```
Question for students with database implementation background
--------------------------------------------------------------

Consider the standard tuple-based nested-loop join algorithm for
computing T1 JOIN T2:

for each row R1 in T1:
  for each row R2 in T2:
    if R1,R2 satisfy the join condition then add R1/R2 to result

Suggest three separate possible improvements to this algorithm.
Assume a standard DBMS storage system and query processing context.
Improvements could depend on additional assumptions or scenarios.

ANSWER:

1. Nested-block join

Process T1 and T2 block-at-a-time instead of row-at-a-time.
Considerably reduces the number of times T2 is scanned, without
incurring extra I/O's for T1.
```

☆ "block at a time"

```
2. "Rocking"

For T2, scan it forwards the first time, backwards the second time,
forwards the third time, etc.  Takes advantage of LRU page replacement
policy typically used by database buffer managers.
```

forward - backward

last recent use

```
3. Use of keys

If the join condition is T1.A = T2.B and B is a key for T2, then once
a match is found the algorithm can break out of the inner loop.

4. Use of index

If the join condition is T1.A = T2.B and there is an index on T2.B
then the inner loop can find matching T2 rows using the index instead
of by scanning the whole relation.  (Also works for inequality join
conditions if the index is a B-tree.)
```