Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN


Foto: UHH/Esfandiari

Databases and Information Systems (DIS) - Quiz

Universität Hamburg

1

Architecture of Database Systems

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

2

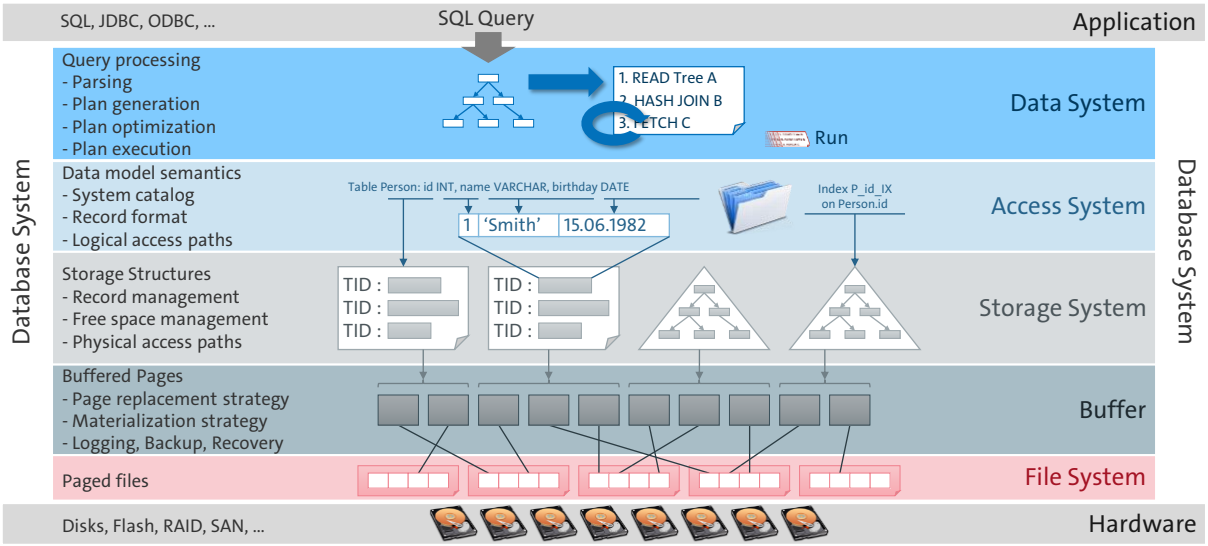
Which of the following layers are a part of the 5-Layer-Model?

- Data System
- Access System
- Network System
- Buffer

Which of the following layers are a part of the 5-Layer-Model?

- Data System
- Access System
- Network System
- Buffer

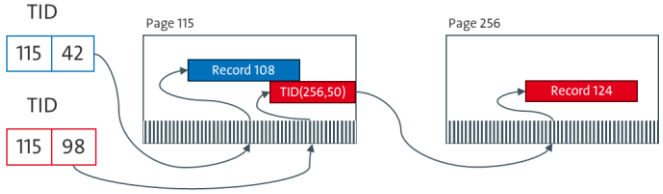
The 5 Layer Model



Explain with your own words what happens to the TID of a record when it is getting larger. What happens if the record is migrated afterwards?

Explain with your own words what happens to the TID of a record when it is getting larger. What happens if the record is migrated afterwards?

1. Nothing happens to the original TID.
2. The original TID still stays the same.
3. In both cases the TID of the current version of the record is stored at the address found via the original TID.



Long answer

Case 1: Space on the page is sufficient to store larger record
→ All records are moved withing the same page, in the page array, the positions of the records on the page are changed (same as with a deleted record), but not the TID
Case 2: Space on the page is not sufficient to store larger record
→ Record is moved to another page and TID is stored on the original page (see Figure). If the record is moved again later, the TID in the original page is changed again → only one additional reference necessary even if record is changed multiple times

7

Transaction Management: Synchronization

Which of the following describe an anomaly that can happenin a Database System without synchronization?

- Phantom Problem
- Unrepeatable Write
- Unrepeatable Read
- Lost Update
- Dirty Read
- Destructive Write

Which of the following describe an anomaly that can happenin a Database System without synchronization?

