1 Short Answer

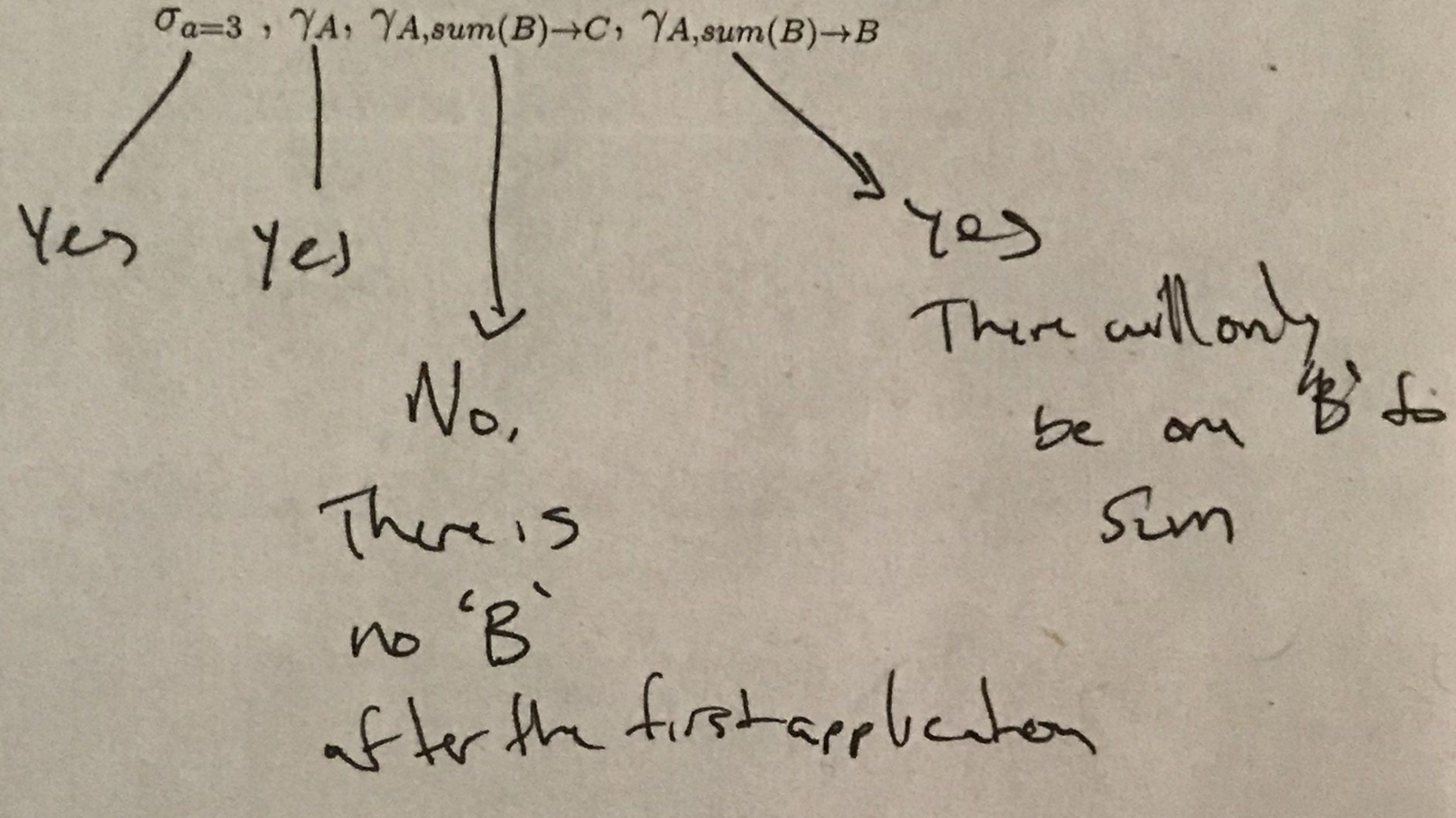
a) Given that strict 2PL is enforced in SQL, why might a dirty read still be possible?

Strict 2PL is always enforced for writing, but not always for reading, depending on the isolation level

b) In the SQLite locking scheme, at what point does the DBMS refuse to issue new read locks?

Once a transaction has been issued a pending lock

c) A unary operator is said to be *idempotent* if applying the same operator two or more times yields the same result as a single time. For example, repeated applications of the δ (duplicate elimination) operator yields the same result as a single application and therefore is idempotent. For each of the following operations, indicate whether or not they are idempotent



d) Give two SQL transactions that might suffer from the phantom problem.

