Datenbanken – 11

1)
a)
T2 versucht, 2500 vom Stand aller Konti abzuziehen, doch der Constraint besagt Stand >= 0, bei Konto A wäre der neue Stand -500, was den Constraint verletzt, weswegen die Transaktion abgebrochen wird. Kein Stand wird verändert, da die ganze Transaktion abgebrochen wird.
T3 kommt zum Update, sieht eine Schreibsperre, wartet bis T2 fertig ist, sieht dann, dass T2 abgebrochen wurde, und kann sein Update ausführen. Der Commit ist erfolgreich, da das Update keinen Constraint verletzt. Somit ist bei A der Stand 4000 und bei B 3000.
T4 beginnt und kreiert einen Snapshot noch während T3 läuft. Wenn T4 zum Update kommt, sieht sie im Snapshot, der seither nicht aktualisiert wurde, dass eine Schreibsperre besteht. Da T3 schlussendlich erfolgreich abschliesst, wird T4 abgebrochen. Der Stand ist daher unverändert.
T5 beginnt nach T3 und während T4. T5s Snapshot beinhaltet also das erfolgreiche T3 und wartet ab, was mit T4 passiert. Da T4 nicht erfolgreich ist, kann laut Fallunterscheidug T5 updaten und den Stand von B verdreifachen. Somit ist der Stand von A 4000 und von B 9000.

SELECT * FROM konto;

konto

Kid	Stand
A	4000
B	9000

b)
pg_clog

Xid	Status
1	2
2	1
3	2
4	1
5	2

2)
a)

```
SELECT l.name , a.aid , f.leiter
FROM l, a, f
WHERE l.lid = a.lid
AND f.fid = a.fid
AND f.leiter = 'Momo ' ;
```

π l.name , a.aid , f.leiter σ l.lid=a.lid $\wedge$ f.fid=a.fid $\wedge$ f.leiter='Momo' ((l $\times$ a) $\times$ f)

b)
Statt kartesisches Produkt von 3 Tabellen, erst f.leiter selektieren, dann mit Θ-Joins l.name und a.aid suchen, was effizient geht, da die Join-Bedingungen Primärschlüssel sind und somit ein Index existiert.

π l.name , a.aid , f.leiter (l $\bowtie$ l.lid=a.lid (a $\bowtie$ a.fid=f.fid (σ f.leiter='Momo' (f))))