- Phantom Problem
- Unrepeatable Write
- Unrepeatable Read
- Lost Update
- Dirty Read
- Destructive Write

Unrepeatable Write and Destructive Write are events that can happen in memory management (on the operating system level), but are not part of the Database anomalies

Anomalies

Salary change T_1

```
SELECT SALARY INTO :salary
FROM EMPL
WHERE ENR = 2345

salary := salary + 2000;



UPDATE EMPL
SET SALARY = :salary
WHERE ENR = 2345
```

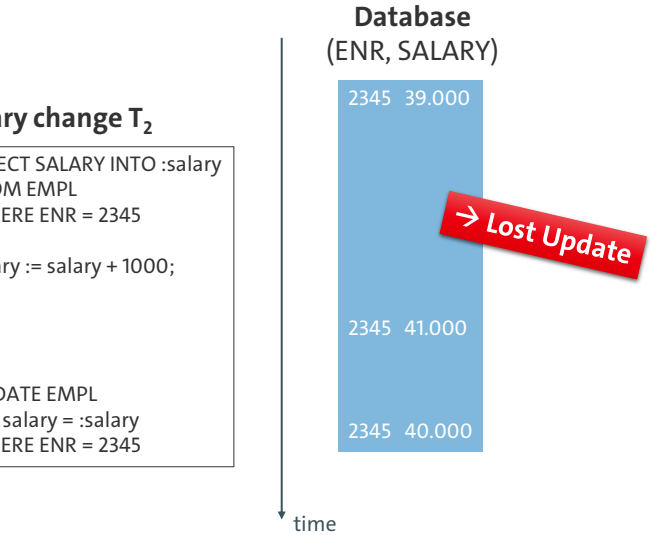
Salary change T_2

```
SELECT SALARY INTO :salary
FROM EMPL
WHERE ENR = 2345

salary := salary + 1000;



UPDATE EMPL
SET salary = :salary
WHERE ENR = 2345
```



Anomalies (II)

Salary change T_1

```
UPDATE EMPL
SET SALARY = SALARY + 1000
WHERE ENR = 2345




ROLLBACK
```

Salary change T_2

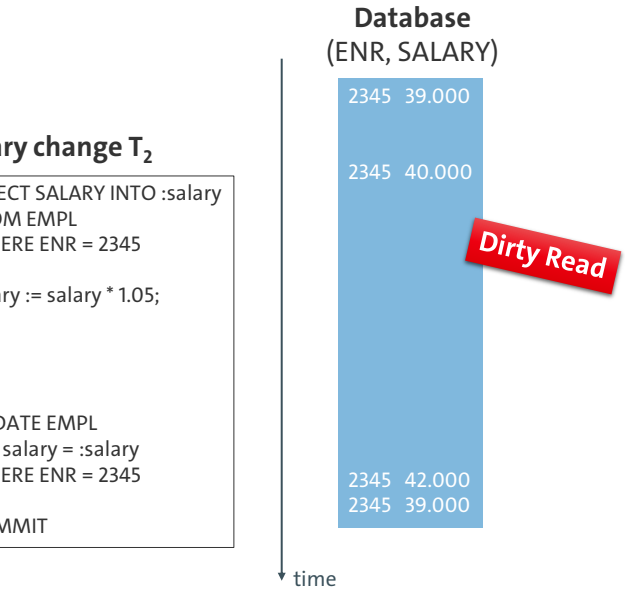
```
SELECT SALARY INTO :salary
FROM EMPL
WHERE ENR = 2345

salary := salary * 1.05;



UPDATE EMPL
SET salary = :salary
WHERE ENR = 2345

COMMIT
```



Anomalies (III)

Salary change T_1

```
UPDATE EMPL
SET SALARY = SALARY + 1000
WHERE ENR = 2345
UPDATE EMPL
SET SALARY = SALARY + 2000
WHERE ENR = 3456
COMMIT
```

Get salaries T_2

```
SELECT SALARY INTO :g1
FROM EMPL
WHERE ENR = 2345

SELECT SALARY INTO :g2
FROM EMPL
WHERE ENR = 3456

sum := g1 + g2
```

Database
(ENR, SALARY)

2345 39.000
3456 45.000

2345 40.000

3456 47.000

Non-Repeatable Read

time

What is the result for *sum* and which result did we expect?

Anomalies (IV)

Get salaries T_1

```
SELECT SUM(Salary)
INTO :Sum1
FROM Empl
WHERE DepNr = 17

SELECT SUM(Salary)
INTO :Sum2
FROM Empl
WHERE DepNr = 17
```

Create new record T_2

```
INSERT INTO Empl(ENR, DepNr, Salary)
Values( 4567, 17, 55.000)
COMMIT
```

Database
(ENR, DepNr, Salary)

Sum

...
2345 17 39.000
3456 17 45.000
...