T_1

Select * from A

T_2

Insert Into A

Select * from A

e) Given a relation R = $\{A, B, C, D, E, F\}$ and a set of functional dependencies, FD = $\{A \rightarrow B, D \rightarrow B, B \rightarrow F, D \rightarrow A, F \rightarrow C\}$ what is $\{A\}^+$

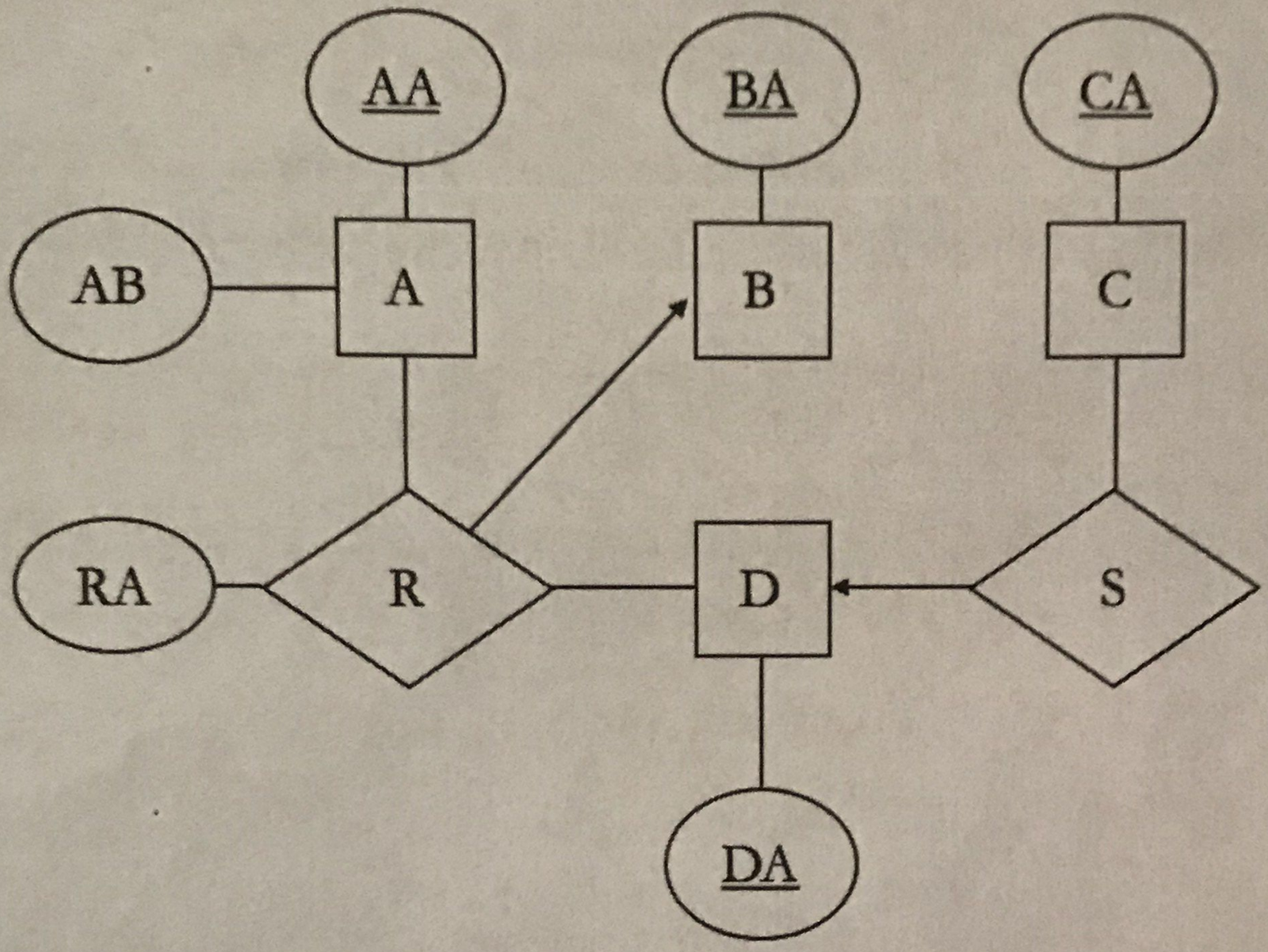
{A, B, F, C}

f) Why are some decompositions from BCNF excluded in 3NF?

If a decompositon breaks up an FD. it becomes mor difficult to enforce that FD.

