* Inbuilt functions :

  * Numbers
  * Strings
  * Date

* Conditional keywords :

  * If
  * If NULL
  * CASE
  * COALESCE

⇒ NUMBERS

1) ROUND ⇒ round(number, no. of digits after decimal)

ex ⇒ round (2.31789, 2) ⇒ 2.32

round (1.004999, 2) ⇒ 1.00

2) TRUNCATE ⇒ truncate (number, no. of digits after decimal)

ex ⇒ truncate (2.31789 , 2) ⇒ 2.31

truncate (1. 004999. 3) ⇒ 1.004

3) CIELING ⇒ next integer ≥ number

ex ⇒ ceiling(3) ⇒ 3

ceiling(2.31) ⇒ 3

ceiling(1.09) ⇒ 2

4) Floor ⇒ next integer ≤ number

ex ⇒ floor (3) ⇒ 3

floor (2.31) ⇒ 2

floor (1.09) ⇒ 1

5) ABS ⇒ absolute value of a no. (eve)

ABS(-2.94) ⇒ 2.94

ABS(-2) ⇒ 2

ABS(1.34) ⇒ 1.34

6) RAND ⇒ returns a random no. b/w 0 to 1. [0-1]

Q Build a fn that generates an almost random no. b/w [0-10] integers

⇒ rand() * 10

⇒ 0.00999 * 10 ⇒ 0.09

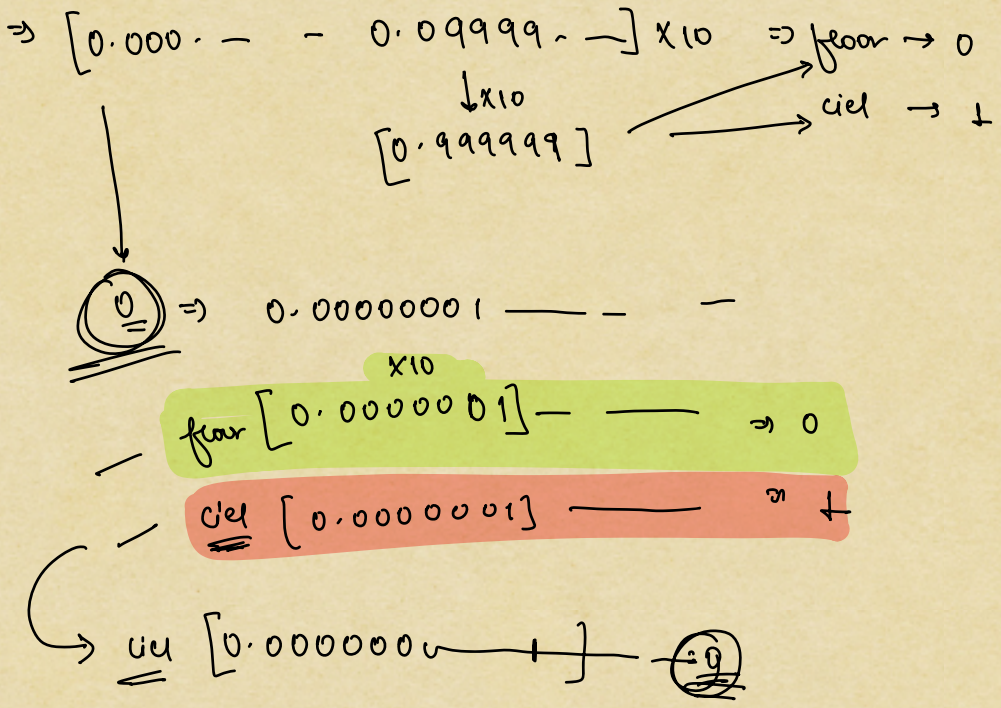
0.142 * 10 ⇒ 1.42

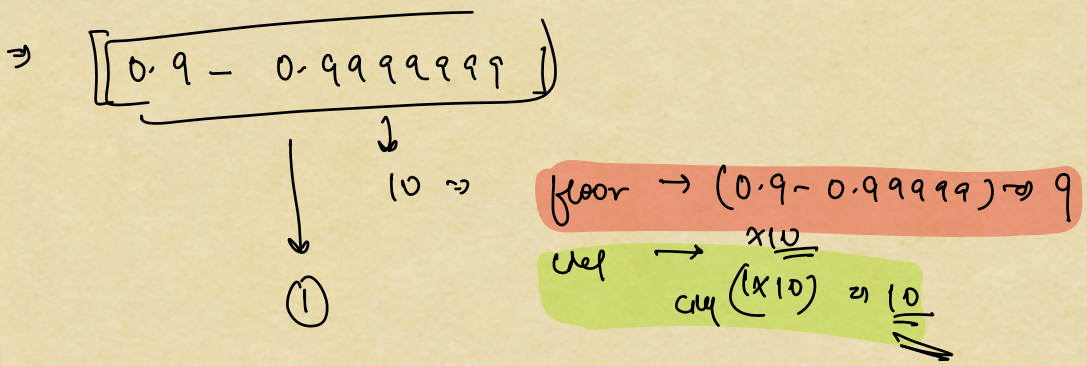
eg ⇒ $1 \times 10 \Rightarrow 10 \Rightarrow ceil(10) \Rightarrow 10$

$floor(10) \Rightarrow 10.$

$0.9999 \times 10 \Rightarrow 9.999 \Rightarrow ceil(9.999) \Rightarrow 10$

$floor(9.999) \Rightarrow 9$





round(round(1), 1) x10

ex: 0.001 ⇒ 0.0 x10 ⇒ 0

