

Q1

HW4 Query 1

Query

```
SELECT
  s.title
  , sd.director
FROM
  series s
  , seriesdirectors sd
WHERE
  s.seriesid = sd.seriesid
  and s.imdbrating <= 5
  and s.seasons >= 15
ORDER BY
  title
  , director
;
```

Index Creation

```
CREATE INDEX serieshw4q1
ON series (imdbrating, seasons, seriesid);
```

Origin Plan

```
Sort (cost=287.37..287.38 rows=1 width=30)
  Sort Key: s.title, sd.director
  -> Hash Join (cost=3.92..287.36 rows=1 width=30)
        Hash Cond: (s.seriesid = sd.seriesid)
        -> Seq Scan on series s (cost=0.00..283.39 rows=3 width=21)
              Filter: ((imdbrating <= '5'::double precision) AND (seasons
>= 15))
        -> Hash (cost=2.30..2.30 rows=130 width=17)
              -> Seq Scan on seriesdirectors sd (cost=0.00..2.30 rows=130
width=17)
```

Full Plan After Index Creation

```
Sort (cost=19.42..19.43 rows=1 width=30)
  Sort Key: s.title, sd.director
```

```

-> Hash Join (cost=16.77..19.41 rows=1 width=30)
    Hash Cond: (sd.seriesid = s.seriesid)
    -> Seq Scan on seriesdirectors sd (cost=0.00..2.30 rows=130
width=17)
        -> Hash (cost=16.74..16.74 rows=3 width=21)
            -> Bitmap Heap Scan on series s (cost=5.69..16.74 rows=3
width=21)
                Recheck Cond: ((imdbrating <= '5'::double precision)
AND (seasons >= 15))
                -> Bitmap Index Scan on serieshw4q1 (cost=0.00..5.69
rows=3 width=0)
                    Index Cond: ((imdbrating <= '5'::double
precision) AND (seasons >= 15))

```

Conclusion

Plan cost reduced a lot by only doing index scan and read for instead of sequence scan for series table.

HW4 Query 2

Query

```

SELECT
    count(*) as nummovies
FROM
    movies m
WHERE
    m.imdbrating is null
    and m.rottentomatoes is null
    and (m.year is null or m.year>2015);

```

Index Creation

```

CREATE INDEX moviewhw4q2
ON movies (imdbrating, rottentomatoes, year);

```

Origin Plan

```

Aggregate (cost=120.68..120.69 rows=1 width=8)
-> Seq Scan on movies m (cost=0.00..120.61 rows=27 width=0)
    Filter: ((imdbrating IS NULL) AND (rottentomatoes IS NULL) AND
((year IS NULL) OR (year > 2015)))

```

Full Plan After Index Creation

```
Aggregate  (cost=5.52..5.53 rows=1 width=8)
->  Index Only Scan using moviewhw4q2 on movies m  (cost=0.28..5.45
rows=27 width=0)
    Index Cond: ((imdbrating IS NULL) AND (rottentomatoes IS NULL))
    Filter: ((year IS NULL) OR (year > 2015))
```

