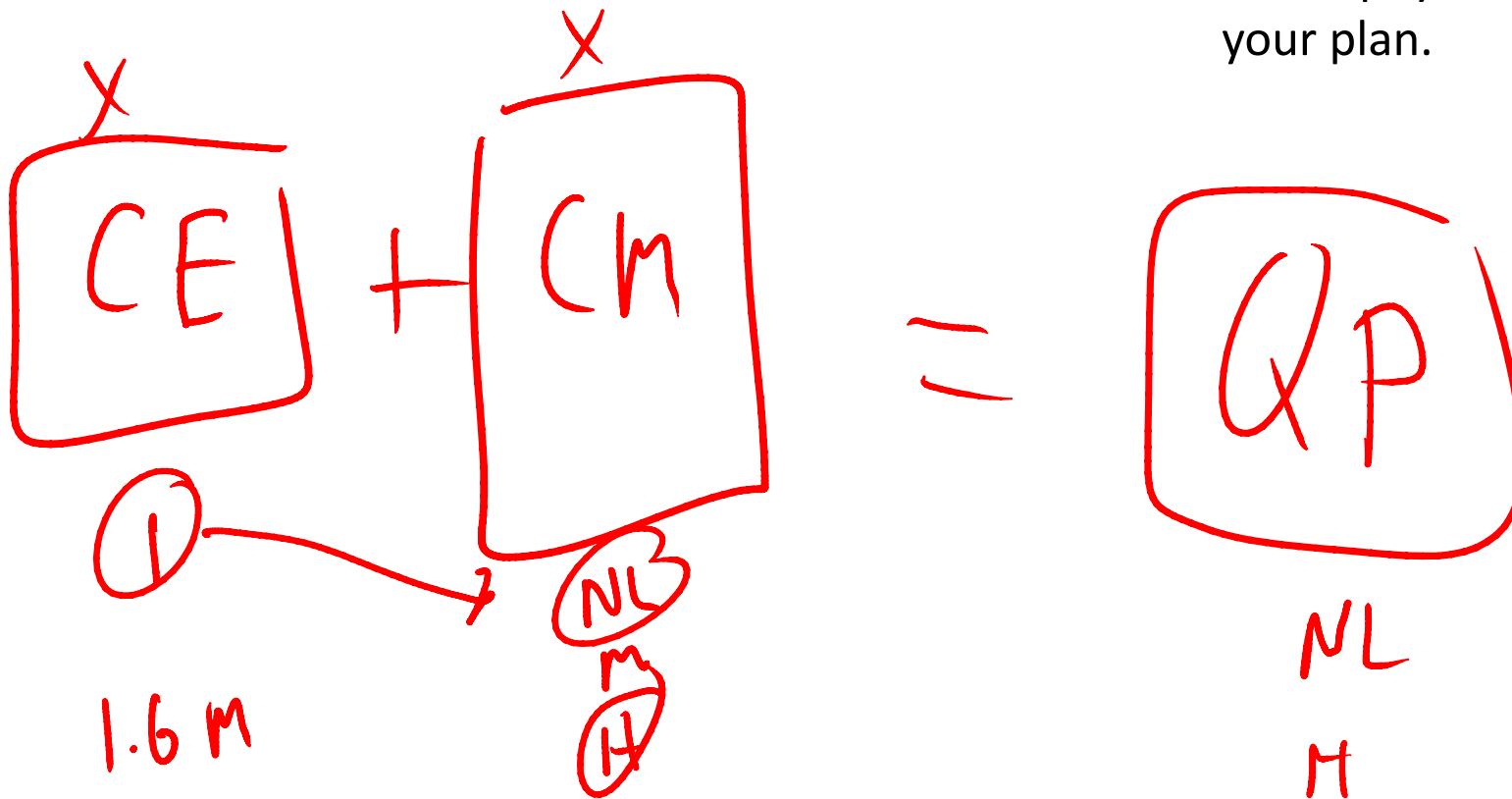
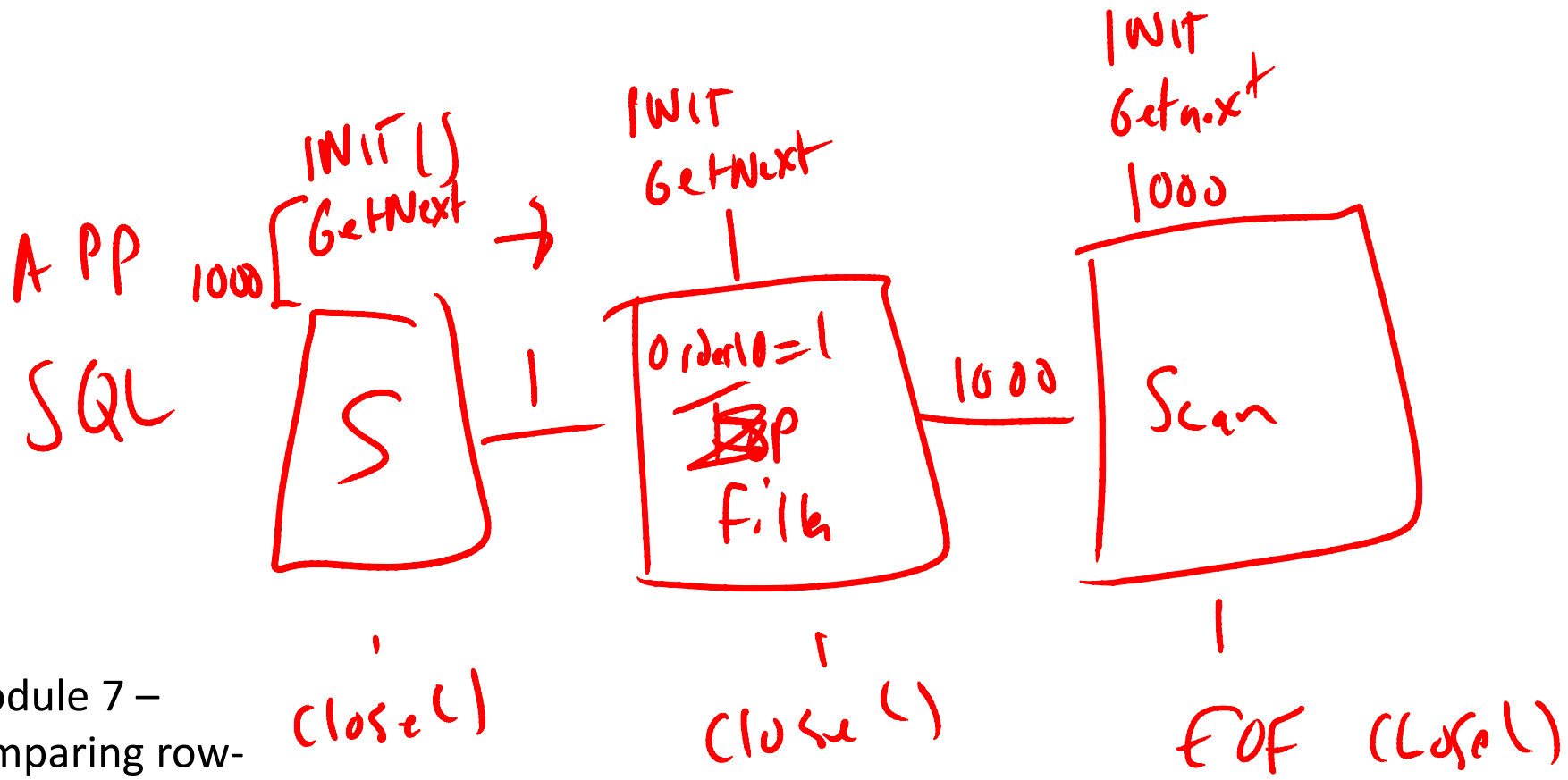


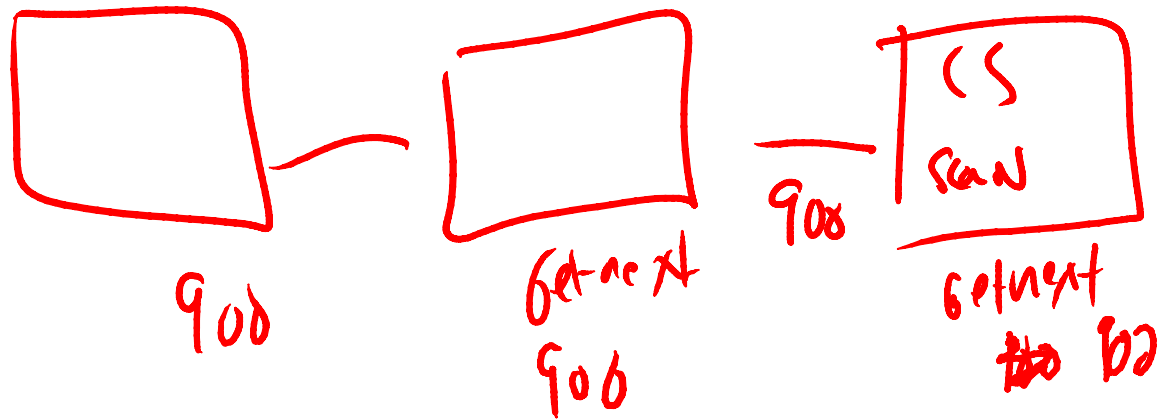
Module 7 – making the point that cardinality estimates are a key input to the cost model of the physical operators in your plan.