0.049999 ⇒ 0

0.050000 — 0.1 x10 ⇒ 1

floor ← ↓ ciel is biased

ciel → floor is biased

⇒ we can't use either

↓

round

round(1) ⇒

(0.000000 ⟶ ⇒ 0.0 x10 ⇒ 0

⇒ 1

⇒ 2

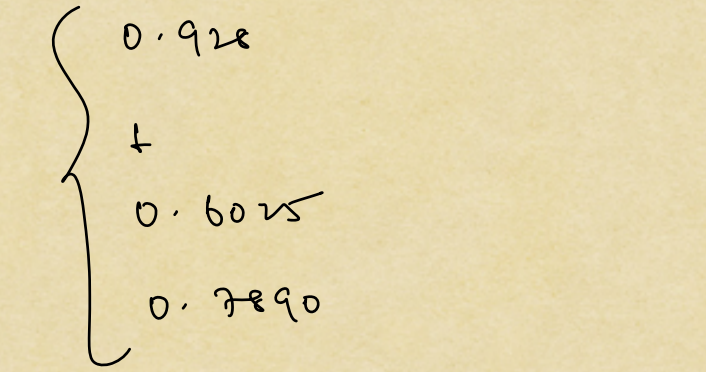
⇒ 3

(1.0000000) ⇒ 1.0 x10 = 10

ex: (0.00049999) x10 ⇒ 0

0.025 ⇒

0.0062



⇒ STRINGS :

1) LENGTH ⇒ length("abcd") ⇒ 4

2) UPPER ⇒ upper("aBcD") ⇒ ABCD

3) LOWER ⇒ lower("aBcD") ⇒ abcd

4) LTRIM ⇒ trim all spaces from left side

ltrim("--Anjana--") ⇒ Anjana--

5) RTRIM ⇒ trim all spaces from right side

rtrim("--Anjana--") ⇒ --Anjana

6) TRIM ⇒ trim all spaces on both sides

trim("--Anjana--") ⇒ Anjana

7) LOCATE ⇒ locate(tofind, fromstring)

locate("el", "delhi") ⇒ 2

⇒ { 0 if doesn't exist
Index from where the search string starts

SQL indexing ⇒ "D E L H I"
1 2 3 4 5

8) Extract a chunk of a string

SUBSTRING ⇒ substring(str, start, length)

start: Starting position
length: how much we want to extract

eg ⇒ substring("ABCDEF", 2, 4)

O/P ⇒ BCDE

⇒ 8.1 ⇒ extracting from left side ⇒ LEFT

eg ⇒ left(str, length)

left("ABCDE", 2) ⇒ AB.

⇒ 8.2 ⇒ Extracting from right side ⇒ RIGHT

ex ⇒ right (str, length)

right ("ABCDE", 2) ⇒ DE.