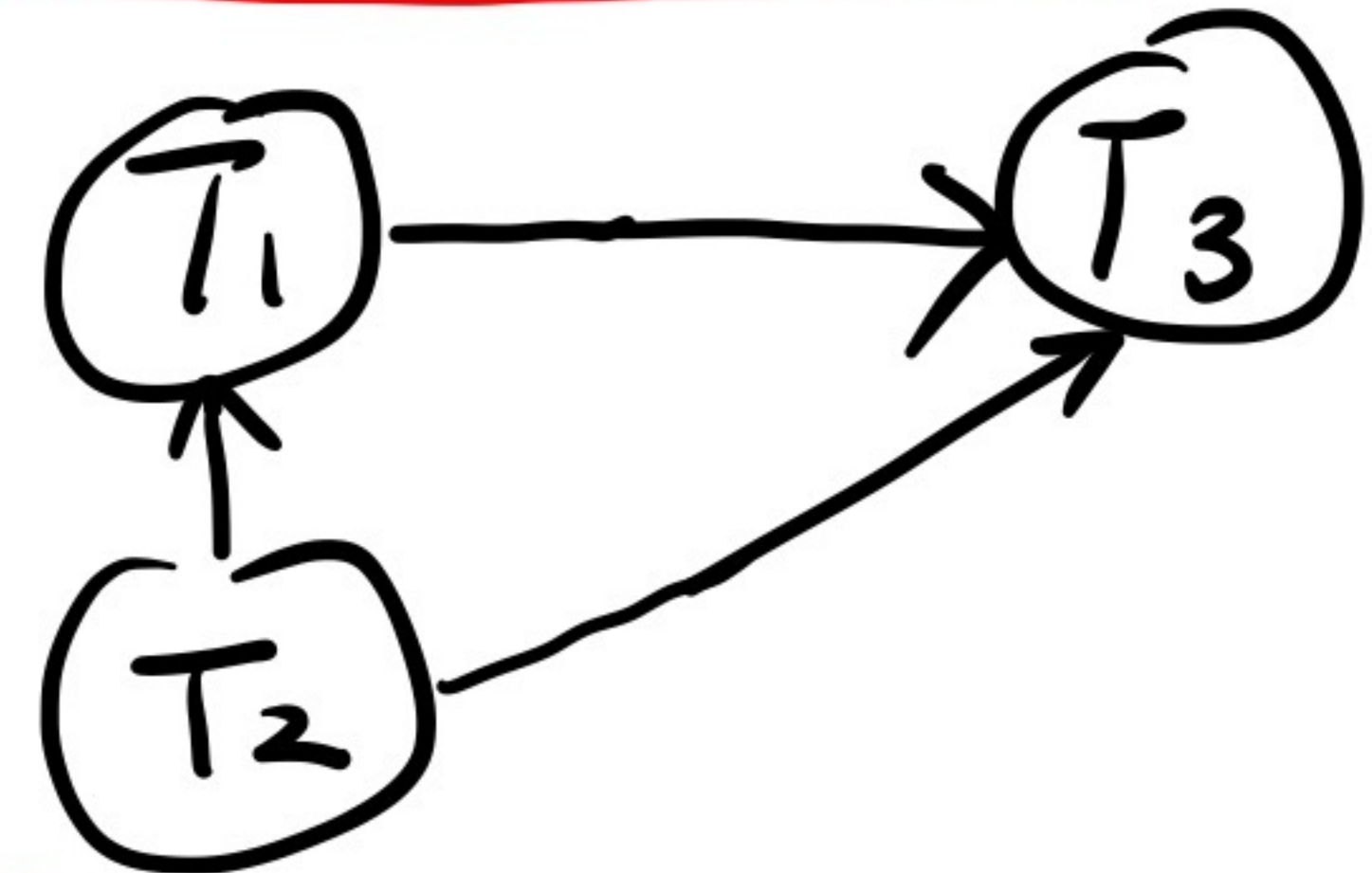
Conclusion

Plan cost reduced significant by only doing index scan instead of sequence scan for movies table.

Q2

S1: r1(x) r2(z) r1(y) w2(w) w2(z) r3(z) w3(x) r1(w) w1(y) w3(z)

a) Conflict Graph:



∴ Serializable

b)

S1: r1(x) r2(z) r1(y) w2(w) w2(z) r3(z) w3(x) r1(w) w1(y) w3(z)

$SL_1(x)$ $SL_2(z)$ $SL_1(y)$ $XL_2(w)$ $XL_2(z)$ $SL_3(z)$ $[UL_1(x), XL_3(x)]$

T₁: unable to acquire lock.