84.000

...
2345 17 39.000
3456 17 45.000
...
4567 17 55.000

Phantom Problem

139.000

time

Which ANSI-SQL isolation level is supposed to avoid all anomalies?

- Read Uncommitted
- Read Committed
- Repeatable Read
- Serializable

Which ANSI-SQL isolation level is supposed to avoid all anomalies?

- Read Uncommitted
- Read Committed
- Repeatable Read
- **Serializable**

ANSI-SQL isolation levels

Isolation Level	Lost Update possible?	Dirty Read possible?	Non-Repeatable Read possible?	Phantom Read possible?
Read Uncommitted	No	Yes	Yes	Yes
Read Committed	No	No	Yes	Yes
Repeatable Read	No	No	No	Yes
Serializable	No	No	No	No

Are two serializable schedules of the same transactions always equivalent? Elaborate on your answer.

Are two correct, i.e. serializable, schedules of the same transactions always equivalent? Elaborate on your answer.

No, e.g. assume that two transactions, T_1 and T_2, are executed without concurrency, i.e. all operations of the same transaction are finished before the other transaction starts.

→ Whether T_1 is executed first or T_2 is executed first does not matter in term of correctness. But the result might be different depending on which transaction starts, e.g. if non-commutative operations are used.

Correctness and Equivalency

A:= 1000, B:=2000

T_1	T_2
read(A)	
A:=A-50	
write(A)	
read(B)	
B := B+50	
write(B)	
	read(A)
	X := A*0.1
	A := A-X
	write(A)
	read(B)
	B := B+X
	write(B)

Both schedules are correct, but they are not equivalent!

T_1	T_2
read(A)	
A:=A-50	
write(A)	
read(B)	
B := B+50	
write(B)	
	read(A)
	X := A*0.1
	A := A-X
	write(A)
	read(B)
	B := B+X
	write(B)

Results
*A= (1000-50) -(1000-50)*0,1 = 855*
*B= 2000+50+ (1000-50)*0,1 = 2145*

T_1	T_2
	read(A)
	X := A*0.1
	A := A-X
	write(A)
	read(B)
	B := B+X
read(A)	write(B)
A:=A-50	
write(A)	
read(B)	
B := B+50	
write(B)	

Results
*A= 1000-(1000*0,1)-50 = 850*
*B= 2000+(1000*0,1)+50 = 2150*

Which of the following schedules are serializable?

$r_1(A)\ w_3(B)\ w_2(A)\ r_2(B)\ w_3(A)\ r_1(A)$

$r_1(A)\ w_3(B)\ w_2(A)\ r_2(B)\ w_1(A)\ r_1(A)$

Which of the following schedules are serializable?

Dirty Read

$r_1(A)\ w_3(B)\ w_2(A)\ r_2(B)\ w_3(A)\ r_1(A)$

Lost Update

$T_3 < T_2$
$T_2 < T_1$
$T_3 < T_1$
$T_2 < T_3$

Loop $T_2 \leftrightarrow T_3$

Not Serializable

$r_1(A)\ w_3(B)\ w_2(A)\ r_2(B)\ w_1(A)\ r_1(A)$

→ Recap next week

Transaction Management: Logging & Recovery

Which phases do we typically execute during a recovery after a crash?

Which phases do we typically execute during a recovery after a crash?

Analysis | **Redo** | **Undo**

Which of the following statements is not correct?

- During the analysis phase, winners and losers are identified.
- The Redo phase is applied to winners and losers.
- The Undo phase is applied to winners and losers.
- Compensation Log Records are created during the Undo phase.

Which of the following statements is not correct?

During the analysis phase, winners and losers are identified.

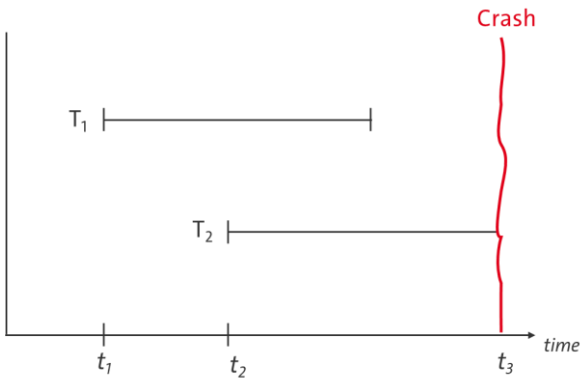
The Redo phase is applied to winners and losers.

