

Überweisung

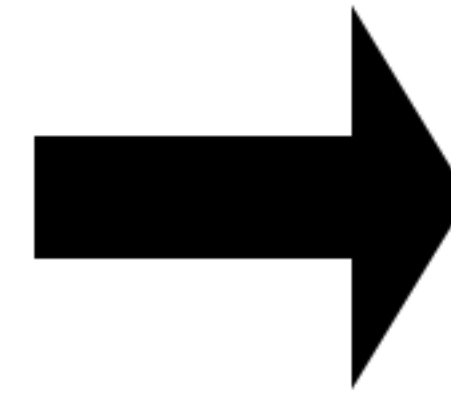
Zustand
vor Überweisung

Konto A
1000 €

Überweise
100 €



Konto B
2000 €

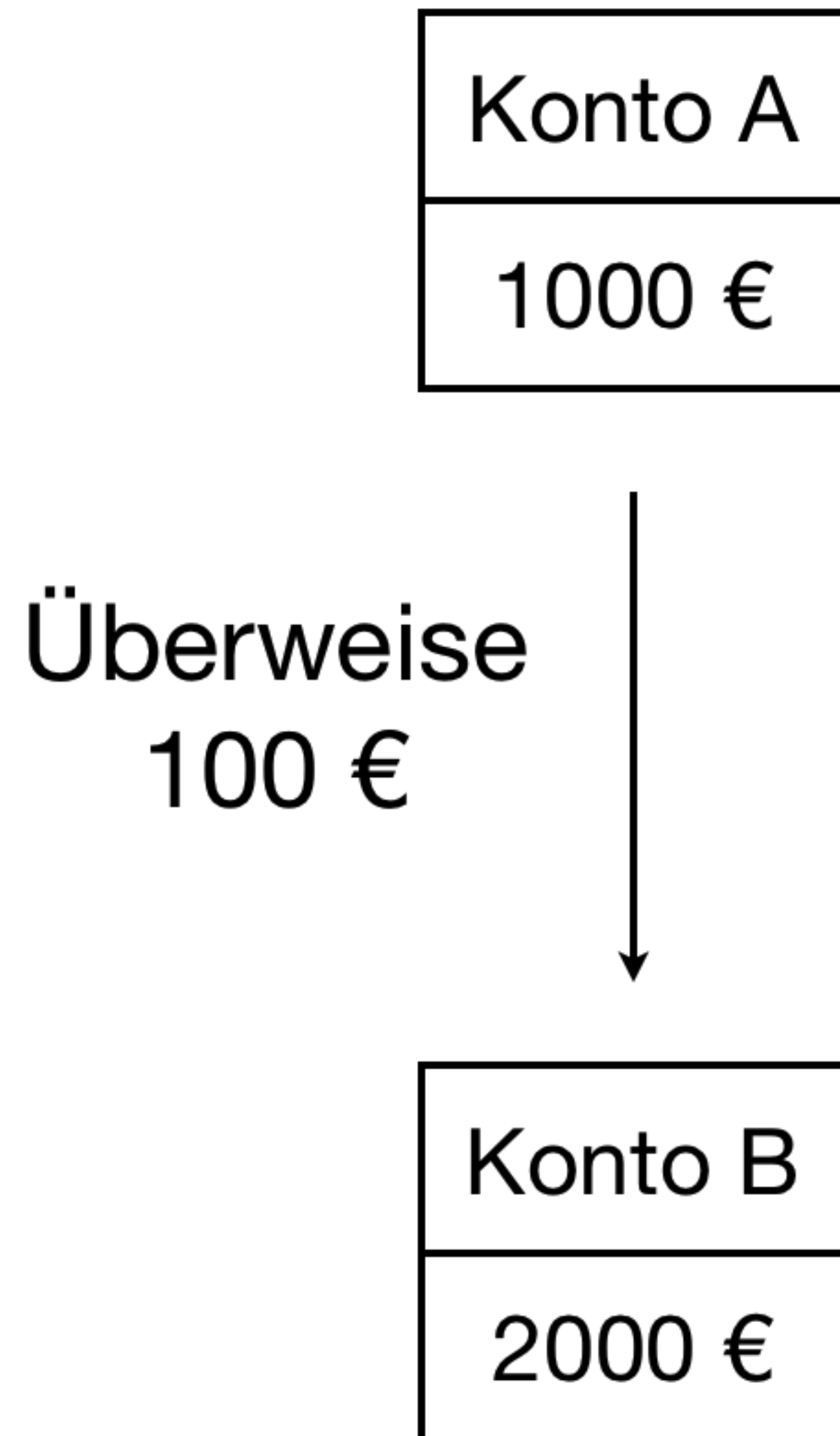


Zustand
nach Überweisung

Konto A
900 €

Konto B
2100 €

Eine Überweisung = mehrere einzelne Aktionen



lese Kontostand von A in k_a

berechne $k_a -= 100$;

