Reads-from:

T_1 reads from T_2 (on item A) if, in the schedule, we have $W_2(A), \ldots, R_1(A)$ without any writes in between

T_1 reads from T_2, then T_2 gets rolled back – what do we do with T_1 then?

$W_2(A), \ldots, R_1(A), \ldots, C_1, C_2$

reads from

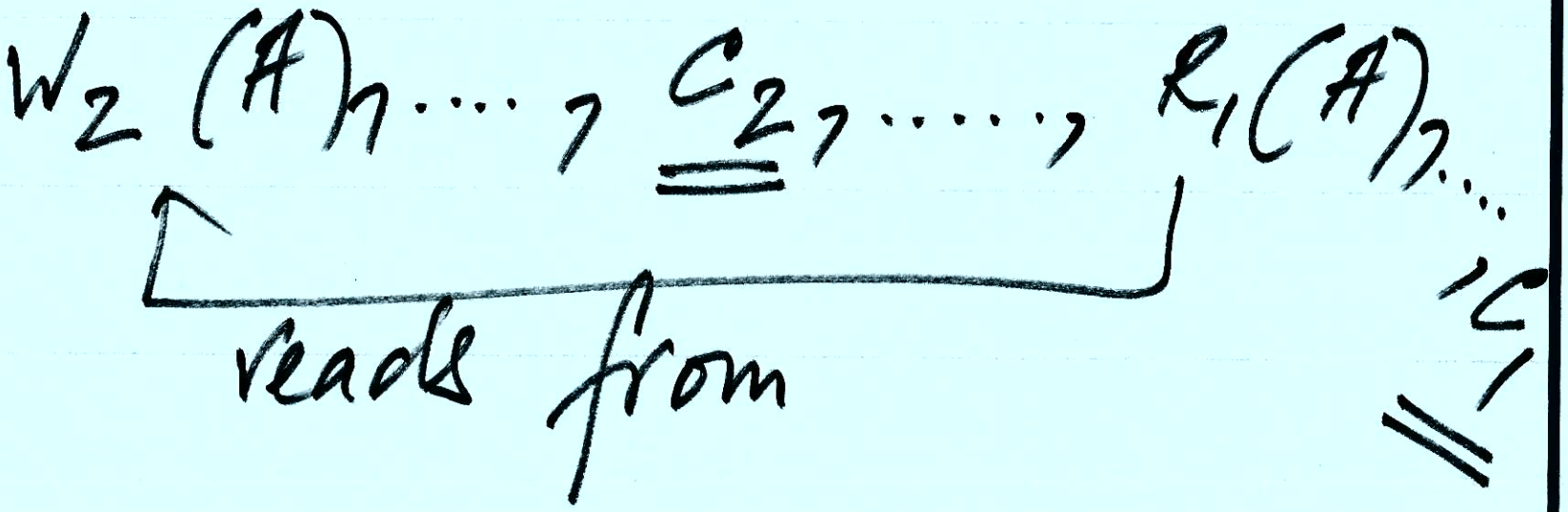
Recoverable: Intuition:

$W_2(A), \ldots, R_1(A), \ldots, C_2, \ldots$

reads from

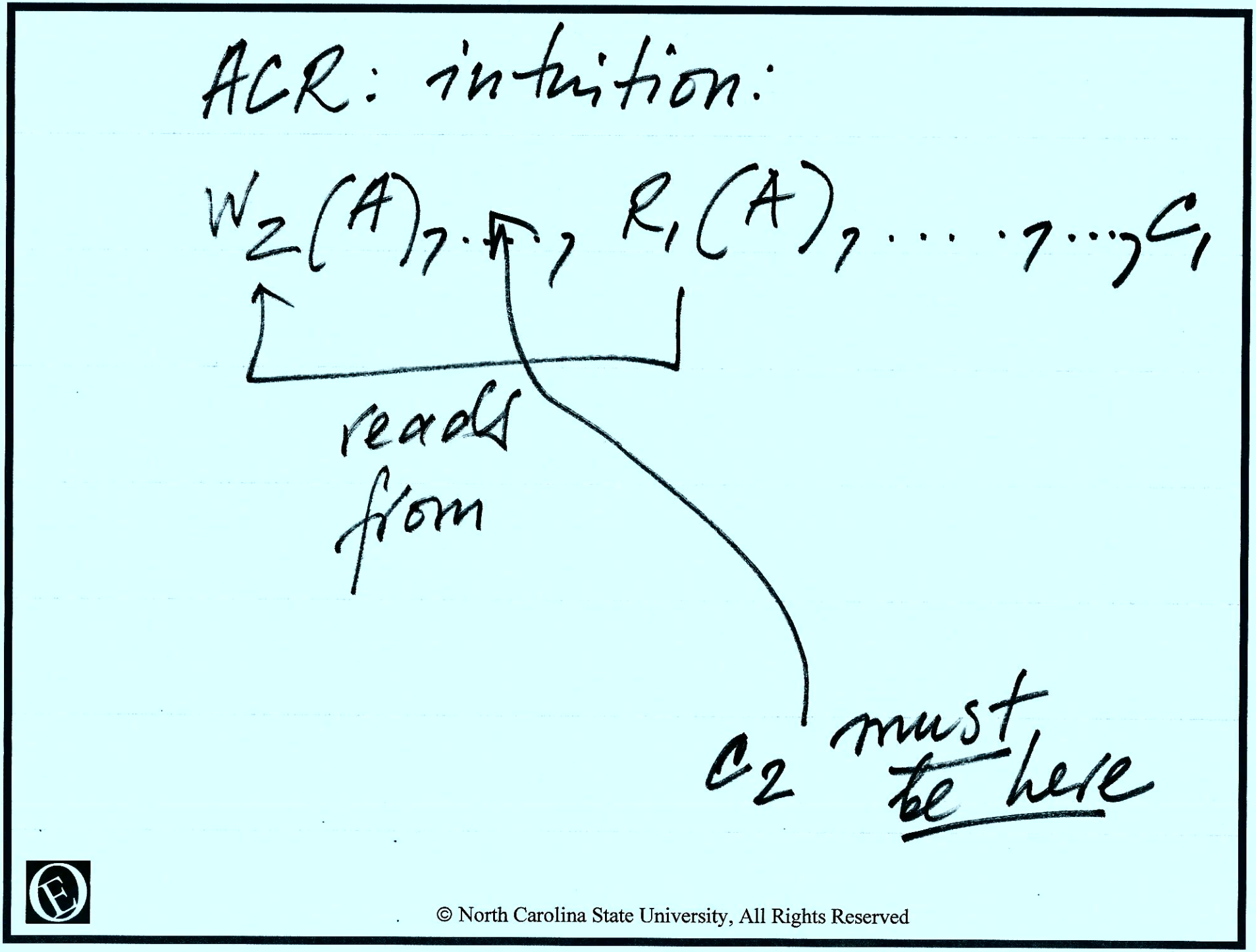
$\Rightarrow$ potentially "cascading rollbacks" if T_2 needs to be rolled back

Another example of recoverable S:

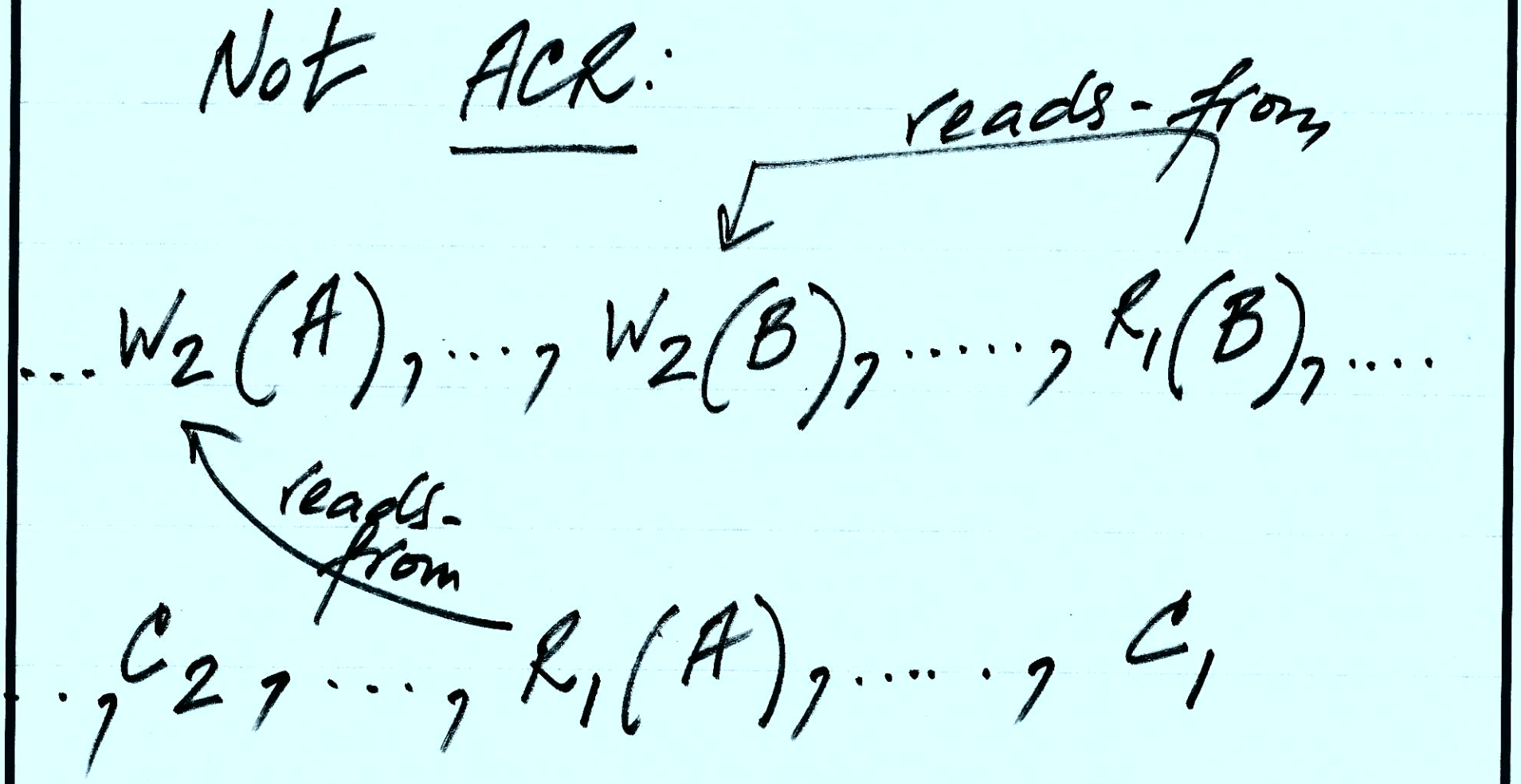


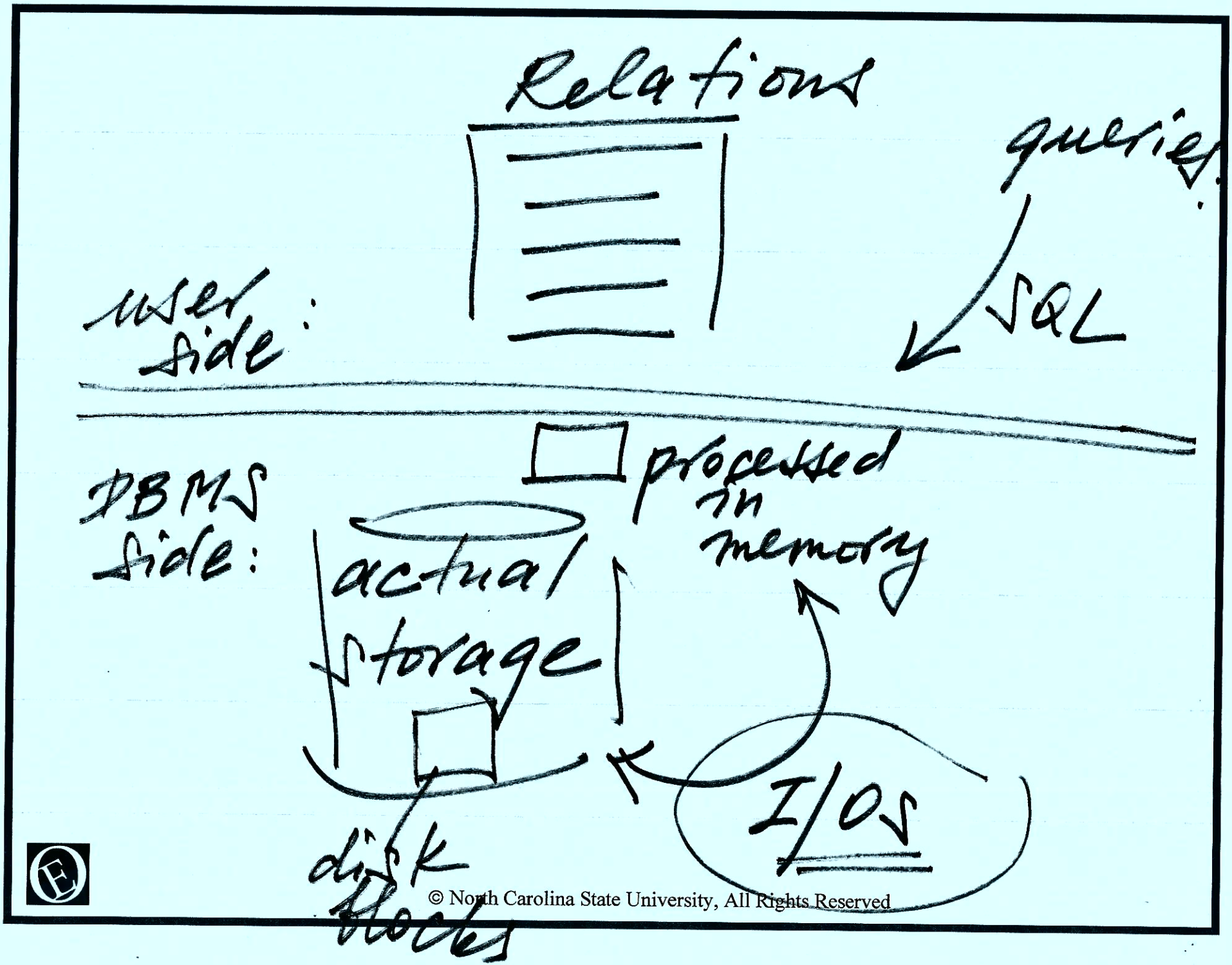
Avoids cascading rollbacks (ACR) schedule:

each T reads from a committed transaction



Not ACR:





Summary:

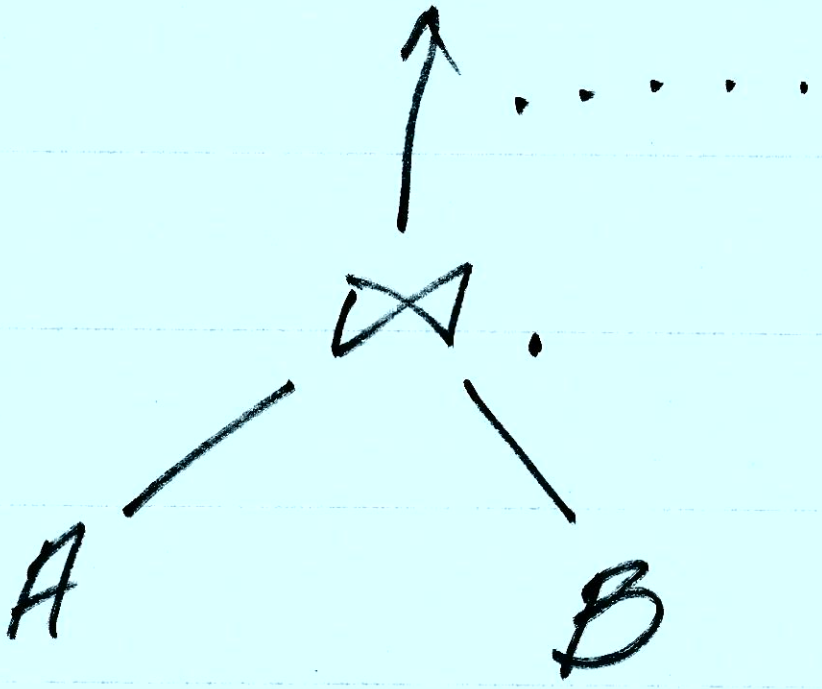
Translate SQL into relational algebra (because procedural), then optimize the resulting expression for your complexity measure

Intuition:

SQL stmt → mapped → default rel-alg (expression)

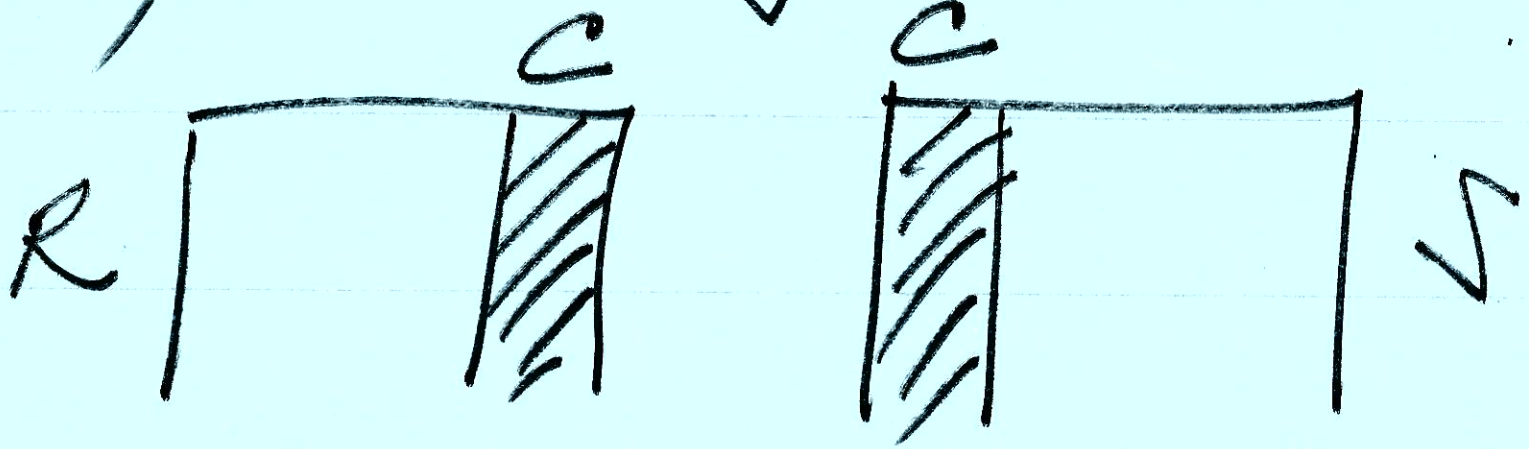
select # α, β, γ
from A, B
where C

$\pi_{\alpha,\beta,\gamma}$ σ_C × (A, B)



