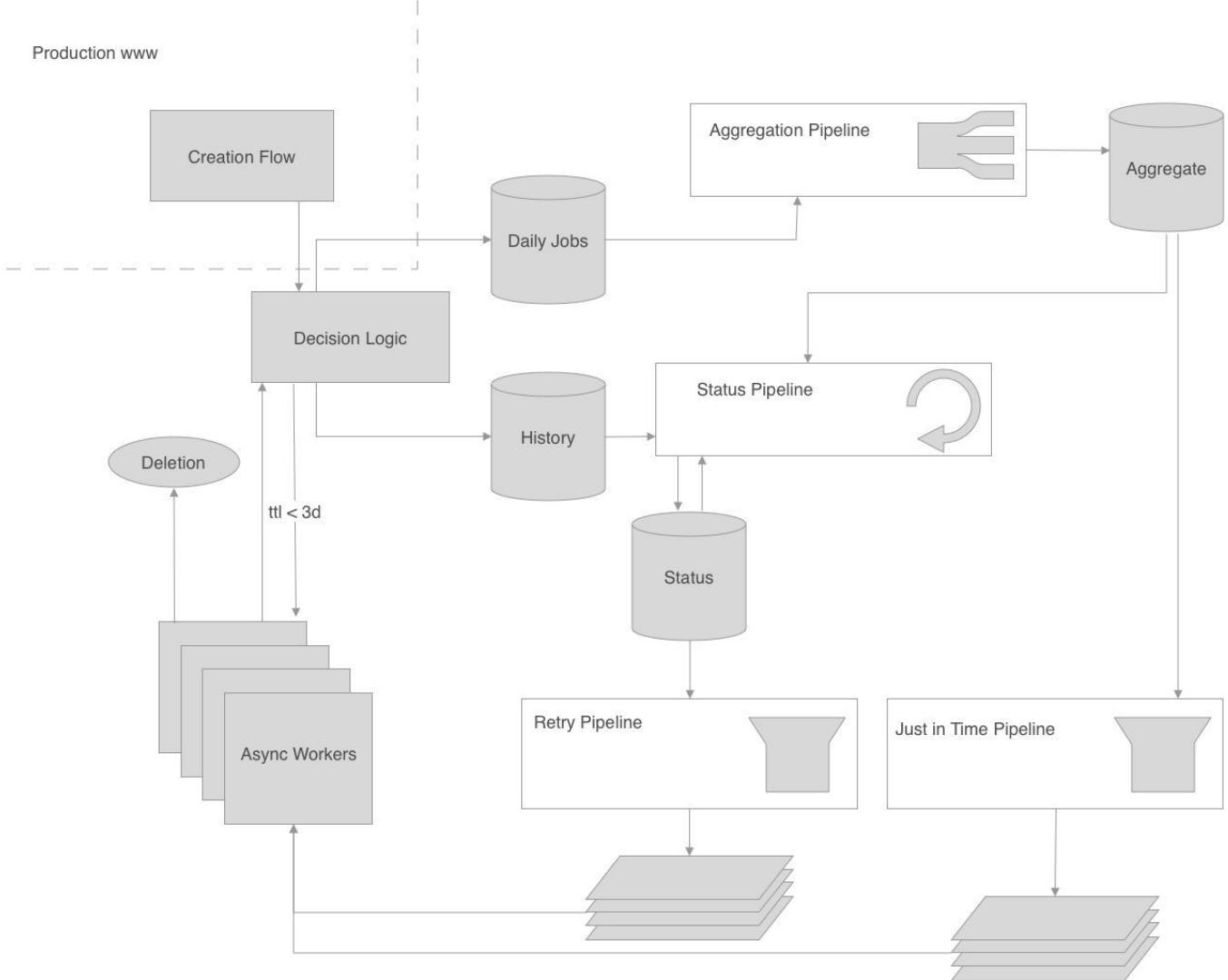
- Post
- Comment

Outline

I. Design Overview
   A. Relying on a Data Definition Language
   B. Walking the graph
   C. Batching
   D. Scheduling deletions in the future
II. Guarantees
III. Monitoring Guarantees
IV. Conclusion

Scheduling Deletions in the Future

- Essential to enable ephemerality at Facebook
  - Like stories, or the account deletion grace period
- Supports scheduling years in the future
- Supports custom TTL logic
  - "Delete this post 9 days after the last comment"
- 160B events processed per day