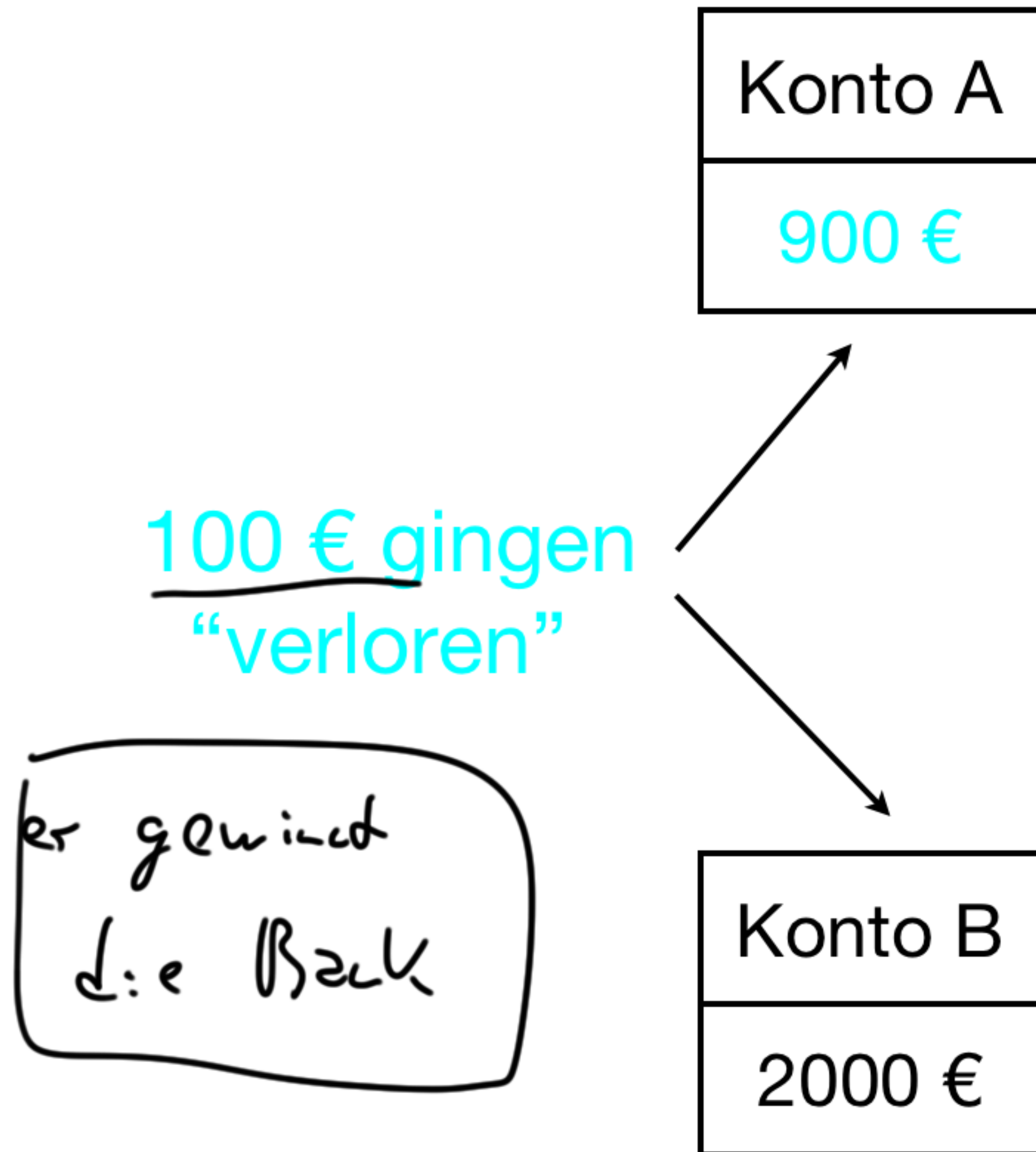
schreibe neuen Kontostand k_a in A

lese Kontostand von B in k_b

berechne $k_b += 100$;

schreibe neuen Kontostand k_b in B

Absturz: Was ist der Zustand der Datenbank?



