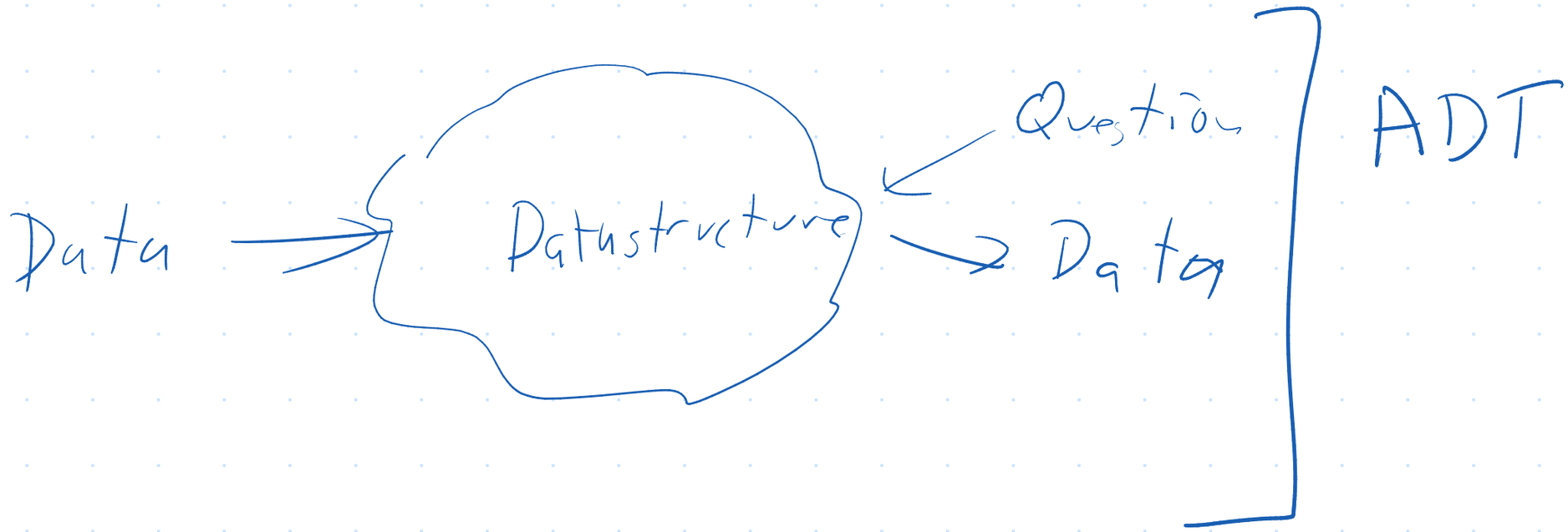
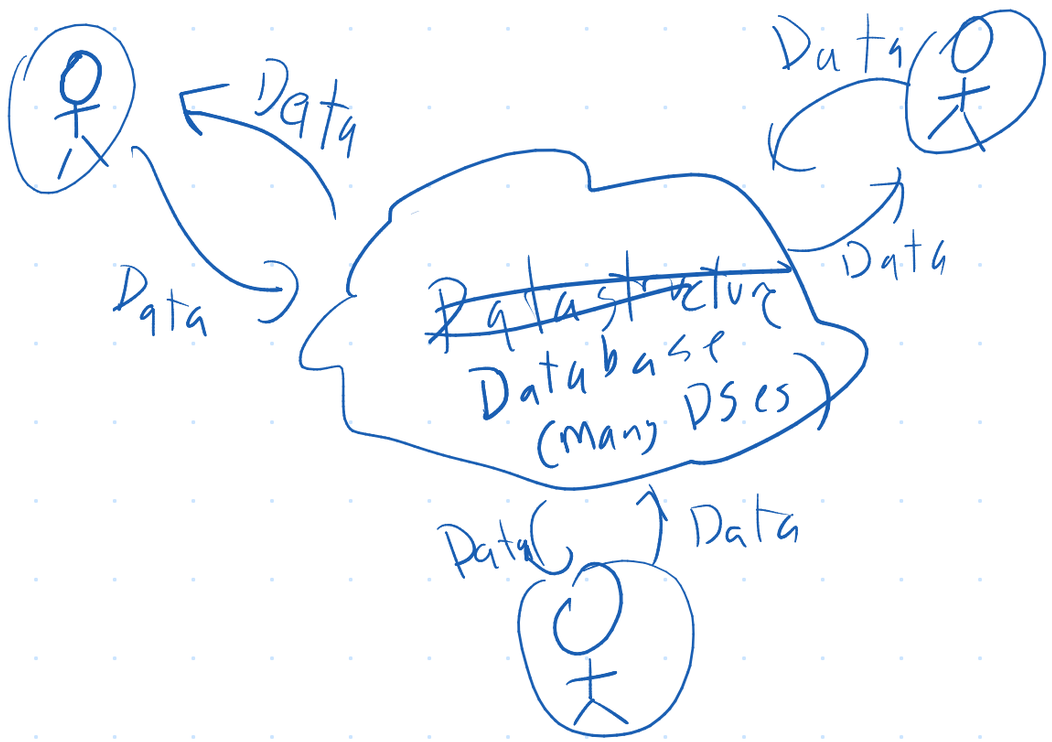
CSE 350