2 E/R Diagrams

Produce a schema for the following E/R Diagram. Underline primary keys and circle foreign keys



A(AA, AB)

R(AA, DA, BA, RA) ← AA, DA Unique

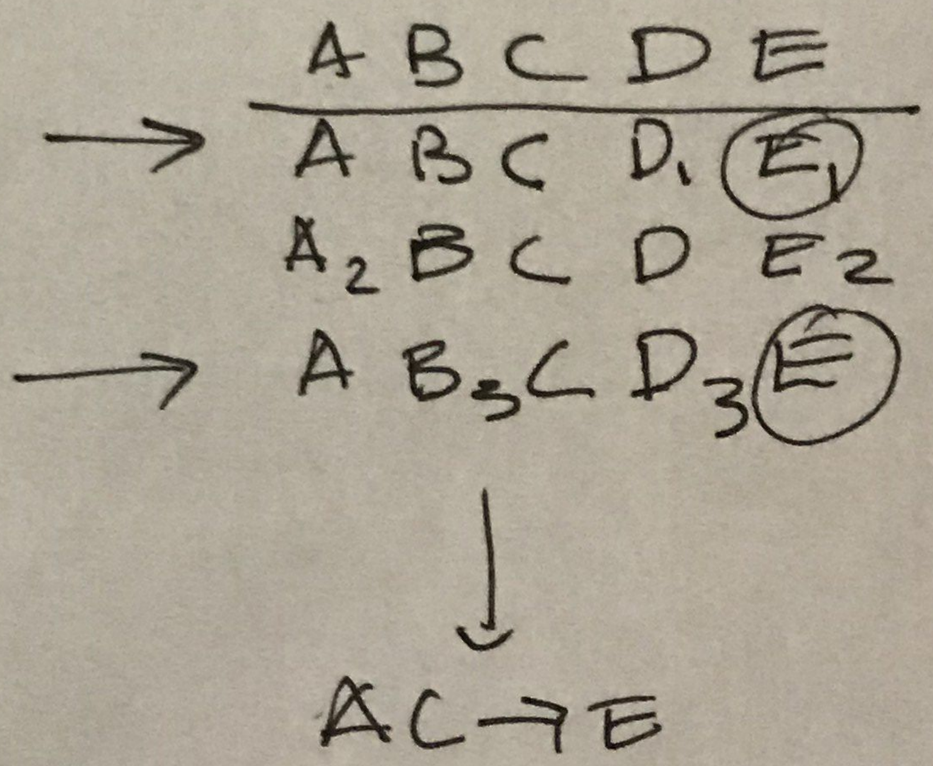
C(CA, DA)

D(PA)