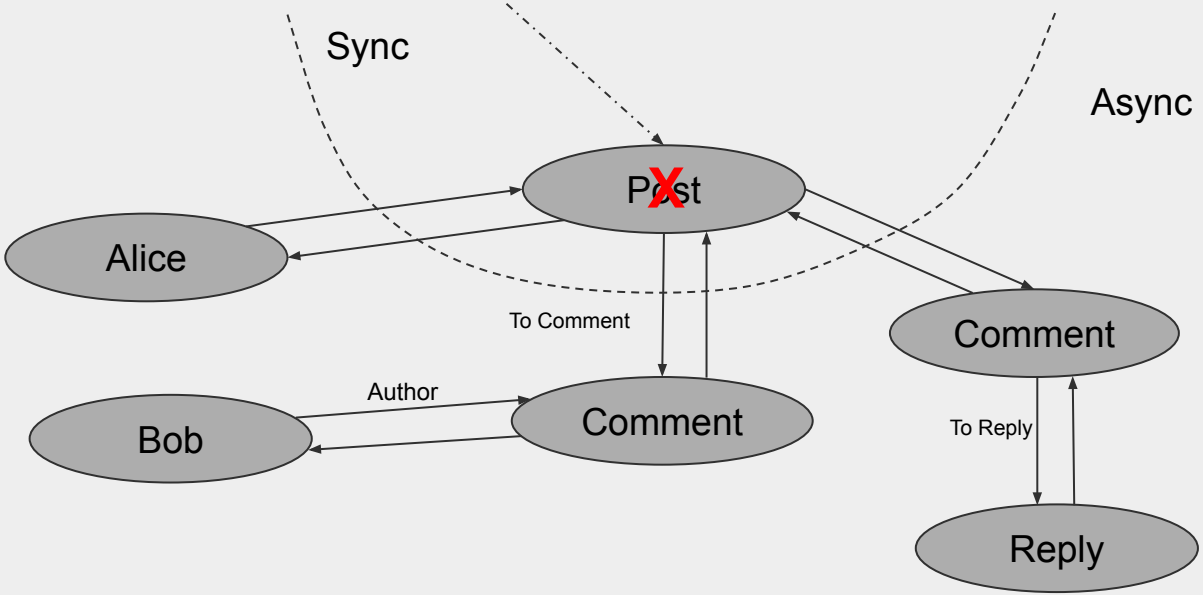

Production www
Creation Flow
Aggregation Pipeline
Aggregate
Daily Jobs
Decision Logic
Status Pipeline
History
Deletion
ttl < 3d
Status
Async Workers
Retry Pipeline
Just in Time Pipeline

Outline

I. Design Overview
II. Guarantees
    A. Scheduling
    B. Eventual Completion
    C. Eventual Completeness
    D. Restorations
III. Monitoring Guarantees
IV. Conclusion

Scheduling

- Deletions run in two phases
  - The "Sync part", running synchronously in the web request
  - The "Async part", which iterates over the graph

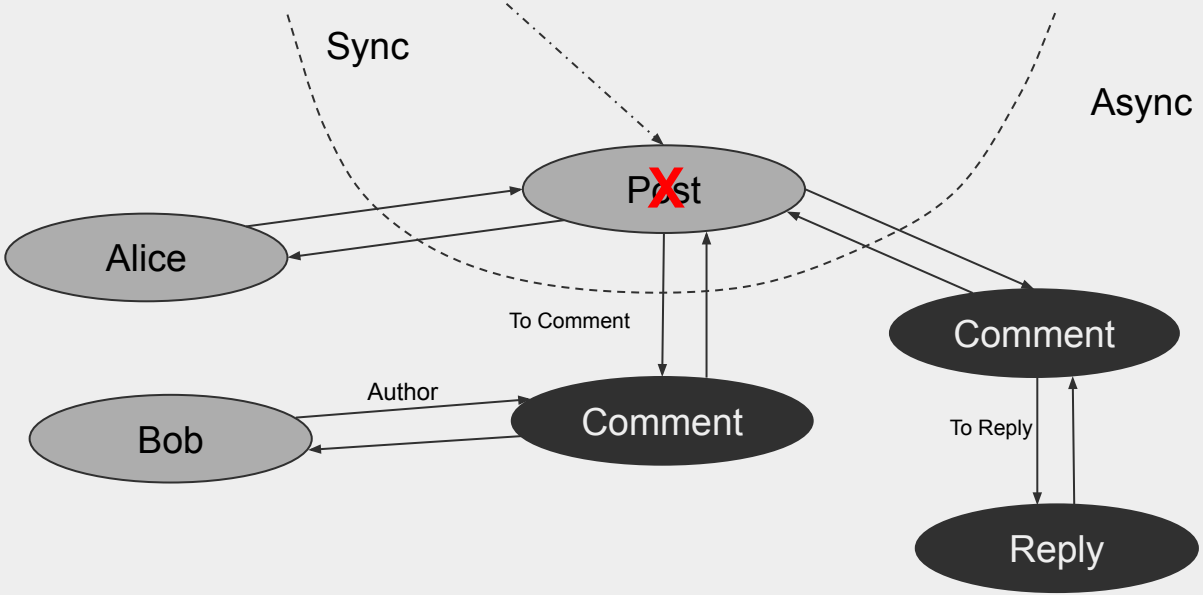


Steps:

1. Write the deletion metadata
2. Log the deletion request
3. Delete the top level object
4. Schedule the async job

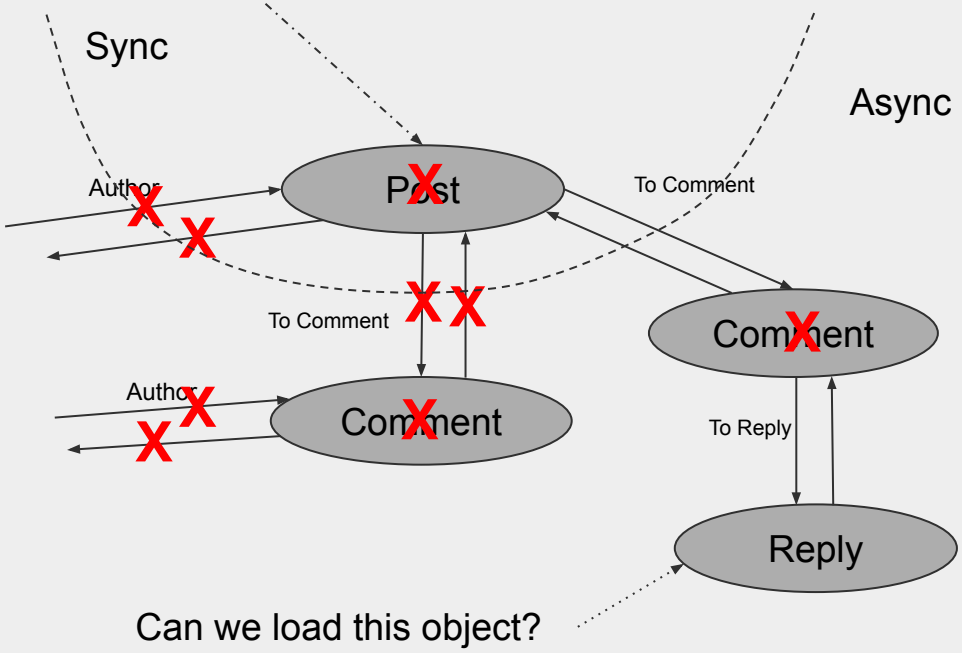
Scheduling

- Deletions run in two phases
  - The "Sync part", running synchronously in the web request
  - The "Async part", which iterates over the graph
- The sync part effectively hides the sub-graph


Sync
Async
Post
Alice
To Comment
Comment
Author
Bob
Comment
To Reply
Reply

Scheduling

- Deletions run in two phases
  - The "Sync part", running synchronously in the web request
  - The "Async part", which iterates over the graph
- The sync part effectively hides the sub-graph
  - The privacy layer is distinct from the deletion graph, so we have checks in place



Actions:

1. Read
2. Checkpoint on stack

Persistent Stack:

- Post
- Comment

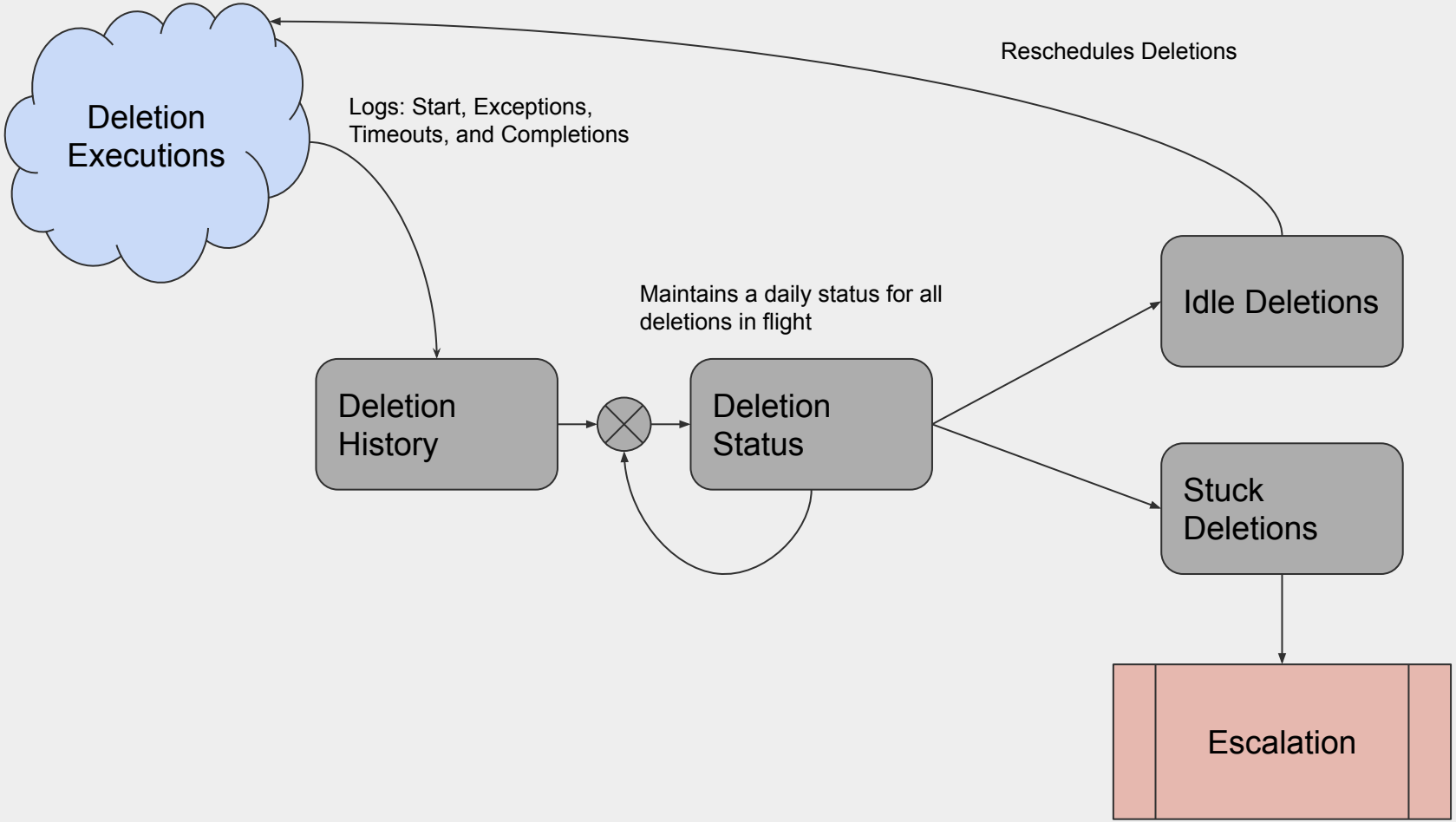
Restoration Logs:
Batched Restoration Record

Outline

I. Design Overview
II. Guarantees
    A. Scheduling
    B. Eventual Completion
    C. Eventual Completeness
    D. Restorations
III. Monitoring Guarantees
IV. Conclusion

Eventual Completion

- Deletions encounter infrastructure issues
- Deletions encounter bugs
- Deletions get dropped by dependencies
- Deletions fail halfway through their scheduling
- Every deletion started must complete

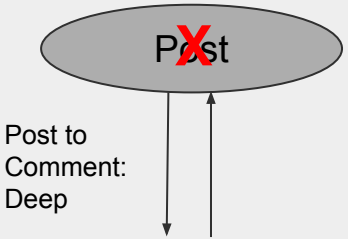

Deletion Executions
Logs: Start, Exceptions, Timeouts, and Completions
Reschedules Deletions
Maintains a daily status for all deletions in flight
Deletion History
Deletion Status
Idle Deletions
Stuck Deletions
Escalation

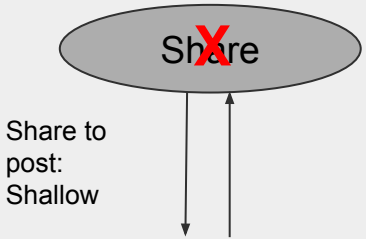
Outline

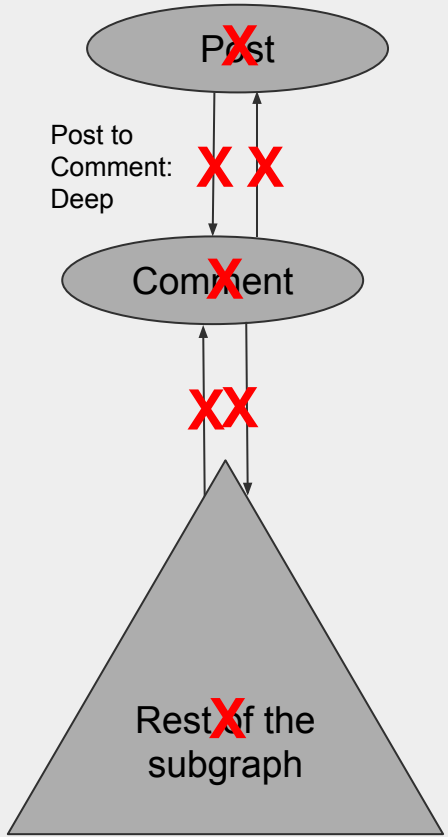
I. Design Overview
II. Guarantees
    A. Scheduling
    B. Eventual Completion
    C. Eventual Completeness
    D. Restorations
III. Monitoring Guarantees
IV. Conclusion