lese Kontostand von A in k_a

berechne $k_a -= 100$;

schreibe neuen Kontostand k_a in A

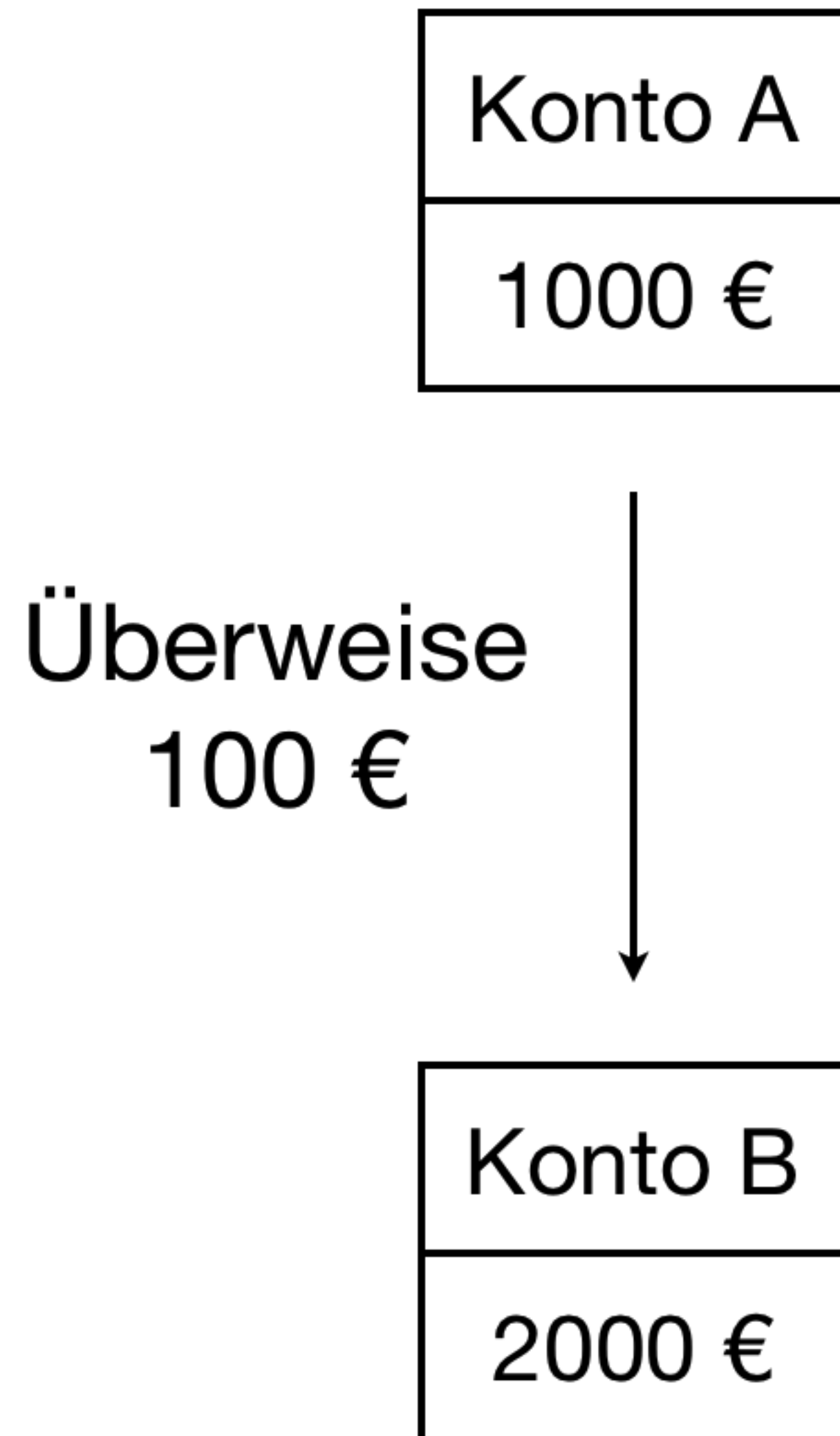
X
Absturz!

