



## String

- Length
- Substring

## Numeric

- Round
- Truncate

## Date time

- Date diff
- YEAR | MONTH

## Miscellaneous

- IFNULL

# Agenda

- Misc. functions
  - Transactions
    - ACID properties
    - Isolation levels
    - Demo
  - Indexing
- 

Misc. functions

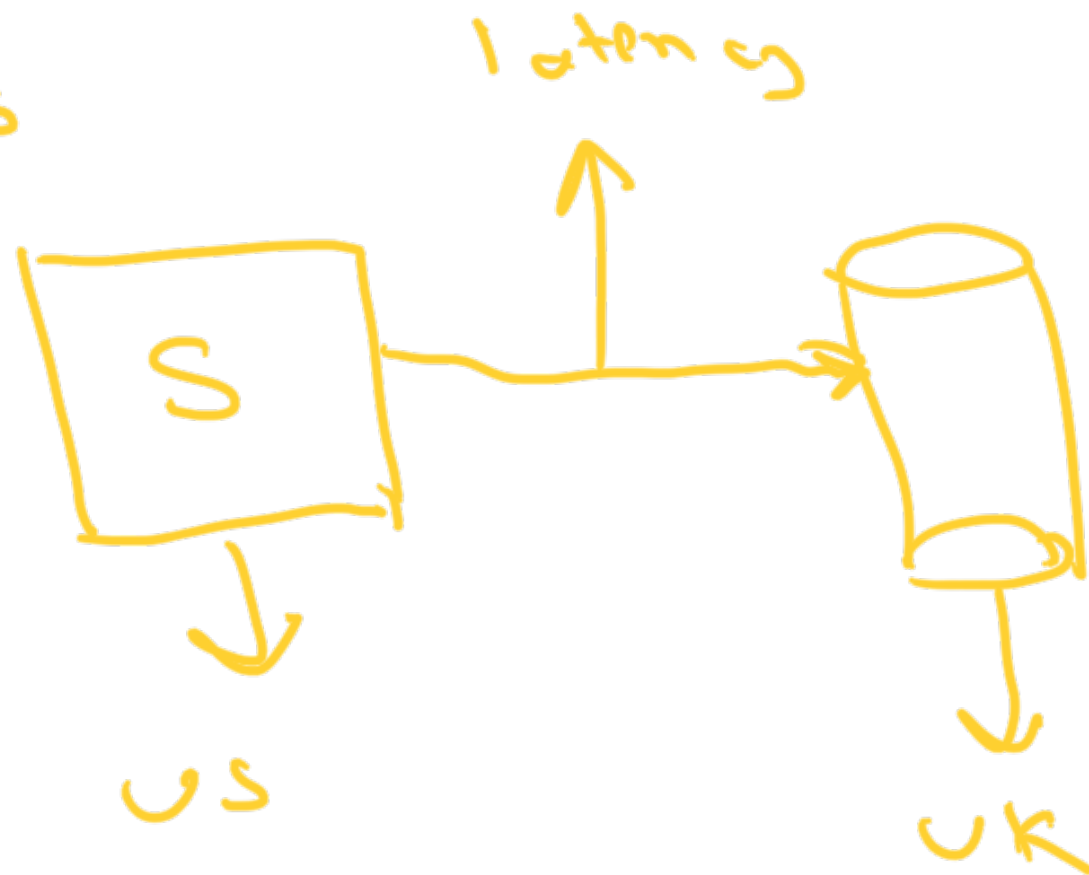
Querying Data

→ Flag - 1 or 0  
- Active or Inactive

Flag → Active / Inactive  
1 / 0

200 ms → user does not wait

2 s - 1.2 s



✓ null checks

- data mapping

- 0/1 - Active / Inactive

My SQL

① IFNULL / ISNULL

→ check if null

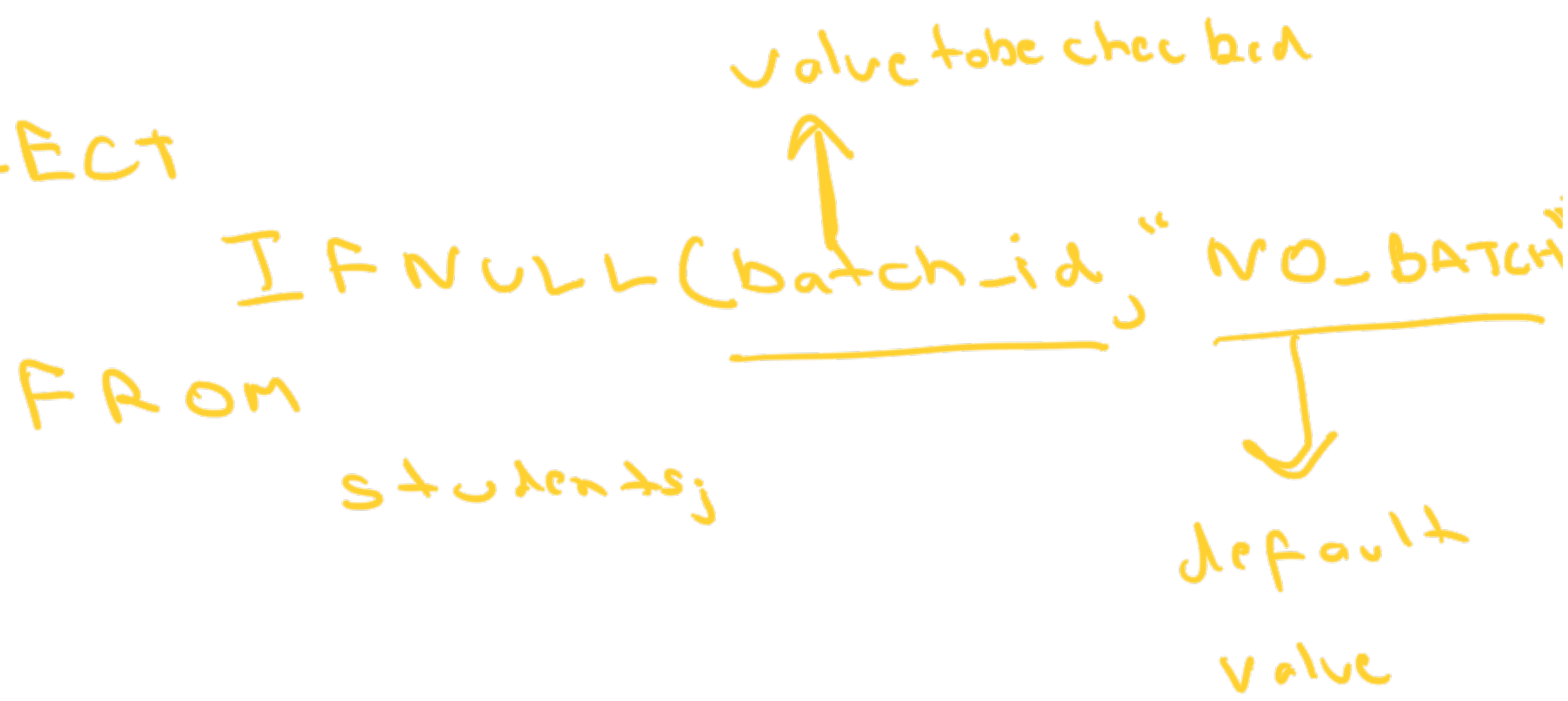
→ If not, return value

→ If null, default



①

SELECT



Down side

→ Business logic / values  
in DB code

---

Select user identifier

- id
- email
- phone?



^ ^

-

-

-

Ifnull ( If null ( phone , email ) )

first name → last name → No NAME

② COALESCE

→ if null recursively

COALESCE ( phone , email , id )





Recursively replace null values

---

Check if a user has a batch

→ YES / NO

batch\_id

if batch\_id → YES

→ NO

SELECT

batch\_id

FROM

USER

↓ condition

① IF ( batch\_id IS NOT NULL,  
YES, ← TRUE  
NO) ← FALSE

ternary operator

batch_id	1	→	<u>ONE</u>
	2	→	<u>TWO</u>
	3	→	THREE
	74	→	<u>BATCH</u>

④

CASE

CASE

WHEN

condition

THEN

ELSE

END

CASE

WHEN batch\_id = 1

THEN "ONE"