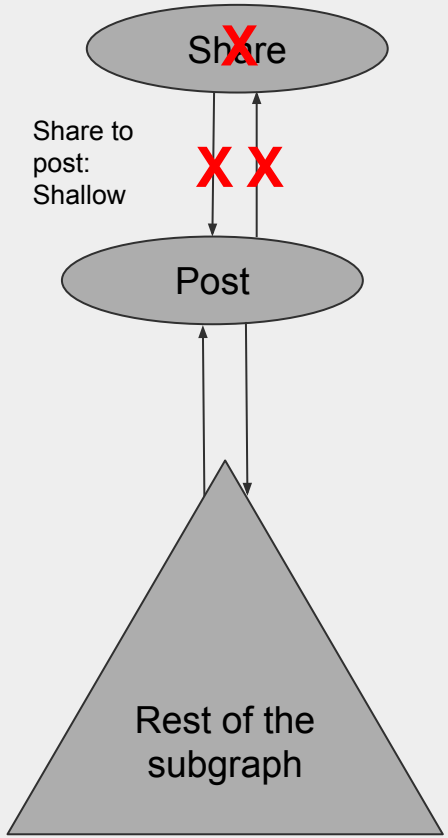
Eventual Completeness

- Issues can lead to orphaned data
  - Bugs in the deletion logic
  - Race conditions
  - Deletion settings mis-configurations
- We need to cleanup such data retroactively
- The solution: **Object re-deletions**


Post
Post to
Comment:
Deep


Share
Share to
post:
Shallow


Post
Post to
Comment:
Deep
Comment
Rest of the
subgraph


Share
Share to
post:
Shallow
Post
Rest of the
subgraph

Eventual Completeness

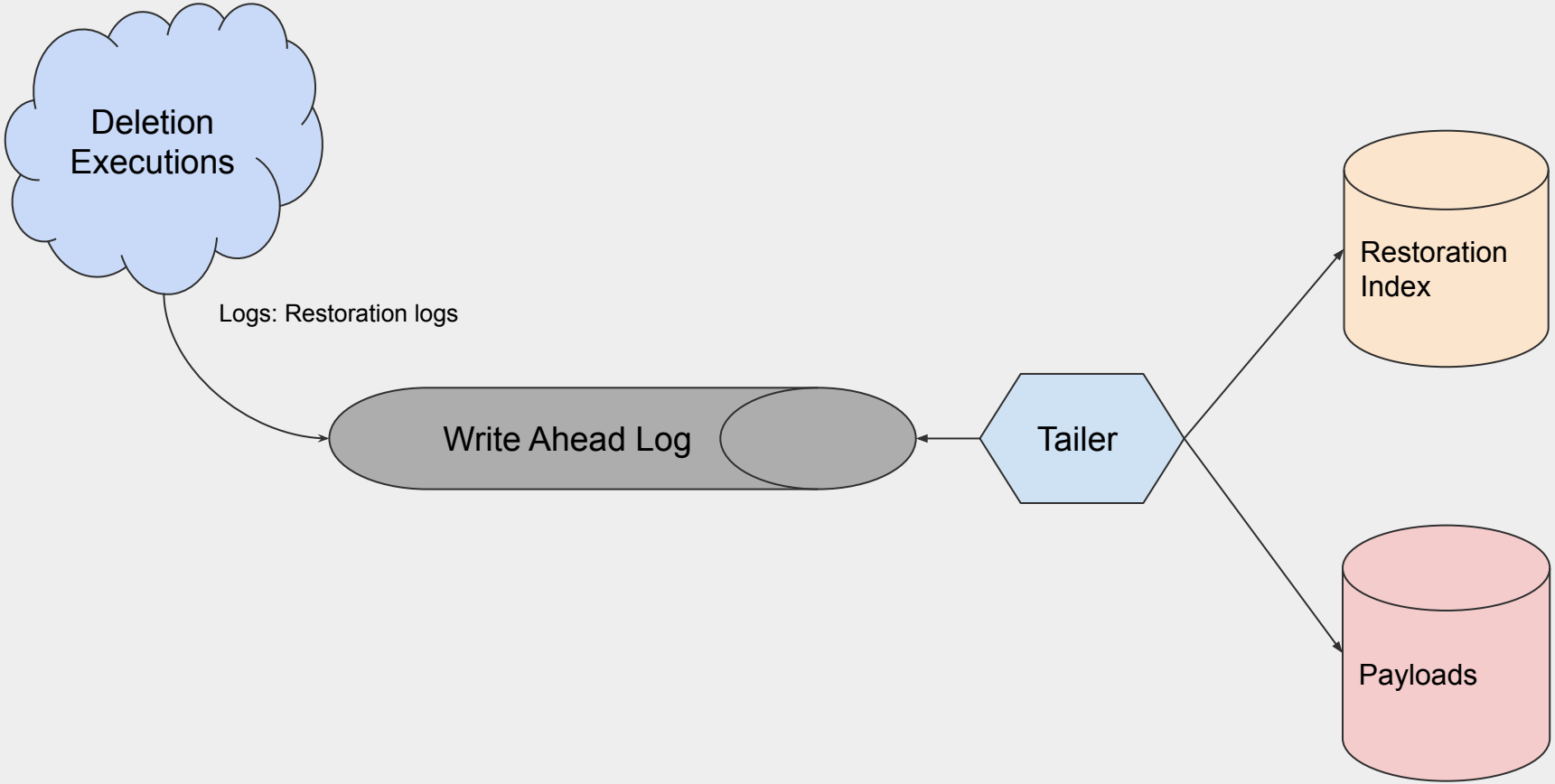
- We run migrations on object types
- Migrations start every other week
- They iterate over all our data, and schedule the deletions