lese Kontostand von B in k_b

berechne $k_b += 100$;

schreibe neuen Kontostand k_b in B

Bündele mehrere Operationen zu einer logischen Einheit

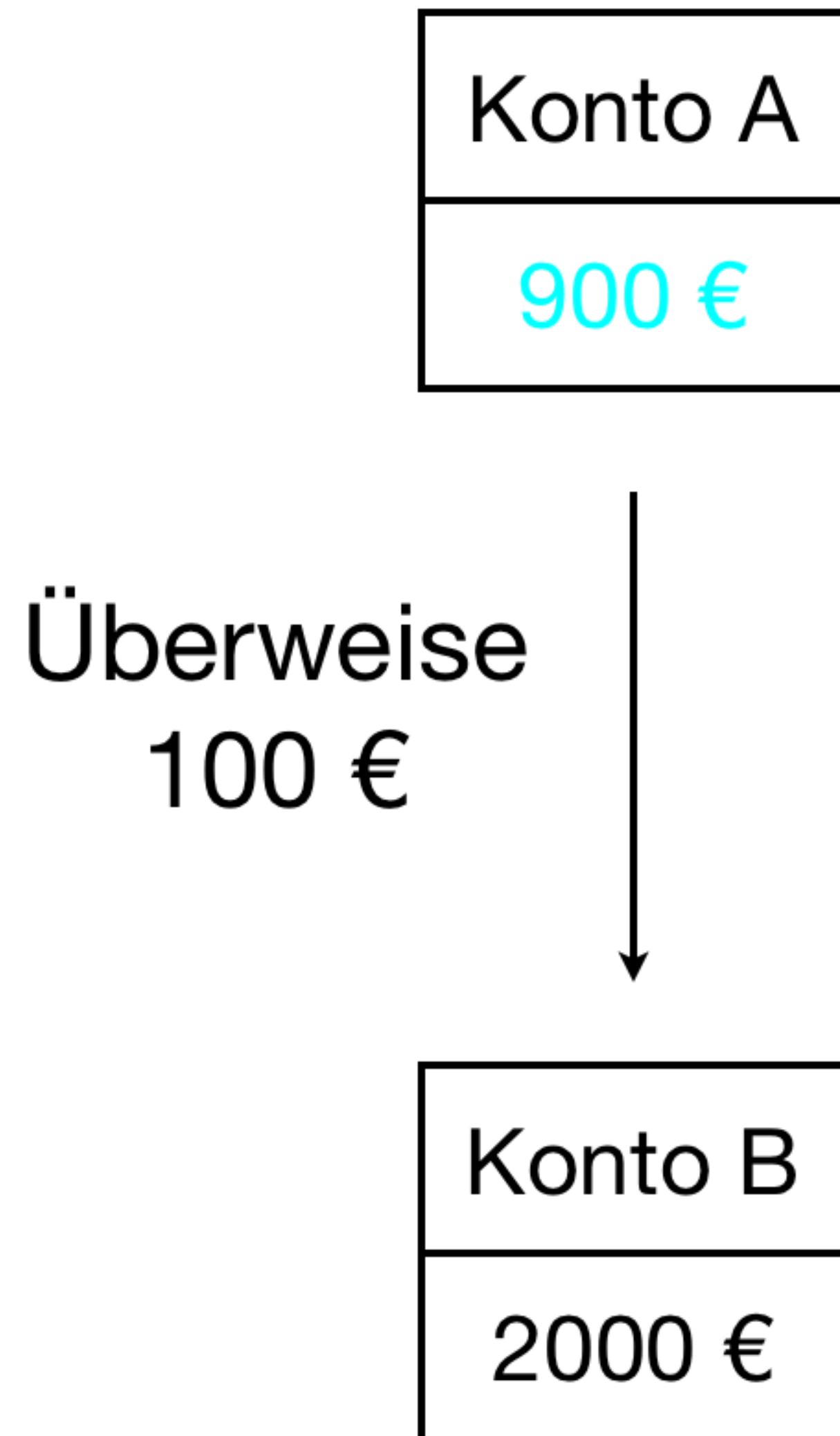


Transaktion

lese Kontostand von A in k_a
berechne $k_a -= 100$;
schreibe neuen Kontostand k_a in A

lese Kontostand von B in k_b
berechne $k_b += 100$;
schreibe neuen Kontostand k_b in B

Absturz: Was ist der Zustand der Datenbank?