The Undo phase is applied to winners and losers.

Compensation Log Records are created during the Undo phase.

The undo phase is only applied to losers

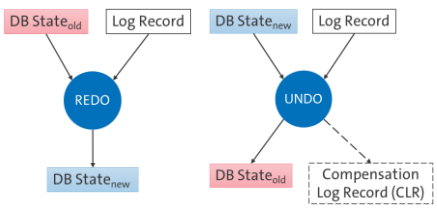
Recovery



Transaction T_1 is a winner → Redo
Transaction T_2 is a loser → Undo

Phases of Recovery

1. **Analysis**
   → Identify winners and losers
2. **Redo**
   → Repetition of history
   → Reads log file forwards
3. **Undo** of losers
   → Reads log file backwards
   → Write CLR



Recovery Example

[#1, T_1, **BOT**, 0]
[#2, T_2, **BOT**, 0]
[#3, T_1, A, A-=50, A+=50, #1]
[#4, T_2, C, C+=100, C-=100, #2]
[#5, T_1, B, B+=50, B-=50, #3]
[#6, T_1, **commit**, #5]
[#7, T_2, A, A-=100, A+=100, #4]

Crash

<#7', T2, A, A+=100, #7, #4>
<#4', T2, C, C-=100, #7', #2>
<#2', T2, -, -, #4', 0>

Analysis
Winner: T_1
Loser: T_2

Redo
IF LSN(A) < 3 THEN A-=50 (and replace LSN(A))
IF LSN(C) < 4 THEN C+=100 (and replace LSN(C))
IF LSN(B) < 5 THEN B+=50 (and replace LSN(B))
IF LSN(A) < 7 THEN A-=100 (and replace LSN(A))

Undo → Only for losers (T_2)
Entry #7
- A+=100
- Write CLR
- LSN(A) = #7'

Entry #4
- C-=100
- Write CLR
- LSN(C) = #4'

Entry #2
- Write CLR

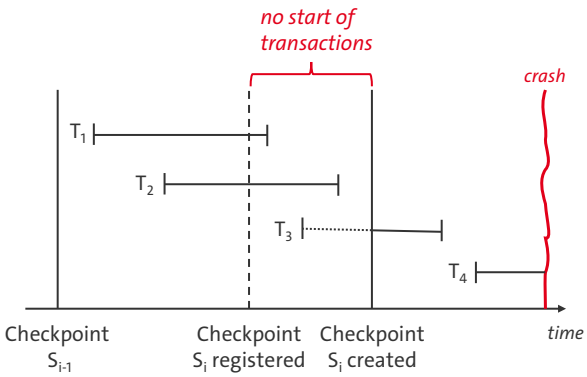
Which of the following checkpoints introduces the longest system downtime?

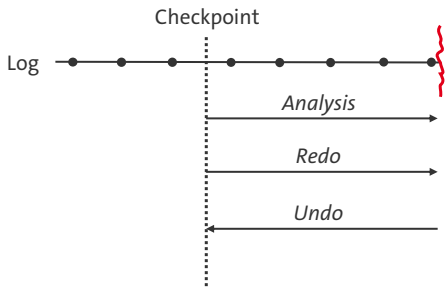
Action Consistent CP | **Transaction Consistent CP** | **Fuzzy CP**

Which of the following checkpoints introduces the longest system downtime?

Action Consistent CP | **Transaction Consistent CP** | **Fuzzy CP**

Transaction Consistent Checkpoints





Modern DBS: Distributed Systems

Which of the following Commit Protocols is the most efficient in terms of the least number of messages and log entries created?