3 Decompositions

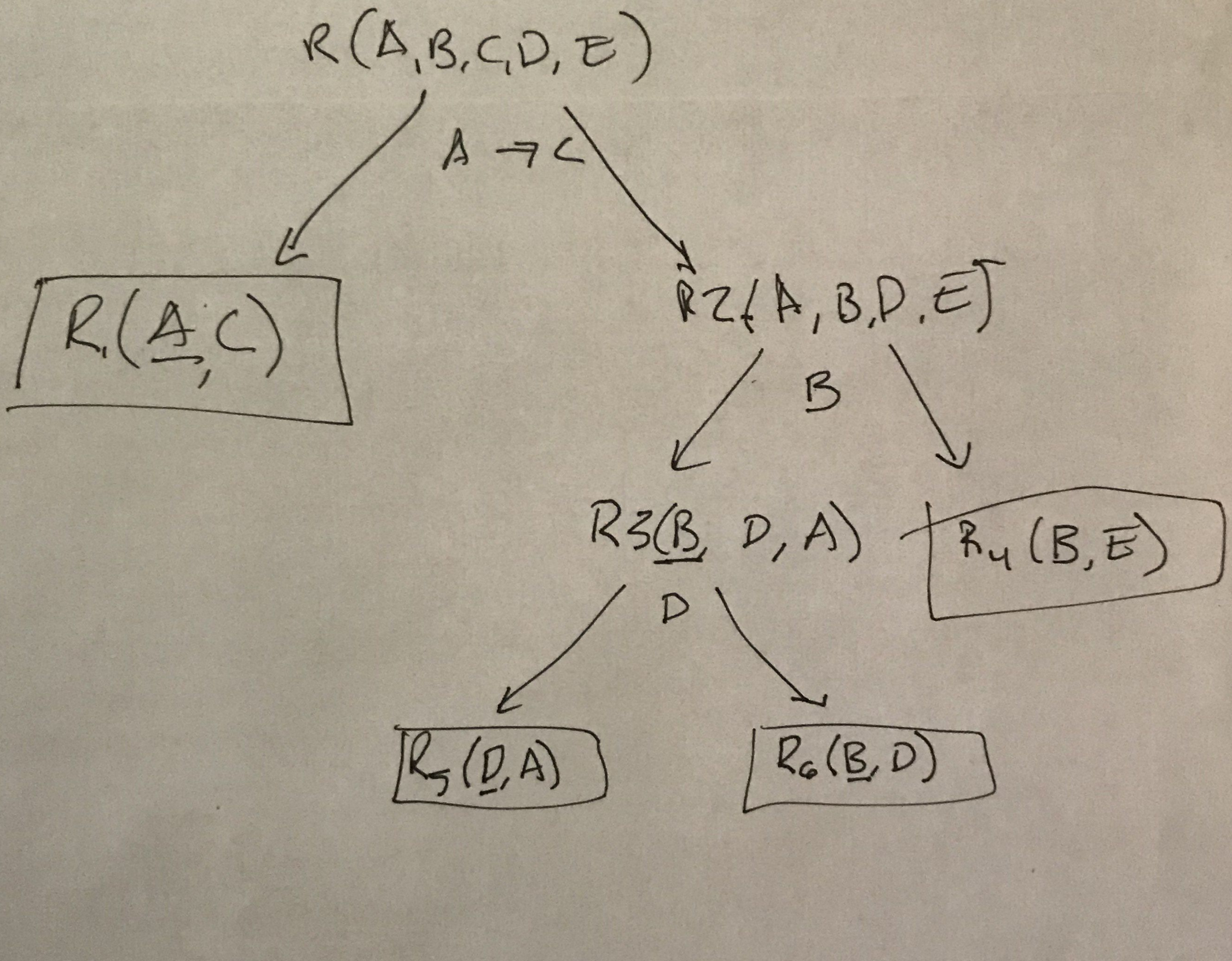
a) Let R(A,B,C,D,E) be decomposed into relations with the following three sets of attributes. $R1(A, B, C)$, $R2(B, C, D)$ and $R3(A, C, E)$. Use the chase test to tell whether the decomposition of R is lossless given the following functional dependencies. Show your work.

$$AC \rightarrow E \text{ and } BC \rightarrow D$$



b) Given the schema R(A,B,C,D,E), perform BCNF decomposition given the following functional dependencies. For each decomposition, give the functional dependency which violates BCNF.

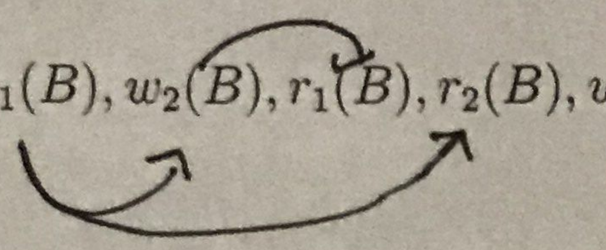
(a) $A \rightarrow C$, $B \rightarrow D$, $D \rightarrow A$