WHEN batch\_id = 2

THEN "Two"

ELSE

'batch'

END

5:53 - 5:58

- 10:28

- 
- Transactions
  - Indexes
  - Sub queries & Views
  - Query optimisation, stored procedures
    - ↳ window functions
    - ↳ triggers (cursors)

---

# TRANSACTIONS

- unit of work
- interchange

unit of work

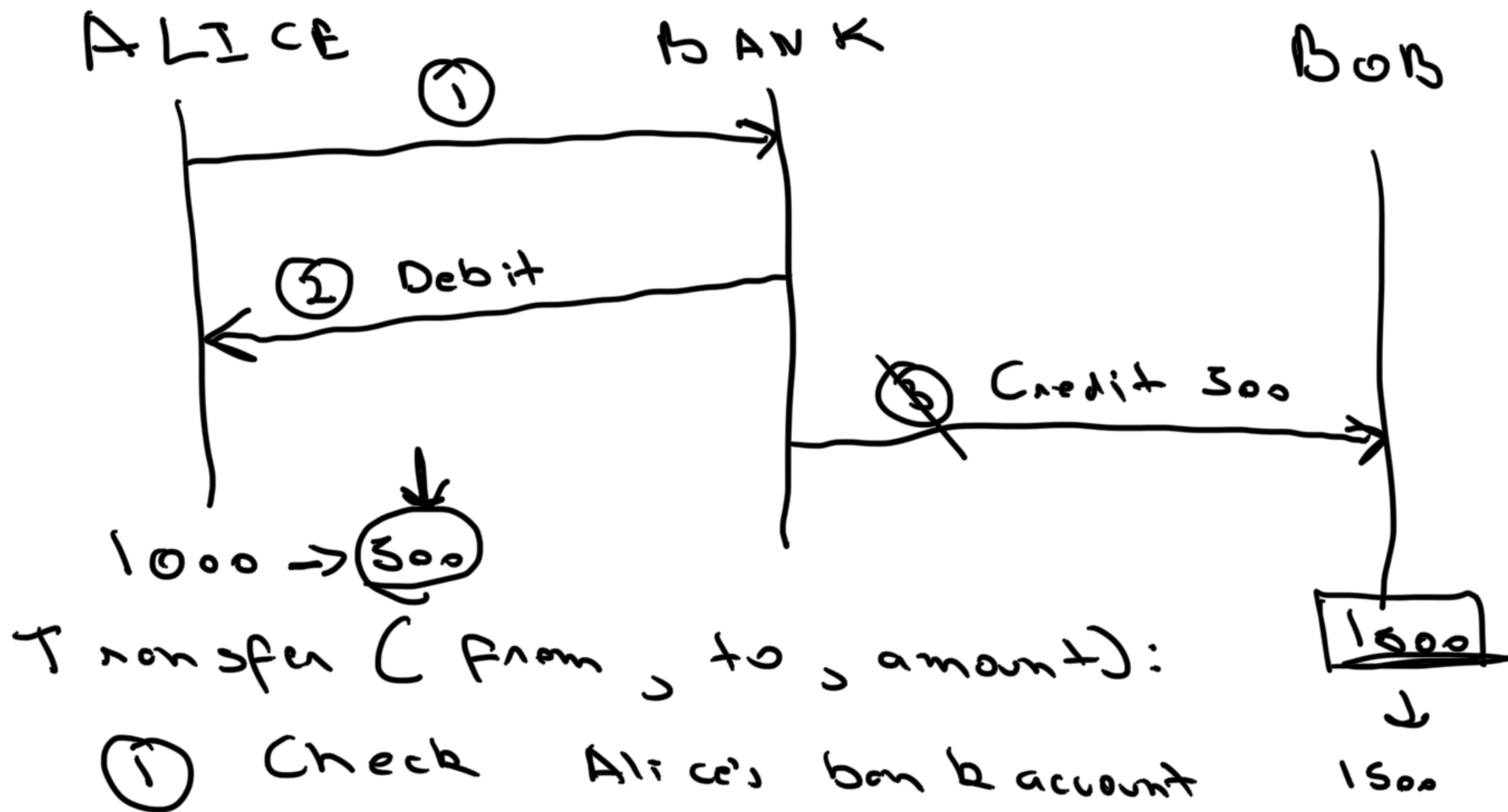
against a database

Transaction

- Set of related DB operations

Banks

---



② De duct 500 from Alice

③ Credit Bob with 300

---

① Alice's acc. was debited  
but credit failed

↓  
inconsistent state


② Both operations succeeded

Consumption → 1000 - 1000  
500 - 1500

- inconsistency

Transaction - ACID

A	-	Atomicity
C	-	Consistency
I	-	Isolation
D	-	Durability



---

Atomicity → a single unit

→ relational model



→ atomic

→ single value

ALICE - 500 debited

Bob - credit 500 failure

- refund - nothing

- retry - all

- All or nothing

- Try an section

- All your operations are complete
- Else, no operation will be done

## Consistency

ALICE	1000	300
BOB	1000	1500
t=0	2000	2000

ALICE	1000	300
BOB	1000	1000
	<hr/> 2000	<hr/> 1500

→ State of the DB should be same

→ data types

→ State is valid

→ data integrity

FK should be valid

300

750

-250

---

Isolation

↳ independent transactions

---

Multiple transactions executing

together, we can have inconsistent state

1000 ALICE  $\rightarrow$  DEBIT 500

500

1000

CALCULATE = 500 + 1000  
= 1500

BOB  $\rightarrow$  CREDIT 500

1500

2000

~~Concurrent transactions~~

Concurrent transactions

Isolation

- concurrent transactions  
are supposed to be  
independent of each other.

Isolation levels

Durability

→ physical bandwidth

→ if operations are successful

→ data should be  
permanently stored  
in disk

→ backups

Atomicity - all or nothing

Consistency - state of system should

Isolation — <sup>be some</sup> concurrent transactions  
Durability — should be independent  
if all operation succeed,  
then data should be  
permanently stored

