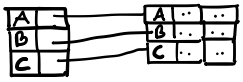
1) $sl_1(V)\, r_1(x)\, xl_3(x)\, w_3(x)\, xl_3(y)\, w_3(y)\, xl_3(z)\, sl_3(V)\, u_3(x)\, sl_1(x)\, r_1(x)\, u_3(y)\, sl_2(y)\, r_2(y)\, u_1(x)\, u_1(v)\, xl_2(x)\, w_2(x)$

- $w_3(z)\, u_3(z)\, sl_2(z)\, r_2(z)\, sl_2(v)\, r_2(v)\, r_3(v)\, u_3(v)\, xl_2(z)\, u_2(v)\, xl_4(v)\, w_4(v)$! Now T_4 and T_2 both want access to z → Not 2PL
- $T_4 < T_2$ for the last write of z
  $T_2 < T_4$ because must read from "default" v] → impossible → Not view-serializable
- Not ACR cause $r_1(x)$ reads from T_3 not already committed
  ⇒ Not strict ⇒ Not rigorous

Not recoverable cause T_1 commits before T_3

2) By definition if a schedule is accepted by a timestamp based scheduler then it is conflict-serializable ⇒ view-serializable.
Thomas Rules just ignore outdated writes and the obtained schedule is conflict serializable

3)
Sorted: entries are sorted
Primary: search key includes primary key of relation
Dense: one data entry for each data record of data file
Clustered: sorted according some criteria of data file



To find the given K we can use binary search.

4) Ordered file on code
$1K/20 = 50$ fields per page → 5 tuples per page → $10KK/5 = 200\,000$ pages

- Since we have to show all books we do $200\,000$ p.a.
- To access the first needed book (code ≥ 30) we use binary search: $\log_2 200\,000$ + #page needed to store book with code ≥ 30 and ≤ 50

5) R without duplicates must fit $M-1$ buffers: read one page at time and keep track of already seen records in hash/tree in $M-1$ buffers

- Read R and create $M-1$ sorted sublists of M pages. Load one page for each of them check duplicates
  $B(R)/M \leq M-1 \sim B(R)/M + 1 \leq M-1 = B(R) \leq M\cdot(M-2) \sim B(R) \leq (M-2)^2$ ⇒ we need $\sqrt{B(R)} + 2$ buffers frames