T₁ shrinking

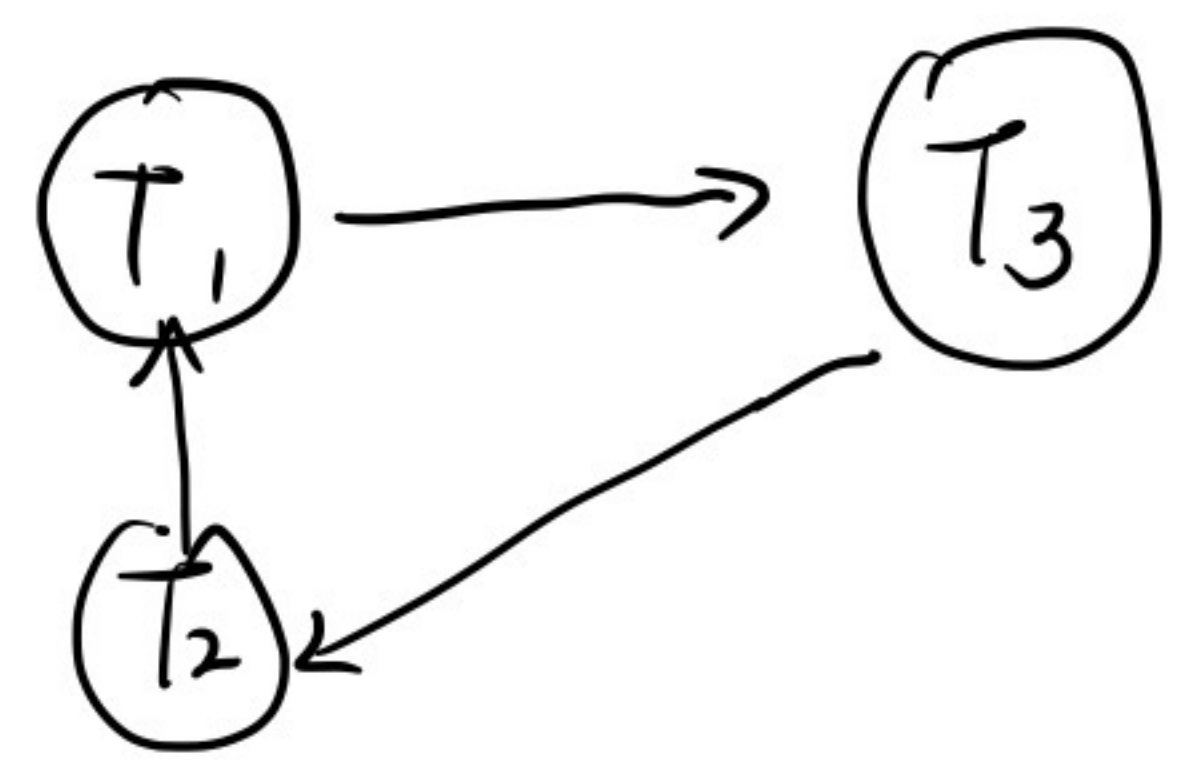
∴ Not possible to obtain schedule, T₁ will try acquiring SL on w in shrinking phase.

(2)

S2: r1(x) r1(y) r2(z) r3(z) r3(x) w3(x) w2(w) w2(z) r1(w) w1(y)

a)

Conflict Graph :



∴ NOT serializable

b) Not possible, since this schedule is not serializable, there will always a cycle prevent valid schedule.

Q 3

LOG:	LSN	Entry
	100	T1 update P2
	101	T2 update P1
	102	T2 commit
	103	T3 update P1
	104	T3 update P3
	105	T1 update P4
	106	T3 commit

Data Page	LSN of Last recorded log entry
P1	101
P2	100
P3	104
P4	-

a)

P_2 100

P_1 ~~101~~ 103

P_3 104

P_4 ~~105~~ - T_1 should be aborted.

1. First, redo : 103, 105

2. Then, undo: 105, 100

b)

No Force, since T_3 is partially written to data page after commit, which won't happen if force used

c) STEAL used, as T_1 is an uncommitted transaction but its changes reflected to the DATA page., also, no force is used, which makes it impossible to be part of the force action.