DATE :

1) NOW ⇒ now() ⇒ gives current date and time

2) CURDATE ⇒ curdate() ⇒ gives current date

3) CURTIME ⇒ curtime() ⇒ gives current time

4) Extractions in date :-

YEAR(DATE) ⇒ YEAR("2023-03-21")

⇒ 2023

MONTH(DATE) ⇒ Month("2023-03-21")

⇒ March

DAYNAME(DATE) ⇒ Dayname("2023-03-21")

⇒ Tuesday

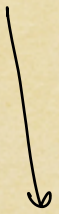
* if we want to add / substract intervals into dates:

i) date_add(date, interval)

↓

x wins

x days



prints the date, after the interval has passed from the given date.

ii) date_sub (date, interval)

↓

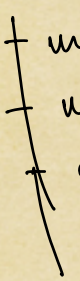
x wins

x days

prints the date, before the interval has passed from the given date.

iii) datediff ⇒ difference b/w 2 dates

- milliseconds
- minutes
- days



* we will do demo on all in-built functions.

⇒ Transactions :

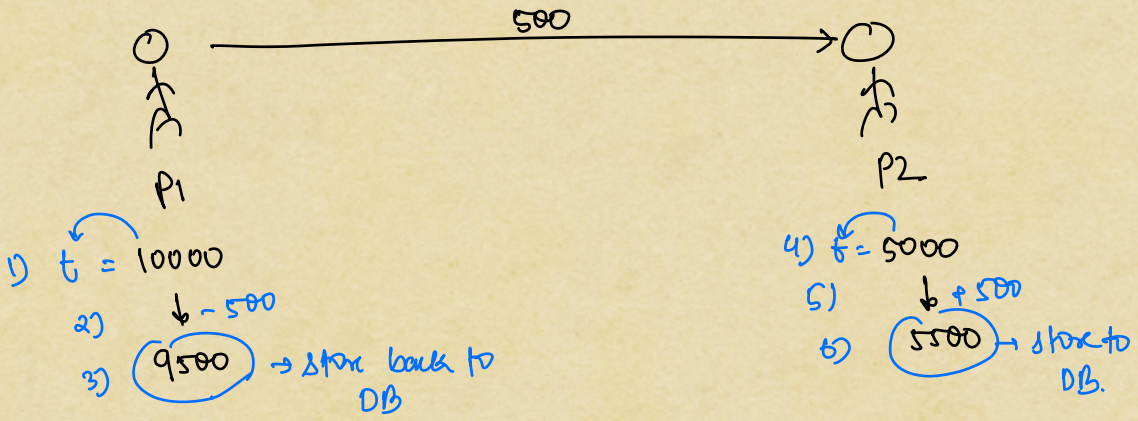
- Intro
- ACID
- Isolation levels
- Demo.

⇒ what are transactions ?