---

Atomicity

— all or nothing

①

get Alice's balance

②

check Alice's balance

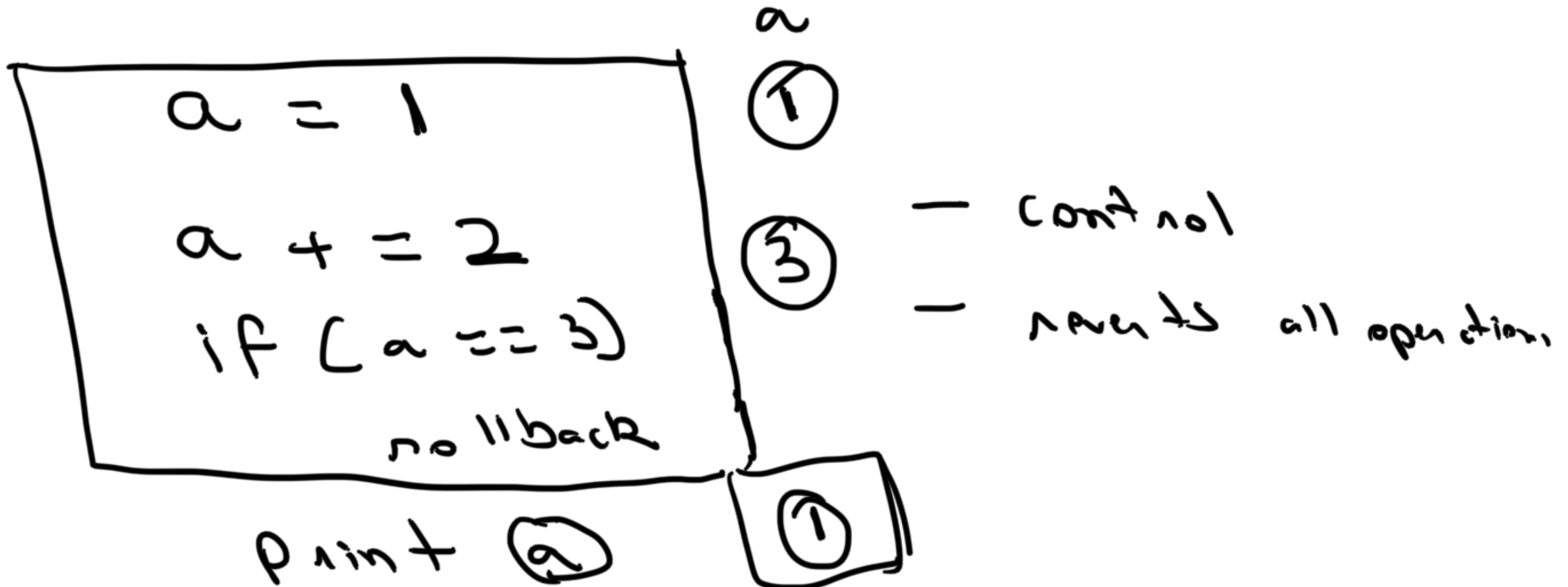


- noise exception

# ROLLBACK

- Exception

- Revert all operations



ALL OR NOTHING  
↓  
ROLLBACK

① ROLLBACK

② COMMIT - ALL

① START TRANSACTION

→ Data integrity removed

ALICE := READ()

IF ALICE < 500:

## ② ROLLBACK

ALICE := ALICE - 500

WRITE (ALICE)

BOB := BOB + 500