- 1PC
- 2PC
- Linear 2PC
- 2PC with read-only transactions
- 3PC
- 3PC with flattened trees

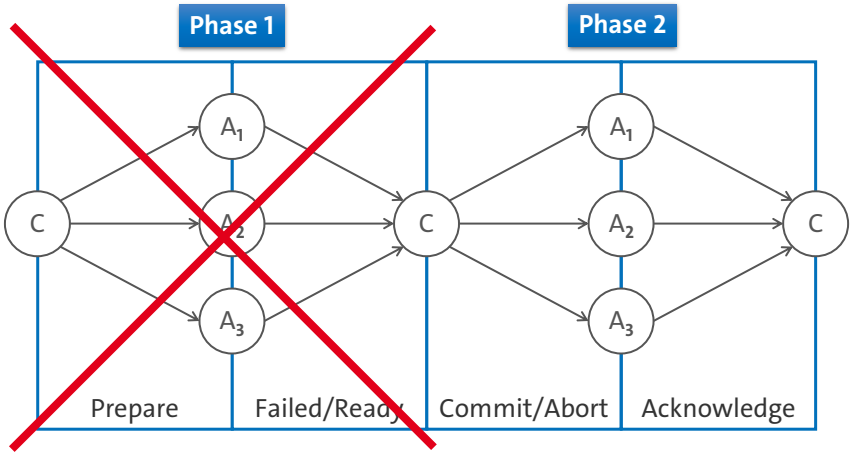
Which of the following Commit Protocols is the most efficient in terms of the least number of messages and log entries created?

- 1PC
- 2PC
- Linear 2PC
- 2PC with read-only (sub)transactions
- 3PC
- 3PC with flattened trees

If all transactions are read-only, 2PC w/ read-only TAs require just as many messages as 1PC.

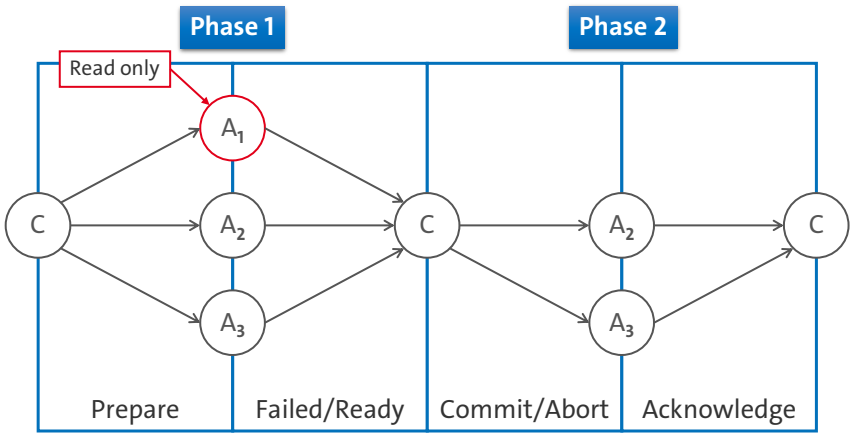
One-Phase-Commit Protocol

- Actual work starts here, before phase 1 (after calling a function *f()*)
- READY message is sent when *f()* has returned, and the participant is aware of the prepare request



Read-Only (Sub-)Transactions with 2PC

No 2nd phase if transaction is only reading



Explain one taxonomy for Heterogeneous Federated Databases of your choice

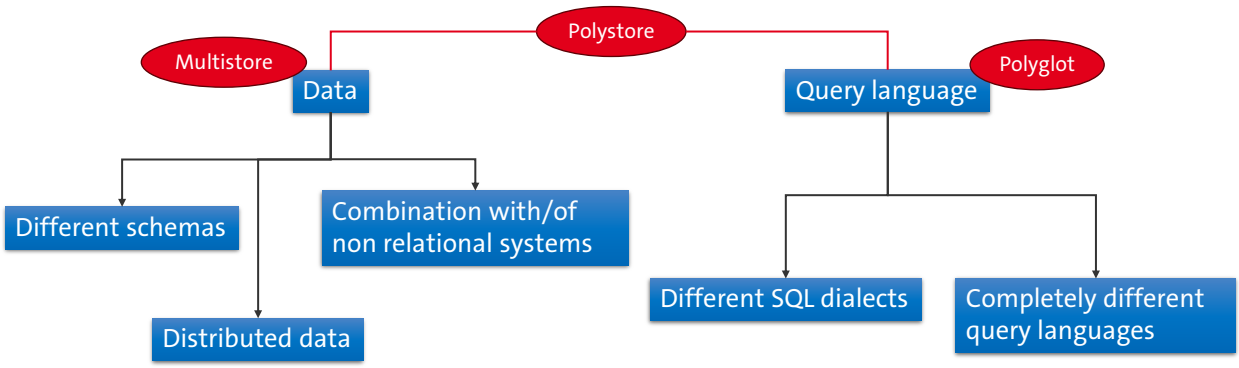
Explain one taxonomy for heterogeneous Federated Databases of your choice