Ans ⇒ A set of logically related operations

* In reality, to perform a task, a single query might not be enough

example ⇒ Bank



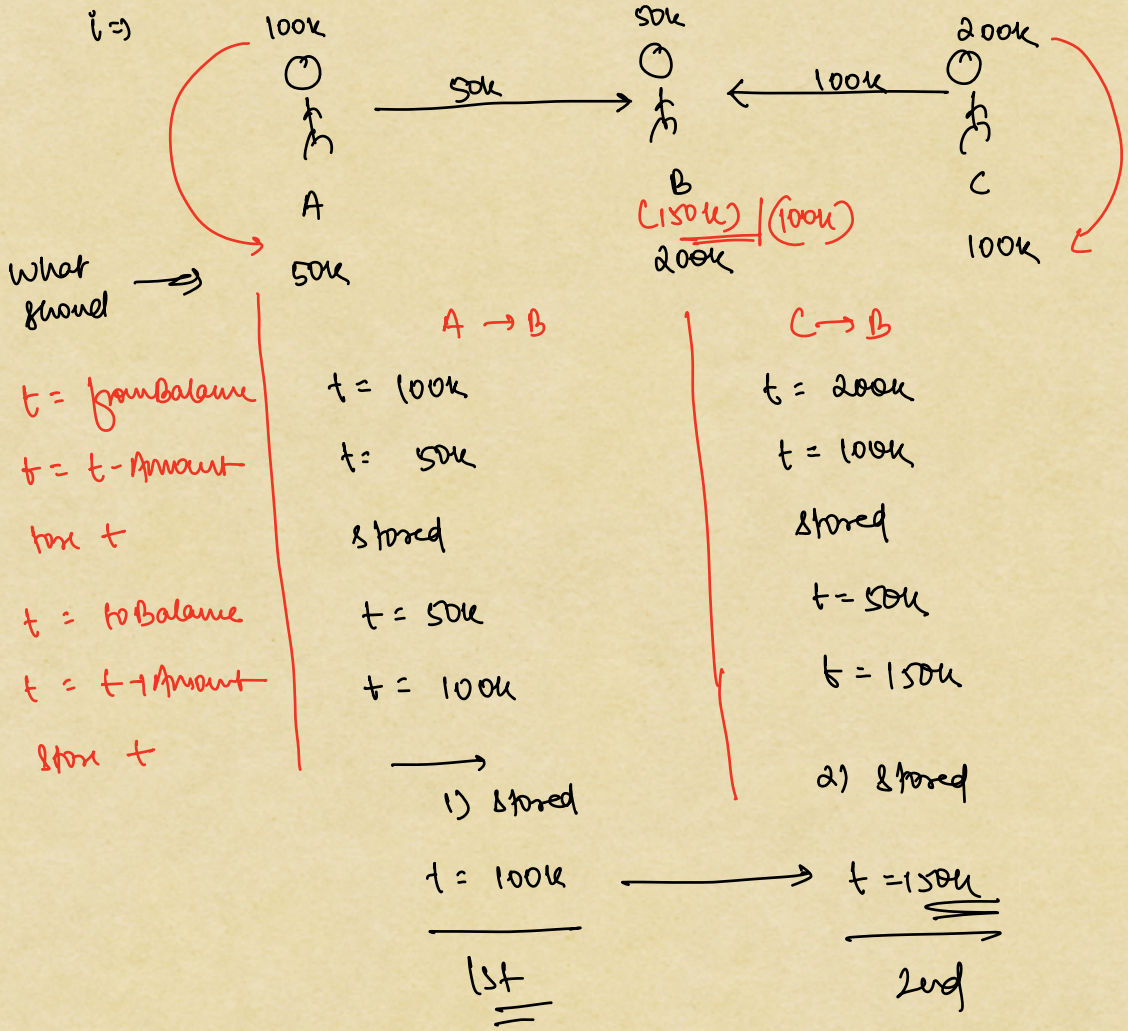

```
SendAmount( fromAccount, toAccount, amount){
    1) t = get the current balance for "fromAccount"
    2) t = t - amount
    3) store t back to DB
```

4) t = get the current balance for "toAccount"

5) t = t + amount

6) store t back to DB

}



* we solve the above problem using transactional isolation.

ACID properties:

A → Atomicity

C → Consistency

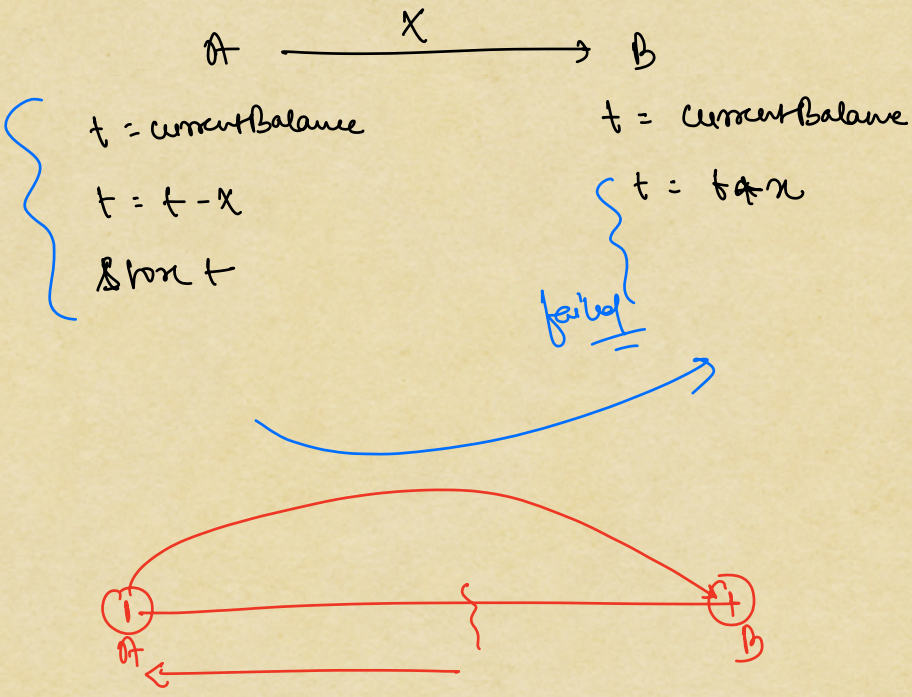
I → Isolation

D → Durability

* **Atomicity:** [0 or 1]