Transaktion

- ✓ lese Kontostand von A in k_a
 - ✓ berechne $k_a -= 100$;
 - ✓ schreibe neuen Kontostand k_a in A
-
- Absturz!
- lese Kontostand von B in k_b
 - berechne $k_b += 100$;
 - schreibe neuen Kontostand k_b in B

Zurücksetzen zum Zustand vor der Transaktion!

Konto A
900 €

Konto B
2000 €

ROLLBACK ↑

Transaktion

lese Kontostand von A in k_a

berechne $k_a -= 100$;

schreibe neuen Kontostand k_a in A



Absturz!

lese Kontostand von B in k_b

berechne $k_b += 100$;

schreibe neuen Kontostand k_b in B

Fertig.

Konto A
1000 €

Konto B
2000 €

ROLLBACK ↑

Transaktion

lese Kontostand von A in k_a

berechne $k_a -= 100$;

schreibe neuen Kontostand k_a in A



Absturz!

lese Kontostand von B in k_b

berechne $k_b += 100$;

schreibe neuen Kontostand k_b in B

Atomarität (Atomicity)

= Alle Aktionen der
Transaktion werden
ausgeführt oder keine
(alles oder nichts).

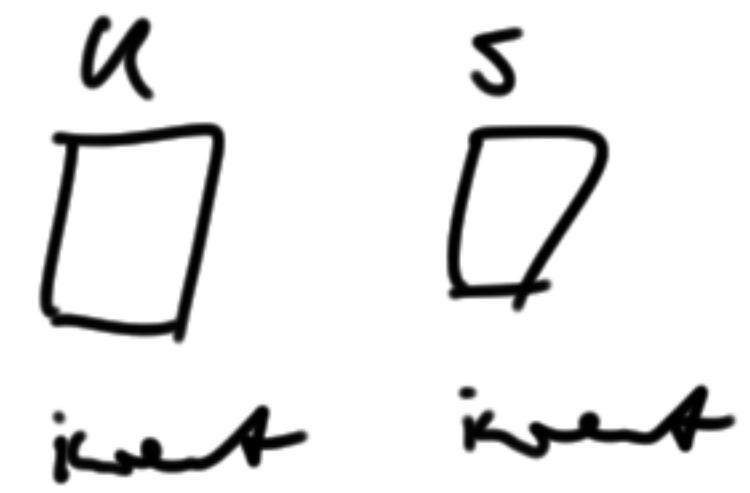
Konsistenz (Consistency)

= Die Transaktion
hinterlässt die
Datenbank in einem
konsistenten Zustand.

Kein Konto darf überzogen
werden:

$k_a < 0$ nach Transaktion
→ ROLLBACK

Integrität ist bedingt
Freiwilligkeit



Transaktion

lese Kontostand von A in k_a
berechne $k_a -= 100$;
schreibe neuen Kontostand k_a in A

lese Kontostand von B in k_b
berechne $k_b += 100$;
schreibe neuen Kontostand k_b in B



Isolation (Isolation)

= Nebenläufige
Transaktionen
beeinflussen sich
(logisch) nicht.



Beispiel für Problem mit Isolation

Überweise 100 € von A nach B

Transaktion 1

lese Kontostand von A in k_a
berechne $k_a -= 100$;
schreibe neuen Kontostand k_a in A

lese Kontostand von B in k_b
berechne $k_b += 100$;
schreibe neuen Kontostand k_b in B

Serialisierung

nötig

→ Isolation Levels

Überweise 300 € von C nach B

Transaktion 2

lese Kontostand von C in k_c
berechne $k_c -= 300$;
schreibe neuen Kontostand k_c in C

lese Kontostand von B in k_b
berechne $k_b += 300$;
schreibe neuen Kontostand k_b in B