WRITE (BOB)

## ③ COMMIT;

git commit

↳ DI check on back

- NULL Replacement

- IFNULL, COALESCE  
↑                   ↑  
single           recursive

- Data mapping

- IF, CASE

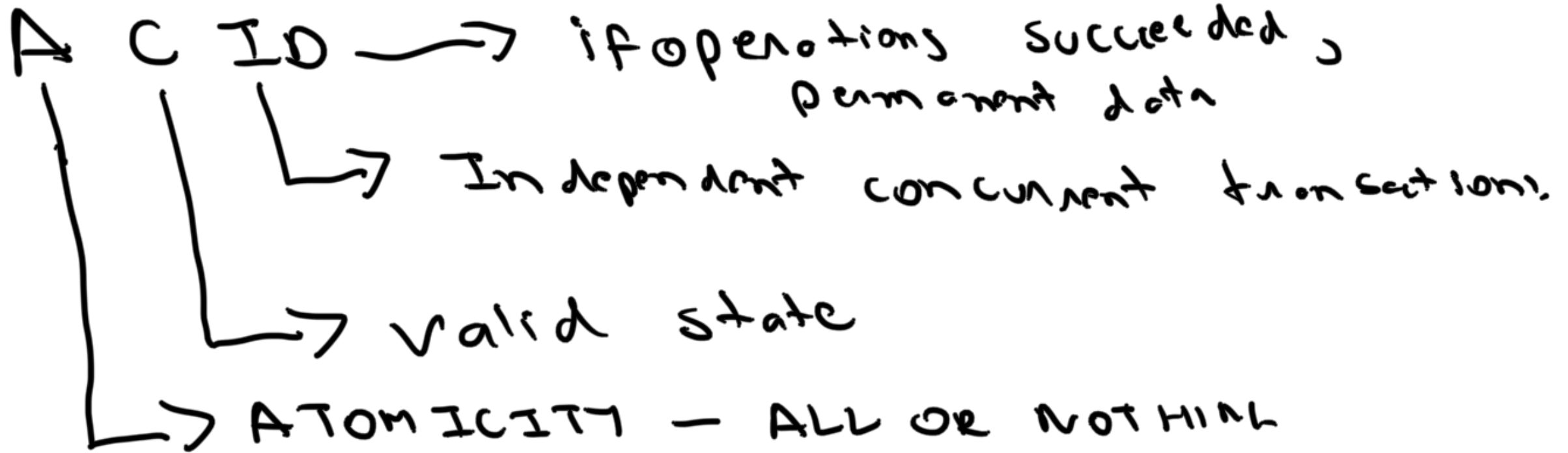
---

Transactions

- Single unit of work against a DB

- whenever related operations,

# In consistency — TRANSACTION



---

BEGIN TRANSACTION

ROLLBACK

COMMIT



CONCEPT

---

Isolation levels