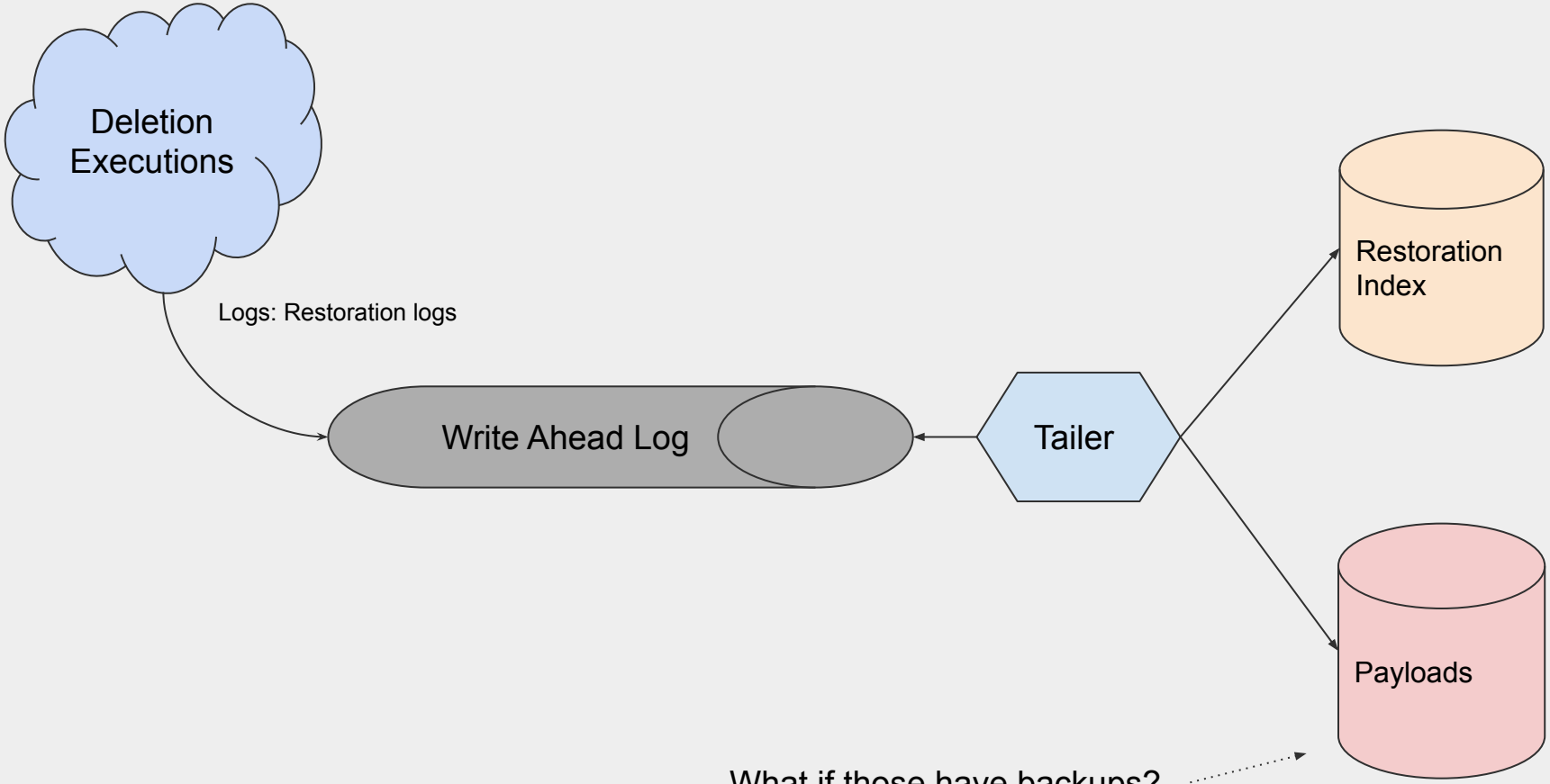
Outline

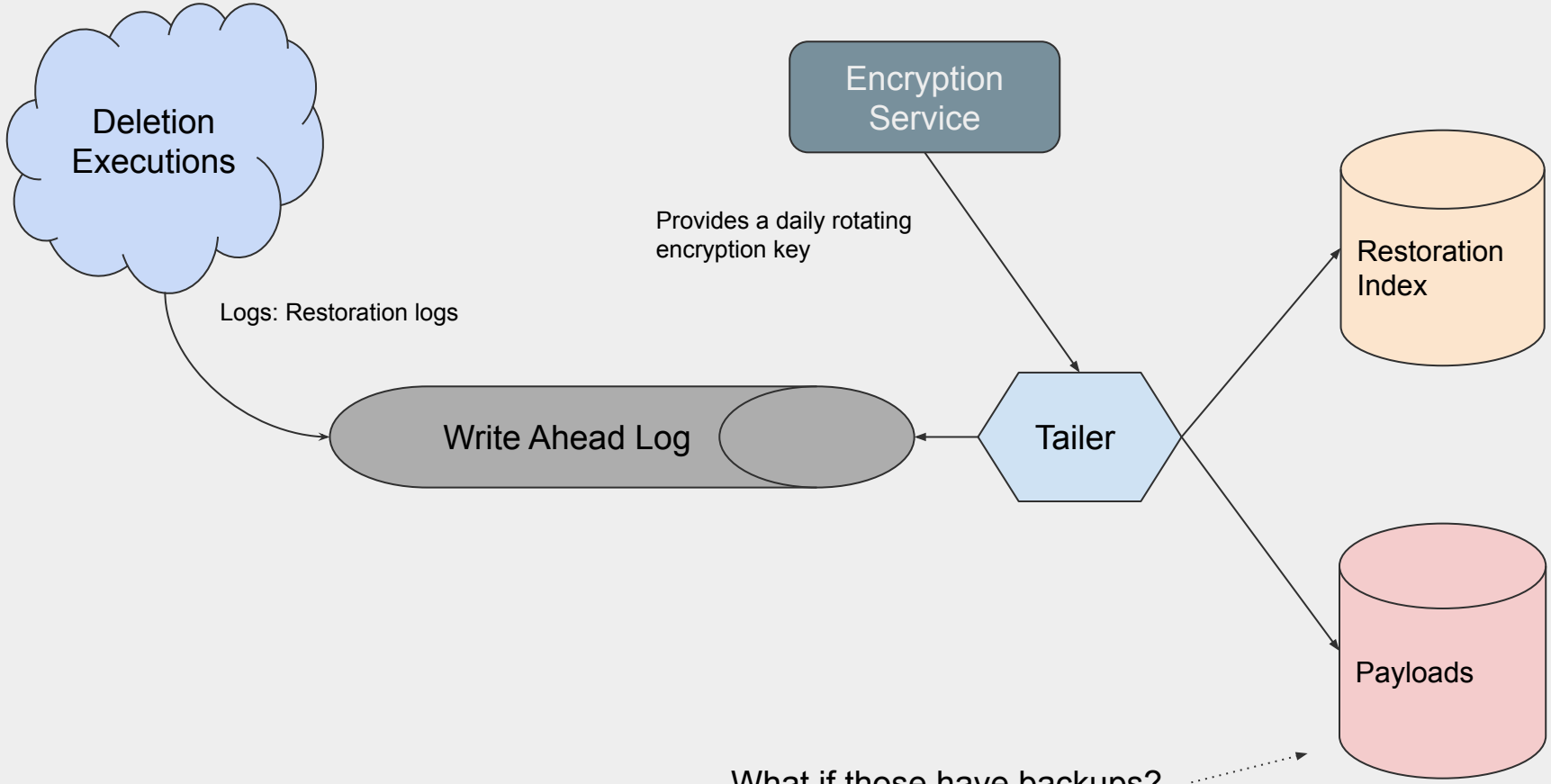
I. Design Overview
II. Guarantees
    A. Scheduling
    B. Eventual Completion
    C. Eventual Completeness
    D. Restorations
III. Monitoring Guarantees
IV. Conclusion

Restorations

- Data losses are inevitable
  - Product bugs
  - Deletion misconfiguration
- We log restoration logs before any delete we issue
- Graph indexed, different from the data stores backups


Deletion Executions
Logs: Restoration logs
Write Ahead Log
Tailer
Restoration Index
Payloads


Deletion Executions
Logs: Restoration logs
Write Ahead Log
Tailer
Restoration Index
Payloads
What if those have backups?