Advanced Data Structures

Topic 15: Correctness

Perspectives on a Database





How do we define "Correct" behavior?

- We want
  - rules for how each users actions affect other users
  - rules defined in terms of atomic operations on the underlying data structures
  - * rules for the db to guard access to the data
  - rules that ensure that a user's actions are predictable
  - rules for how and when we can say that the data is "safe"
  - rules for how and when the DB fails safely
  - rules for ensuring that the data always "makes sense" (according to the admin)

What could go wrong?

Desirable Properties

Predictability

Atomicity → Rules for error handling/failure

Consistency → Rules for data making sense

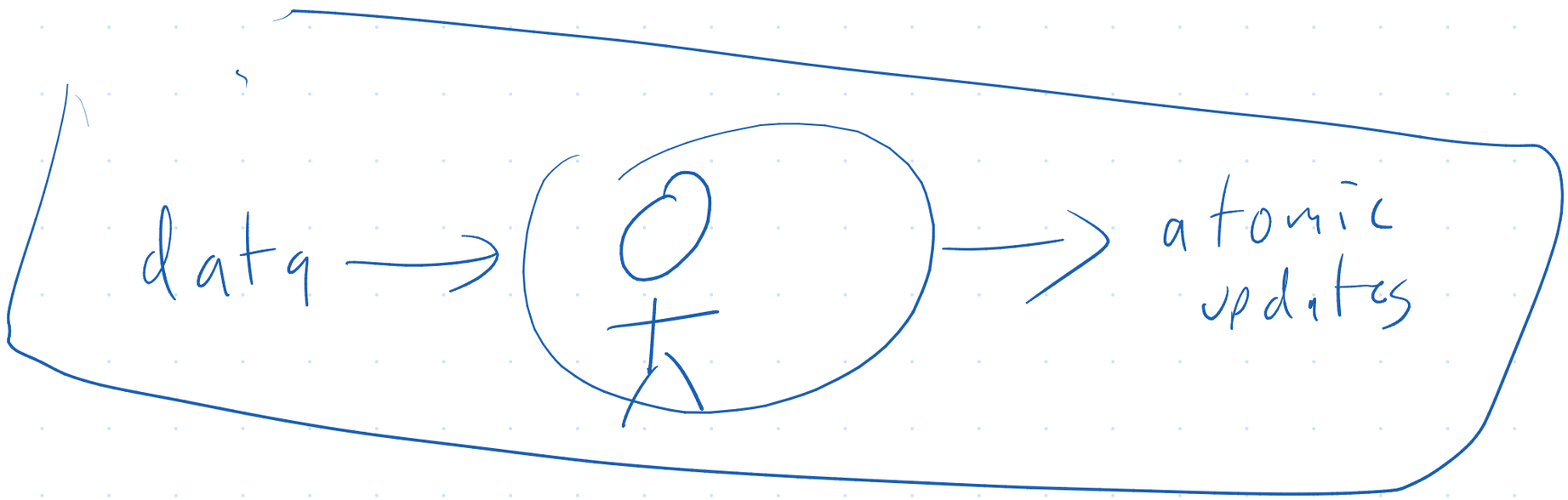
Isolation → Rules for state interaction

Durability → Rules for data being safe

DB read Permits

DB append 10 to Permit_summary

DB add 10 to tracking



Transaction
- ↳ Reads
- ↳ Writes

Atomicity

What happens on a failure?

