CAP theorem

It is one of the most important theorems for partitions only.

C – Consistency
A – Availability
P – Partitioning

Consistency: It applies to partitions only, every node must be updated with the latest values

Lets say, we have two nodes and data is not updated on Node2 then we perform a Read operation on Node2, should we choose Consistency over Availability or vice-versa? If we choose Availability over Consistency, we will show them the stale data. If we choose Consistency over Availability we don't want people to Read stale data then we will display Error "Data Not found" instead of providing stale data. Basics of CAP theorem. If we say we cannot use any stale data then that means we are prioritizing Consistency over Availability.

Why we need to know about CAP theorem?
Whenever you start thinking about an application, what things you need to think about?
1. What are your Functional requirements?
    - Plan out our Features. Plan how they will work. Plan how system will system respond to different data. How Features interact between themselves and between different users?
2. Make a list of Non-Functional requirements
    - Consistency, Availability
- if we design a Banking application, at that time, we cant say we prefer Availability, we absolutely need Consistency whereas for updating Social media feed or a Document site, we don't need that much Consistency, but we want our data Available all the time, we cant throw an error for unupdated data.

Story:
Two users User123 and User567 are staring a BPO. They both have their own Database Tables where they maintain data. They are only answering the phone. Lets say User123 picked up the phone and makes an entry in the Table about the Customer call (Opg).
User123 table

Customers
Opg
Bell

User567 table

Customers
Cibc
TD

Again Opg calls the toll-free number and goes to User567. Now Opg refers to their previous call but User567 doesn't have the record in their Table. It is in User123's Table. Basically, the BPO company will lose one customer (Opg).

Issues:
1. Consistency: Both Nodes should sync up: they created a channel between them. Whenever User123 updates his table, he will inform User567 to update his table and vice-versa. So now they are maintaining data Consistency. The table updates could happen sync or async.

2. Availability: Say one of the users is not working at the moment. He is taking a break. Lets say User123 makes a list of all the customer and pass it to User567 when he is again up. By that Consistency issue can be solved, after Availability issue is resolved.

3. Partitioning: Someday, conflict happens and communication between users breaks. This is called as Partitioning. When they have split, one option is, they can work on their own (in this case, data will be inconsistent but they will be available) and other option is, they will try to fix the communication channel again (in this case, to fix they will have to wait that will basically affect the Availability for a while. Consistency is affected at the cost of Availability).

CAP theorem basically says at any point of given time, you cant have Consistency and Availability at the same time if Partitioning happens. You can only have 2 out of 3 at the same time.

Partitioning issues:
- Network between nodes could go down
- Latency between nodes
- Connection cable issues

Choose between CP and AP

CP:
- while partitioning, it will give priority to Consistency and block the request
- Avoid stale data
- Prioritize consistency

AP:
- Always available
- Prioritizes availability
- Always respond
- May respond whatever value it has, could be stale value also
- Resolve inconsistency later

If Partition is made, do you prioritize Consistency or Availability?

Examples:
Ticket booking application: it should prioritize Consistency
Social media: it should prioritize Availability
BookMyShow: C
Review application: A
Banking / Payment app: C
Inventory: C
Streaming app: A

For each features also we can apply CAP theorem
CAP tradeoffs:
- mostly systems are not entirely AP or CP
- CAP can be applied at Feature level

Even within a Payment app, we can decide which feature should work on CP and which should work on AP

Why we cant have AP and CP? Because communication is broken between two replicas. So we don't know whether they both have the same data. You are not able to connect with the other replica thats the conflict.

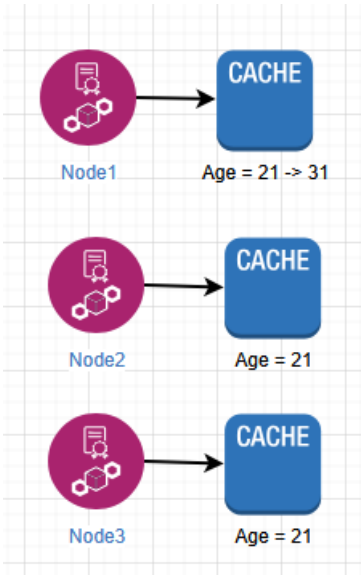
PACELC model: extended version of CAP

If there is Partitioning [P] then choose between Availability [A] and Consistency [C]
Else [E] (single node and no partitioning) choose between Latency [L] and Consistency [C] that's PACELC

CAP talks about Partitioning
PACELC talks about normal operation also. Thats if you have multiple nodes, choose between A and C, if you have only one node (with single database) then choose between L and C.
for Single node application: Say if you choose L, your user can wait for sometime. If you choose C, for the particular operation or feature you have to make it fast. For Single node explanation is, say communication between partitions are working correctly and everything is going good between different partitions, in that case, the system will function like a Single node. Thats the understanding. In that kind of system, where there is absolutely no issue with Partitioning, we got to choose between Latency and Consistency

Eventual Consistency:
Change or update we are making in one node will go to other nodes eventually but not at the same given time / not instantly
- updates to all nodes will not happen instantly but in sometime eventually
- in sometime, all the nodes will have the same updated value
- Async calls

We have 3 Nodes with caches. Node1's cache is updated. Age 21 to 31 but not other caches. However, eventually they will get updated. Kind of async operation. Node1 will inform other nodes that Age needs to be updated but they might put it in the backlog if they are busy with some other tasks because eventual consistency is ok.



Linearizability [Strongest Consistency]:
This is the strongest form of consistency.
- Operations *appear* to occur automatically at the single point in time or instantly.
- Latest data everywhere always

Homework: give it a thought: open WhatsApp and Divide application into Features then think whether you are choosing Availability or Consistency for that particular feature. What's the mechanism of double-tick? How do we get double-tick? When we have different nodes for different users geographically dispersed, when will they get double-tick? Say one is in US and other is in France, they both interact with different nodes, then how do both get double-tick?

Both users are using WhatsApp. When will User123 get double-tick? Either after saving the message "How are you?" in Node1 only or after only saving into Node2 also?