Deletion Executions
Encryption Service
Provides a daily rotating encryption key
Restoration Index
Logs: Restoration logs
Write Ahead Log
Tailer
Payloads
What if those have backups?

Preventing data loss

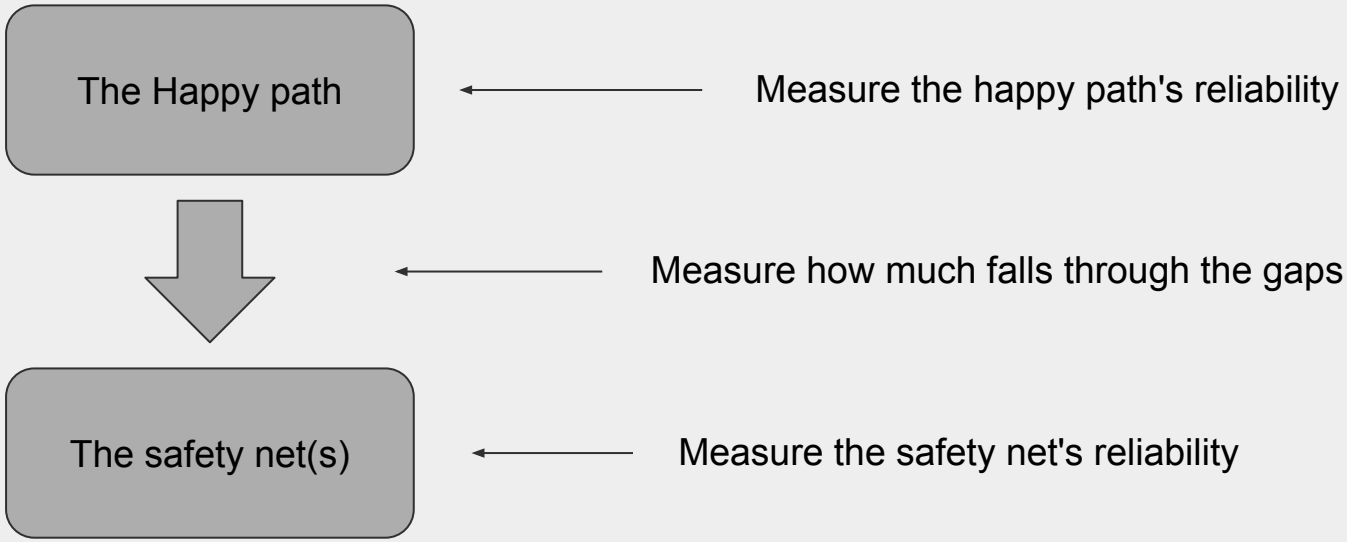
- Static analysis on the deletion graph
- Dynamic checks based on Deletion Constraints
- Predictions on the edges' deletion behaviours

Outline

I. Design Overview
II. Guarantees
III. Monitoring Guarantees
IV. Conclusion

Monitoring Guarantees

- Scheduling
  - Success rate
  - Amount of object detected as still visible during a deletion
- Eventual Completion
  - Amount of deletions getting rescheduled
  - Amount of deletions not getting executed for more than a day
- Eventual Completeness
  - Amount of object remediations
- Restorations
  - Oldest encryption key stored



Bonus points:

- An orthogonal way to measure success
- A second safety net

Conclusion

Conclusion

- Deletion is a hard problem
- The happy path isn't enough
- We need to make it easier for developers to do the right thing than the wrong things

Thanks!


Vivek Kumar


Nina Kalinina


Jyothi Prabhakaran


Leonid Chashnikov


Marius Feteanu


Ariel Zylber


Nikunj Rudani


Alessio Piergiacomi


Tomáš Chvátal


Riccardo Govoni

Ernest Sadykov


Tom Taylor


Julian Hatchwell


Ria Garg


Eda Gür


Mohamed Hashi


Sneha Padgalwar


Amitsing Rajendrasing Chandele


Benoît Reitz


Gustavo Pacianotto Gouveia


Divino Neto


Anurag Sharma


Ohad Almagor


Jordan Webster


Leonid Chashnikov


Mahdy Nasr


Nikunj Rudani


Prakash Verma


Shubhanshu Agrawal


Somya Kumar


Akin Ilerle


Eda Gür


Alessio Piergiacomi

Ernest Sadykov


Jyothi Prabhakaran


Nikita Efanov


Tudor Tiplea


Shradha Budhiraja


Yiannis Papagiannis

nebryum@fb.com