Transaction fully applied "COMMIT"

No trace at all "ABORT"

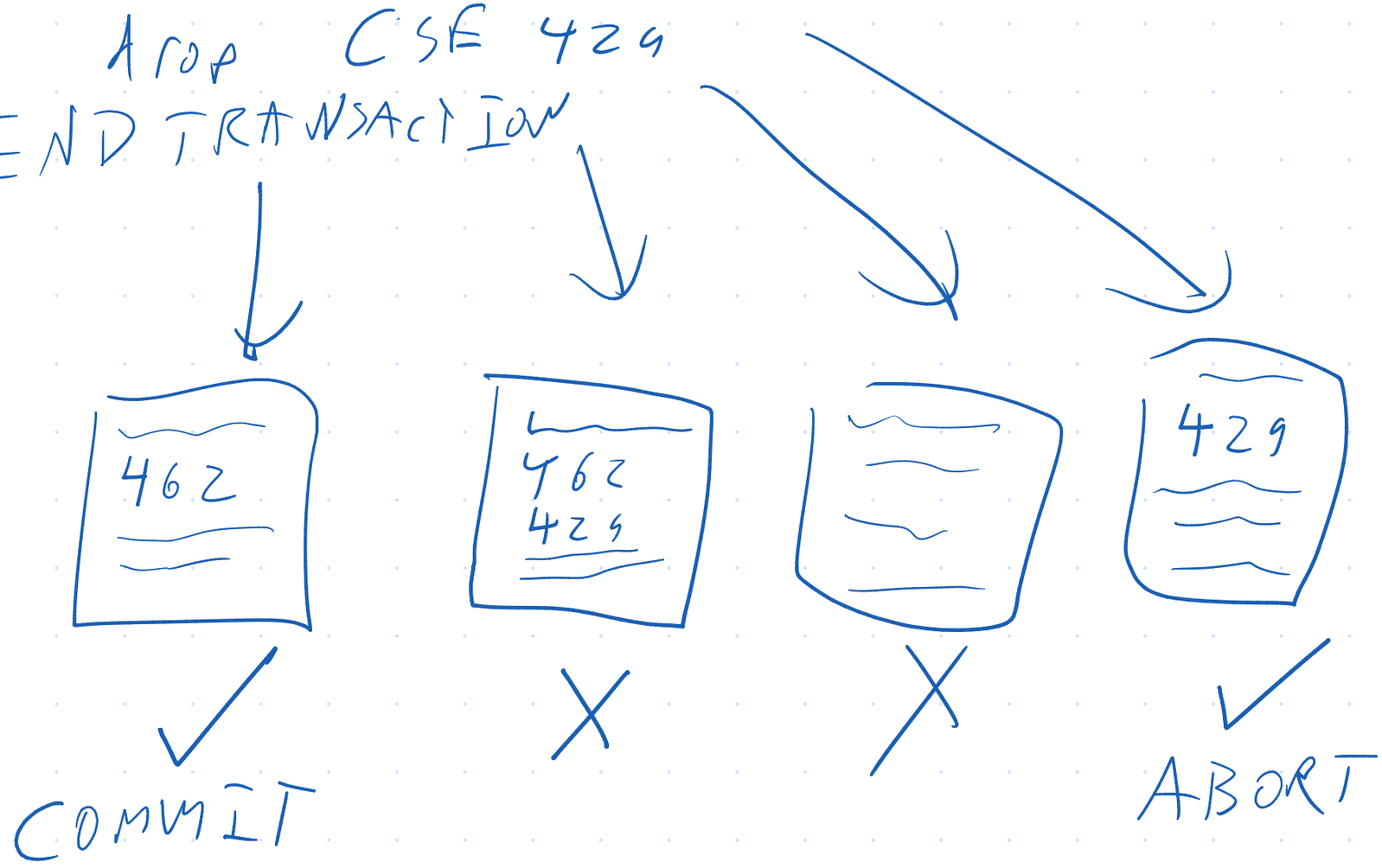
Every transaction commits or aborts
↳ if it commits its effects are fully visible
↳ if it aborts it has no effects

BEGIN TRANSACTION

add CSE 462

drop CSE 429

END TRANSACTION



Consistency

Administrator can define correctness rules

↳ Each transaction must leave the DB in a state that satisfies these rules

Enrolled (
student_id
class_id
)

valid reference

student (
id
name
enrolled
school
)

unique

defined?

class_id shouldn't appear more times than capacity

Constraint

Domain Constraint

↳ e.g. enrollment_date can not be NULL

e.g. school ∈ { SEAS, CAS, Med, ... }

Key Constraint

↳ value is unique

Foreign Key Constraint

↳ Attribute(s) references a key in another table & the reference <u>must</u> be defined

↳ Delete referenced value

↳ Insert value w/ nonexistent reference

↳ Update referenced (or referencing)

ON DELETE, ON INSERT, ON UPDATE

- ↳ set referencing value to NULL
- ↳ flag it as an ERROR
- ↳ Fix it "CASCADE"
  - ↳ DELETE -> Delete references
  - ↳ INSERT -> Create referenced record
  - ↳ UPDATE -> Update references