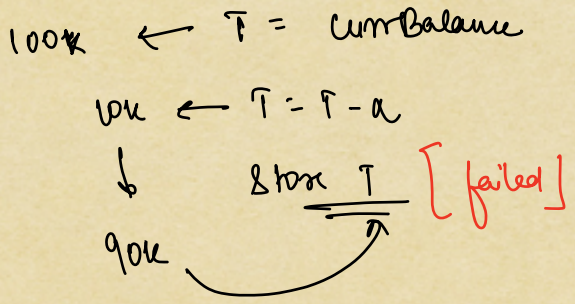
* Smallest possible unit of task
* The entire unit will either complete or nothing will happen

ex: transfer money ⇒ single unit of task to outside

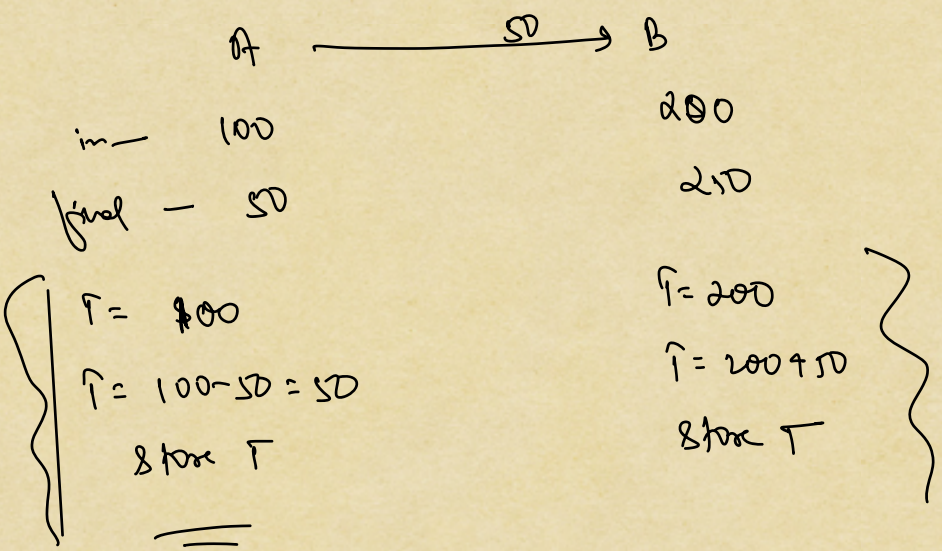


* Consistency : Database should always be consistent.

eg In Bank : withdrawing money from ATM



* System should remain consistent before & after the transaction, during the transaction it can be in a temporary inconsistent state.

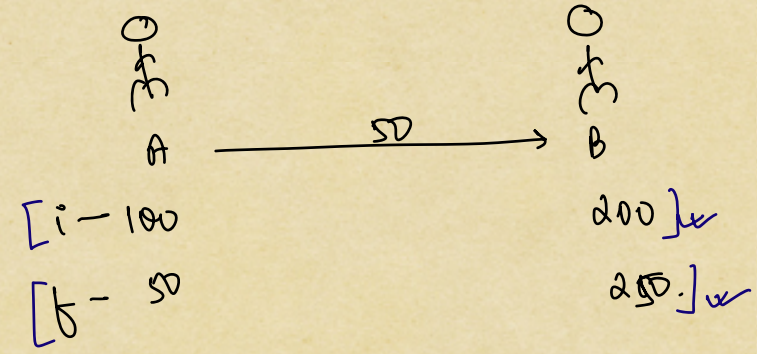


* Isolation: There can be multiple transactions going in the DB but any transaction should not impact the other transactions

* Every transaction should be independent, and produce desired results.

⇒ We can achieve this by configuring "Transaction Isolation levels"

* Durability: Data before or after a transaction should persist.



transaction ⇒ t = 100
t = t - 50
store t

t = 200
t = t + 50
store t