- By autonomy
  - Homogeneous: Local nodes run no local transactions
  - Heterogeneous: Local nodes can run local transactions
- By data coupling
  - Loosely coupled → Local stores accessed by their local language or a common language
  - Tightly coupled → Local stores accessed by a multistore system using the same language for structured and unstructured data (e.g. Hadoop)
  - Hybrid → Some Stores are loosely coupled and some are tightly coupled (e.g. Spark SQL, BigDawg)
- By Data and query language
  - Homogeneous: Data format/storage/location and query language are the same
  - Heterogeneous: Data and/or query language differs

Further reading
An approach to classify Federated DBMS:
Azevedo, Leonardo Guerreiro, et al. "Modern Federated Database Systems: An Overview." ICEIS (1) (2020): 276-283.

Heterogeneous Federated DBs

Components can vary in different aspects → This is one of various possible taxonomies



Modern DBS: Optimizations

Which of the following storage layouts is he most efficient for analytical queries? Why?

Row Store **Column Store**

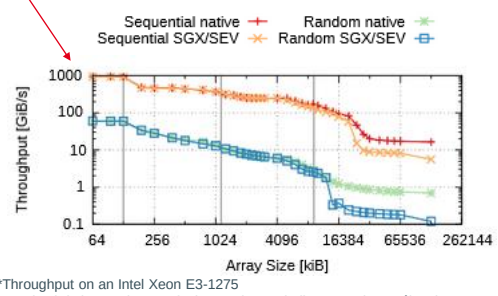
Which of the following storage layouts is he most efficient for analytical queries? Why?

Row Store **Column Store**

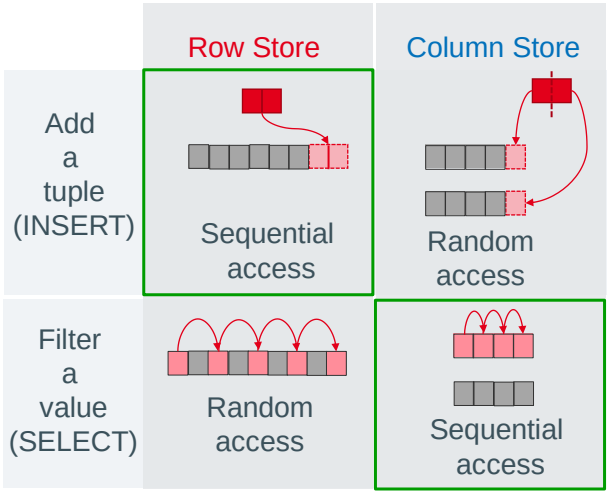
Why should you care?

Memory access is expensive!

Log scale!



*Throughput on an Intel Xeon E3-1275
Gottel, Christian & Pires, Rafael & Rocha, Isabelly & Vaucher, Sébastien & Felber, Pascal & Pasin, Marcelo & Schiavoni, Valerio. (2018). SRDS 2018



Modern DBS: NoSQL Systems

Which of the following paradigms is applied in most NoSQL Systems?

ACID **BASE**

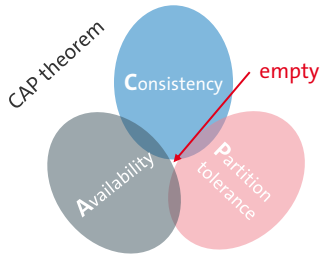
Which of the following paradigms is applied in most NoSQL Systems?

ACID **BASE**

- **B**asically **a**vailable: The database is available for changes, even if these changes are applied later, i.e. it is not necessarily in a consistent state when the user accesses it
- **S**oft state: Data can have a temporary state, e.g. when multiple applications change it at the same time
- **E**ventually consistent: Consistency is reached eventually, but only when (finally) all changes have been applied that were made at the same time

What is the CAP theorem and how is it connected to the BASE and ACID paradigms?

What is the CAP theorem and how is it connected to the BASE and ACID paradigms?



- In a distributed system, only 2 of the 3 properties Availability, Consistency, and partition tolerance can be reached at the same time
- ACID prioritizes Consistency (CA, CP)
- BASE prioritizes Availability (AP)