Table Constraint

↳ Define a query → Returns a (specific) result
→ Returns no result

"Assert"

e.g.
```
(SELECT class_id, count(*)
FROM Enrolled, class
WHERE Enrolled.class_id = class.id
HAVING COUNT(*) > class.capacity
GROUP BY class_id, class.capacity) is empty
```

Isolation

Every transaction runs "as if" it were the only transaction running

db: D

$f(D)$ — db after running f

$g(D)$ — after g

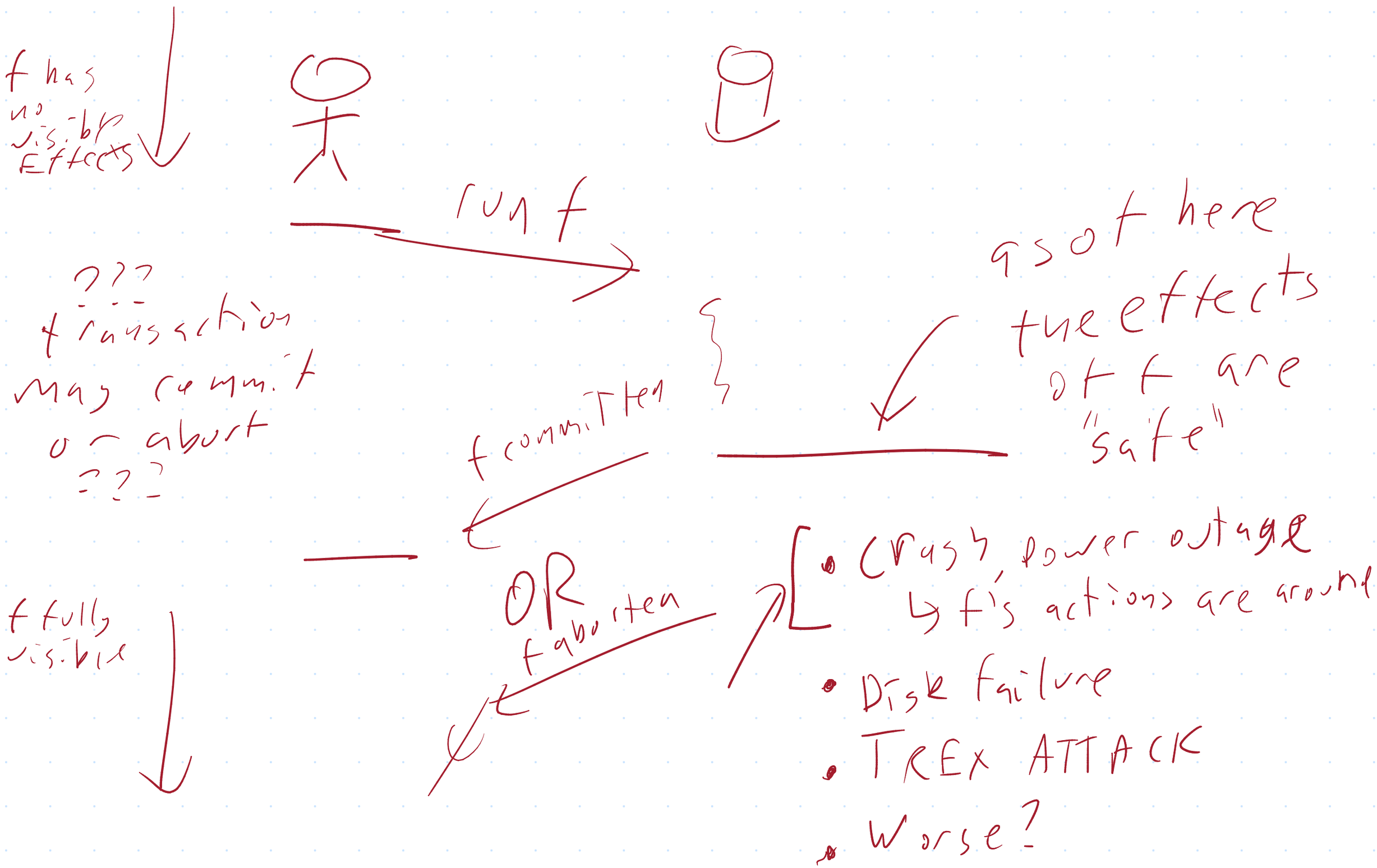
f and g together leads to

$f(g(D))$

$g(f(D))$

either f comes before g
or g comes before f ← "Serializable"

Durability


f has no visible effects
run f
??? f transaction may commit or abort ???
f committed
OR
f aborted
f fully visible
as of here the effects of f are "safe"
• crash, power outage
↳ f's actions are around
• Disk failure
• TREX ATTACK
• Worse?