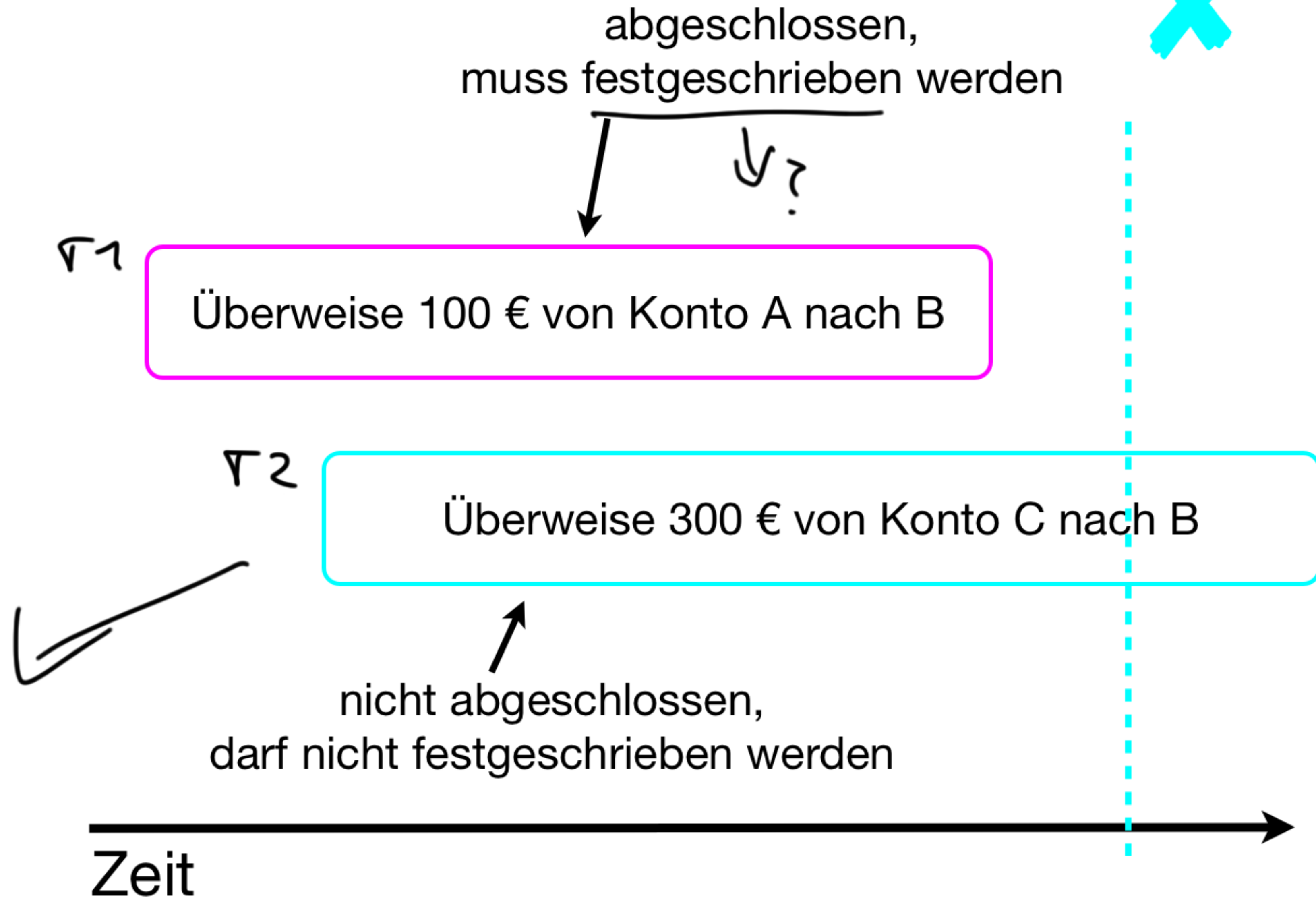
Dauerhaftigkeit (Durability)

= Aktionen einer erfolgreich abgeschlossenen Transaktion bleiben dauerhaft in der Datenbank erhalten.

selbst relativ

↳ Atomicity

Absturz!



ACID

Atomarität	(A tomicity)	← K _{conf}
Konsistenz	(C onsistency)	← K _{conf}
Isolation	(I solation)	← K _{conf}
<u>Dauerhaftigkeit</u>	(D urability)	← K _{conf}

relationale DBMS → ACID - Systeme

= moderne Mytlos