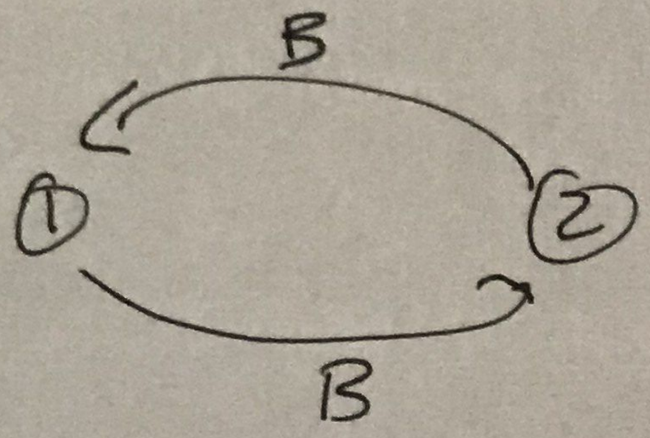


4 Transactions

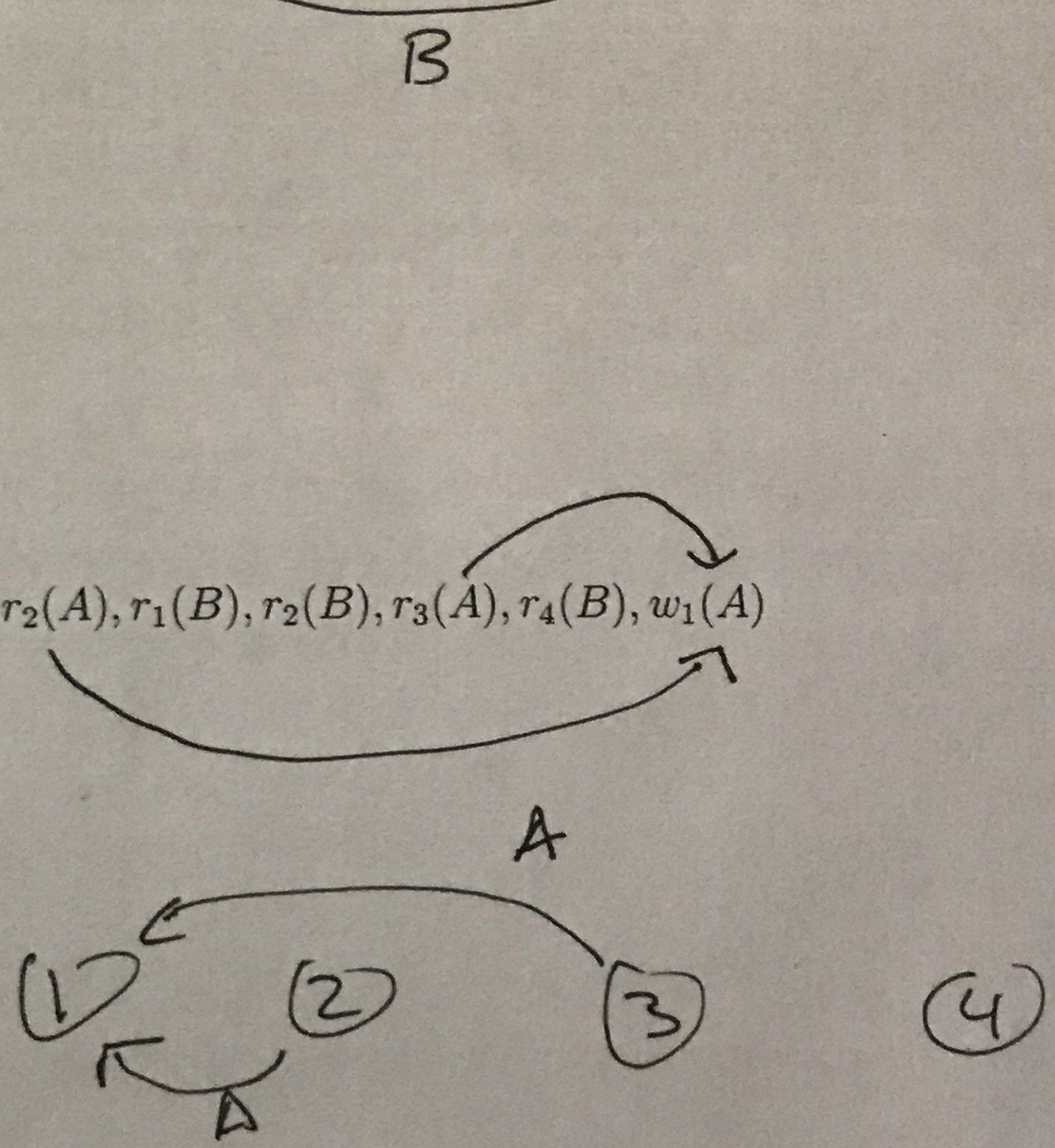
a) Draw the precedence graph for the following schedules. Indicate all resources that create a conflict for each edge. For example, if T_1 must come before T_2 because of both resources A and B, then indicate this. *Do not stop just because you have found a cycle*

(a) $r_1(A), r_2(A), w_1(B), w_2(B), r_1(B), r_2(B), w_2(C), w_1(D)$





(b) $r_1(A), r_2(A), r_1(B), r_2(B), r_3(A), r_4(B), w_1(A)$



b) Write two unique schedules from the following transactions that are conflict-serializable but not serial.

$T_1 : r(A), w(B), r(C)$ $T_2 : w(A), w(B), r(A)$

$r_1(A)\, w_1(B)\, w_2(A), r_1(C), w_2(B), r_2(A)$

$r_1(A)\, w_2(A)\, w_1(B), r_1(C), w_2(B), r_2(A)$

c) Given that there are two transactions i and j and three resources A,B,and C, fill in the following schedule with **all** single operations which make this schedule non-conflict-serializable.

$r_i(A), r_j(B),$ $w_i(B)$ $r_j(C)$ $w_j(C)$ $, w_i(C), w_j(A)$