DELF

"Deletion Framework", by Facebook

presented by Jonathan Weisskoff

Objectives:

(a) **enforce** how user data should be deleted before any data is collected,

(b) **validate** specifications to surface mistakes early

(c) when undetected mistakes occur, **recover** inadvertently deleted data

Question: Are there other frameworks whose central goal is to prevent developers from making mistakes?

Timeline

1. DELF replaced the code mandating deletes to data stores
    a. Used a procedural API
    b. Created logs for restoration
2. Introduced dynamic validation techniques
3. Declarative API based on object and edge type annotations
4. Static validation techniques

FB infrastructure

DELF annotations are added to DDLs that defines data types. (Ex. Thrift)

- DDL is compiled into other languages as in stubs (Eg. Java).
- Instances of objects are made in the other languages in the application code.
- Those objects are put intp various data stores (ex. TAO, Everstore, MySQL, and ZippyDB)

Benefit: annotations are at the type level rather than at the object level, so easier to keep track of.

Phase 3: Example of annotations

```
1   object_type:
2     name: photo
3     storage:
4       type: TAO
5       deletion: directly
6     id:
7       photo_id: integer_autoincr
8     attributes:
9       created_on: datetime
10      caption: string
11  edge_types:
12    handle:
13      to: photo_blob
14      deletion: deep
15    created_by:
16      to: user
17      deletion: shallow
18      inverse:
19        created_photo:
20          deletion: deep
```

Question

Alice likes Bob and Bob's profile pic is now in Alice's friend's list.

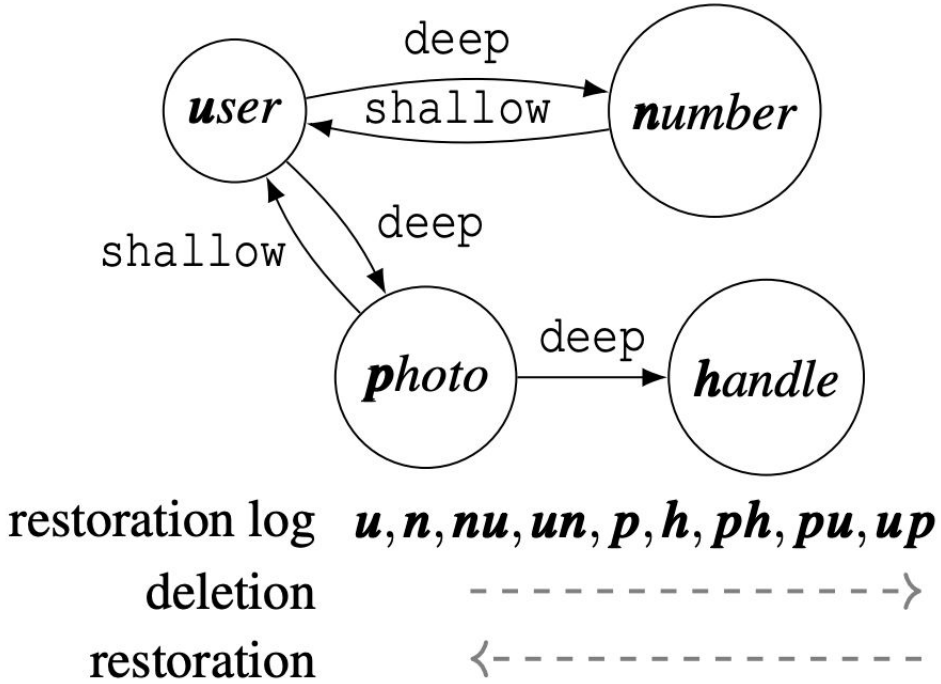
What are the annotations of the edges and the objects?

Alice -------- Bob ---------- profile pic

Phase 1: Restoration Logging

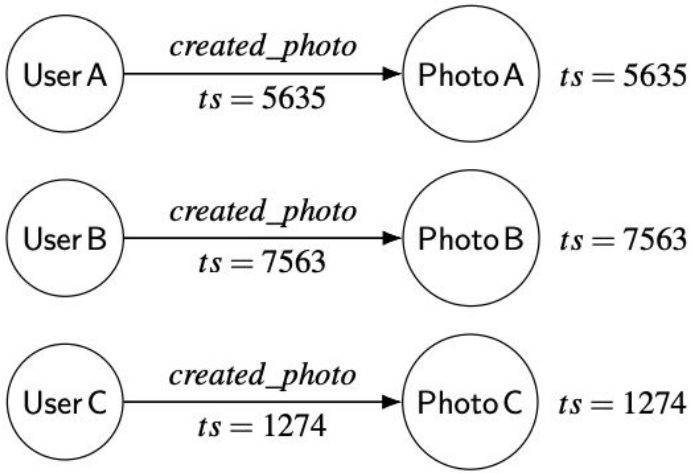
- objects: pre-order
- edges: post-order
- restoration: backwards

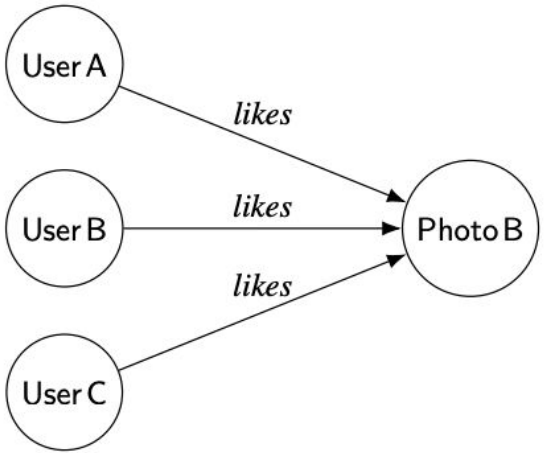
Why?



Phase 2: Dynamic Validation techniques

One of 3 methods: Heuristics





Phase 4: Static validation

(a) DELF rejects any data types found to lack annotations, and

(b) DELF performs a reachability analysis starting from every object type annotated with directly, directly_only, short_ttl, and not_deleted visiting all their edge types annotated deep.

The analysis must reach all defined object types in the system