Natural join ⋈

(1) Hash join: $R \bowtie S$

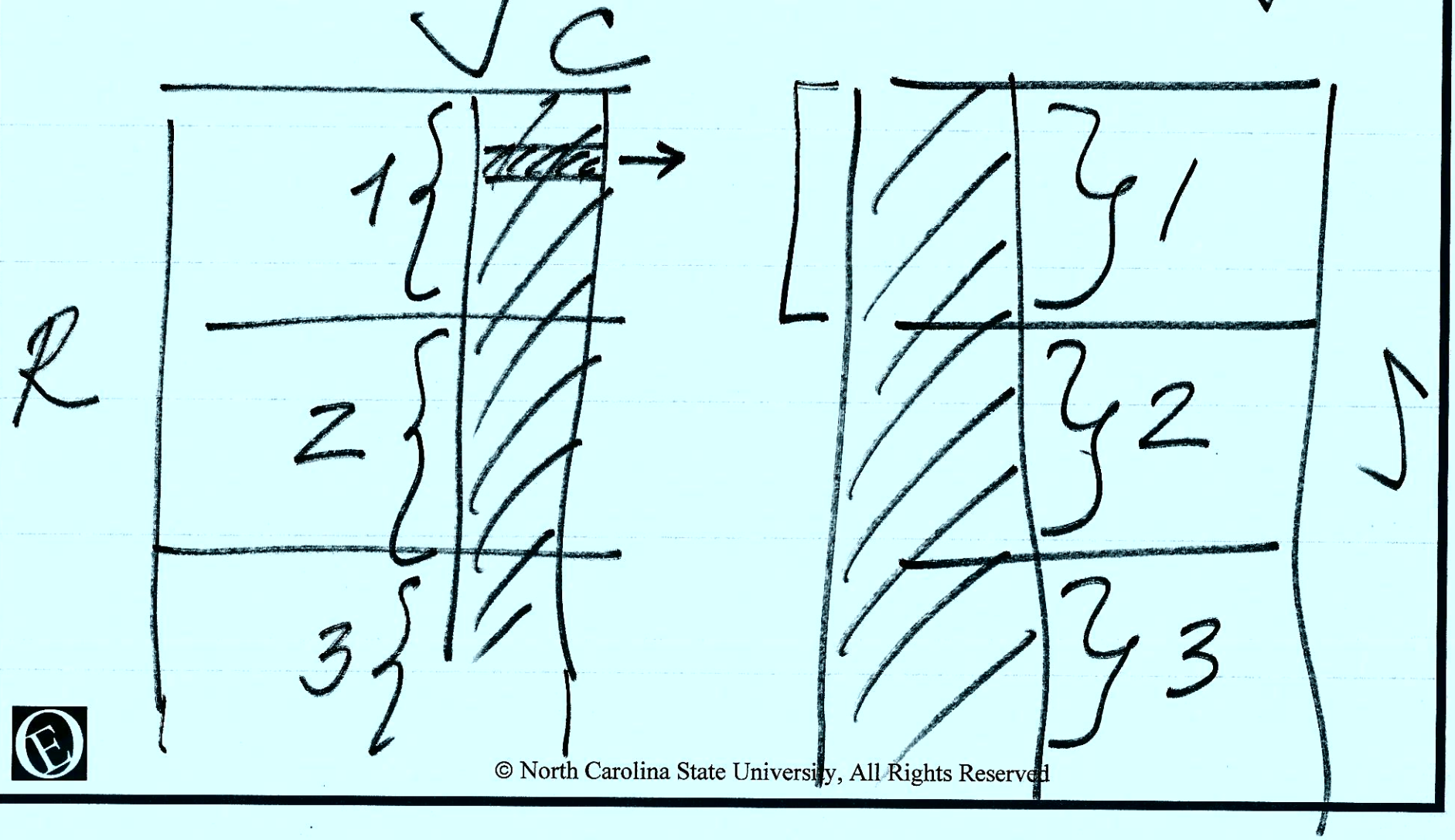


| c=1 | c=2 | c=3 | c=4 |

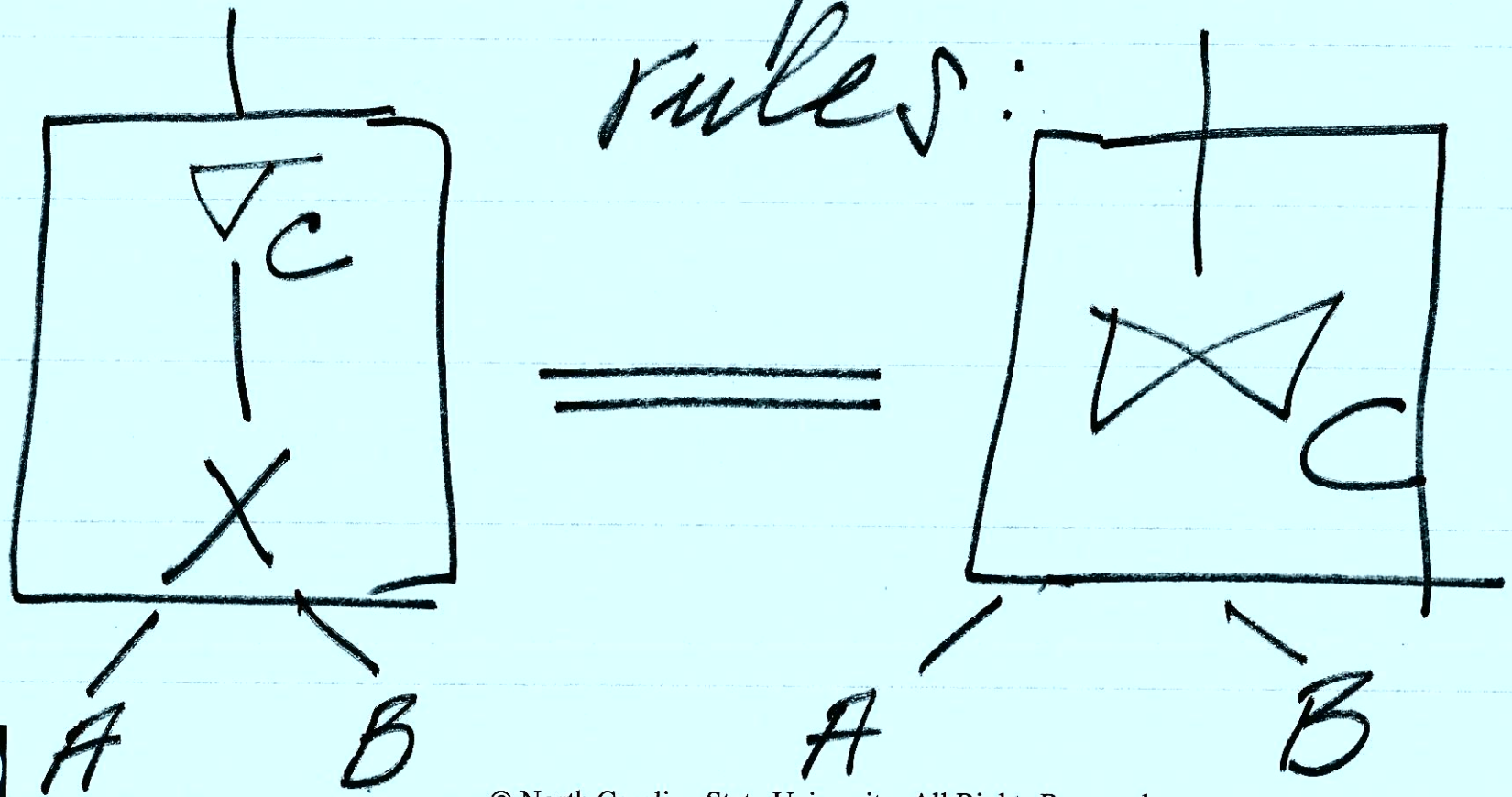
bucketize on C

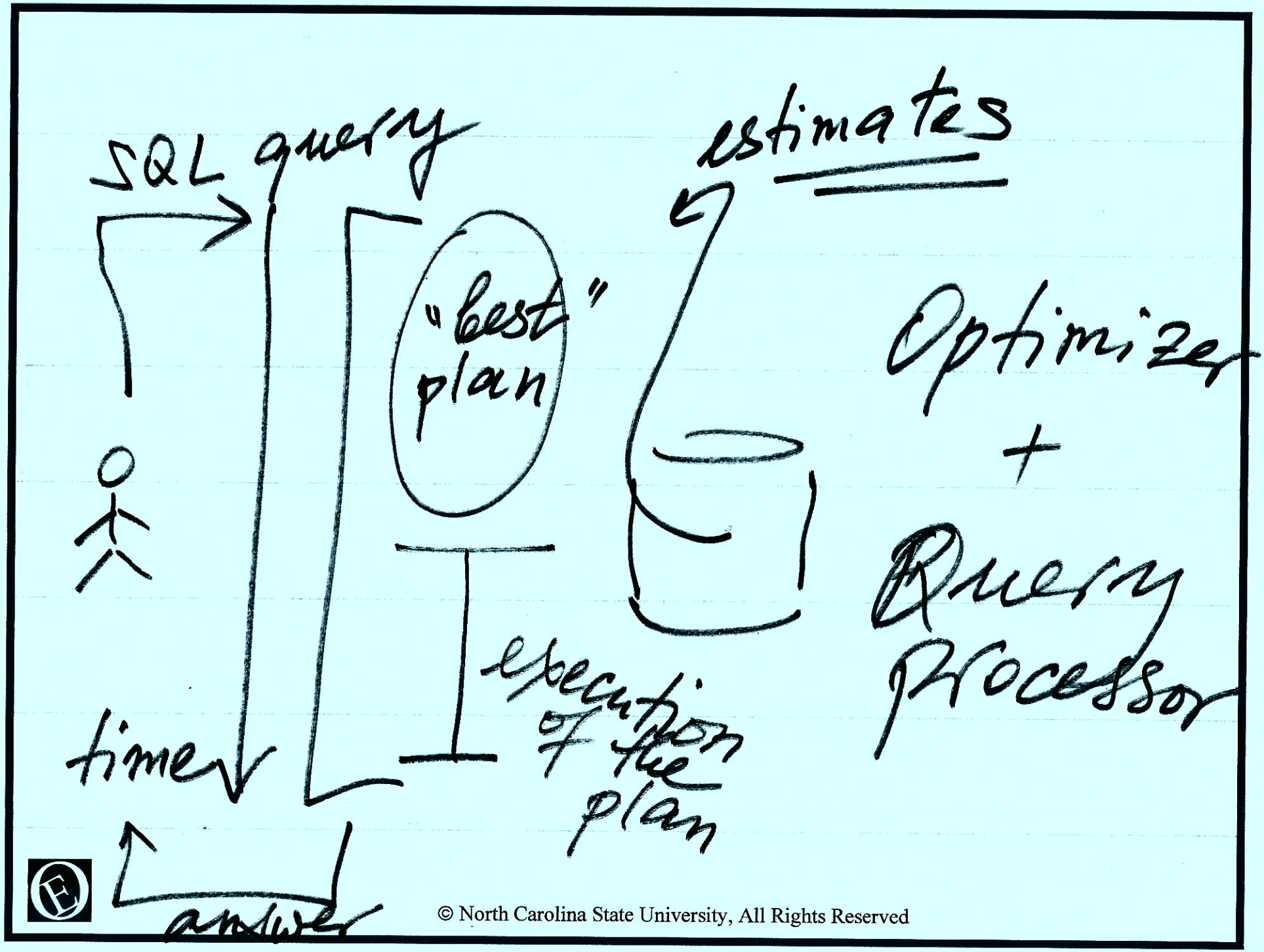
1 scan of R into memory

(2) Sort-merge join: $R \bowtie S$



Optimizers: transformation rules:





Keys to optimization:

(1) work with only necessary data

(2) process the data efficiently at each stage