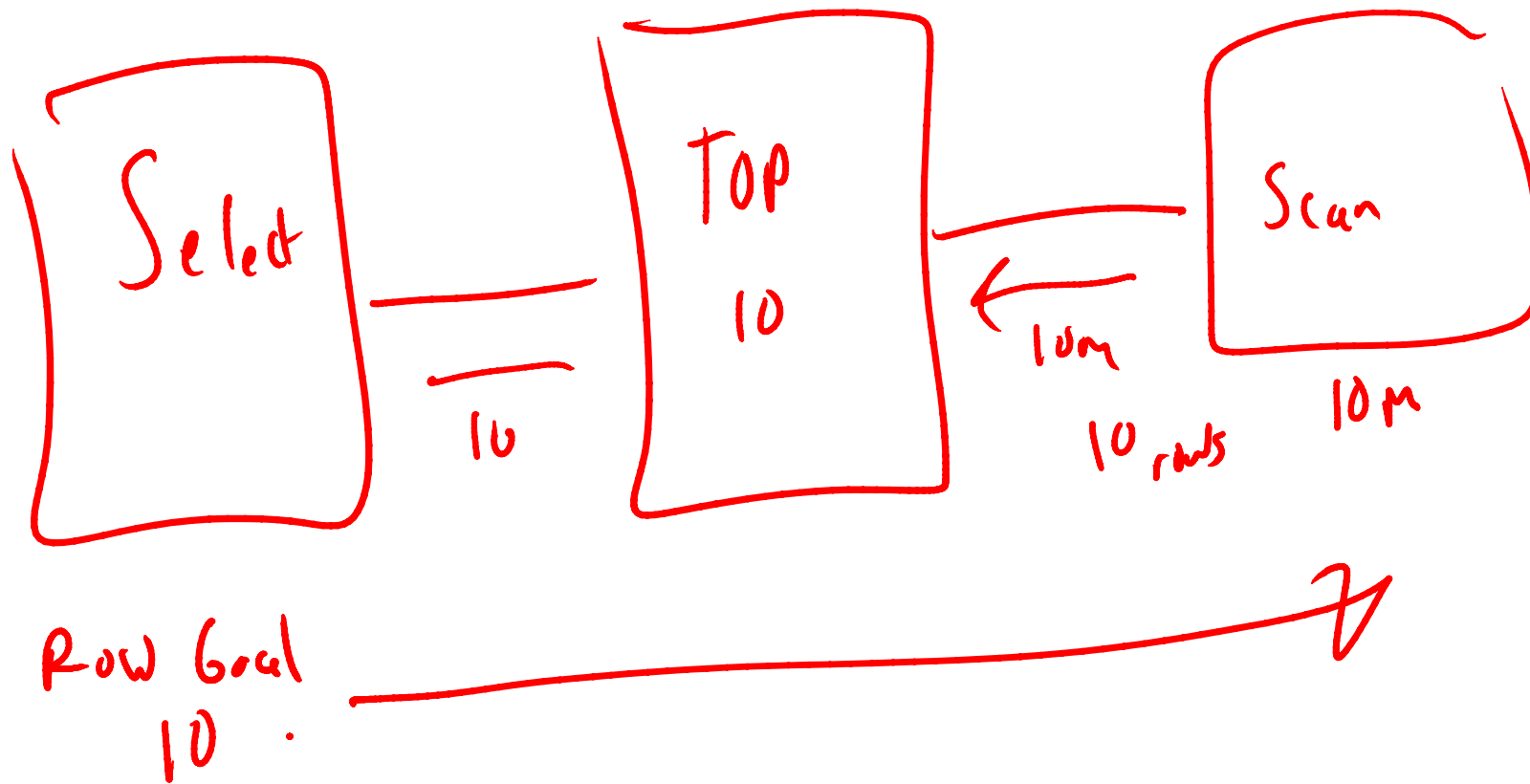


~



Module 7 –
comparing row-
execution mode vs.
columnstore index
batch execution
mode.

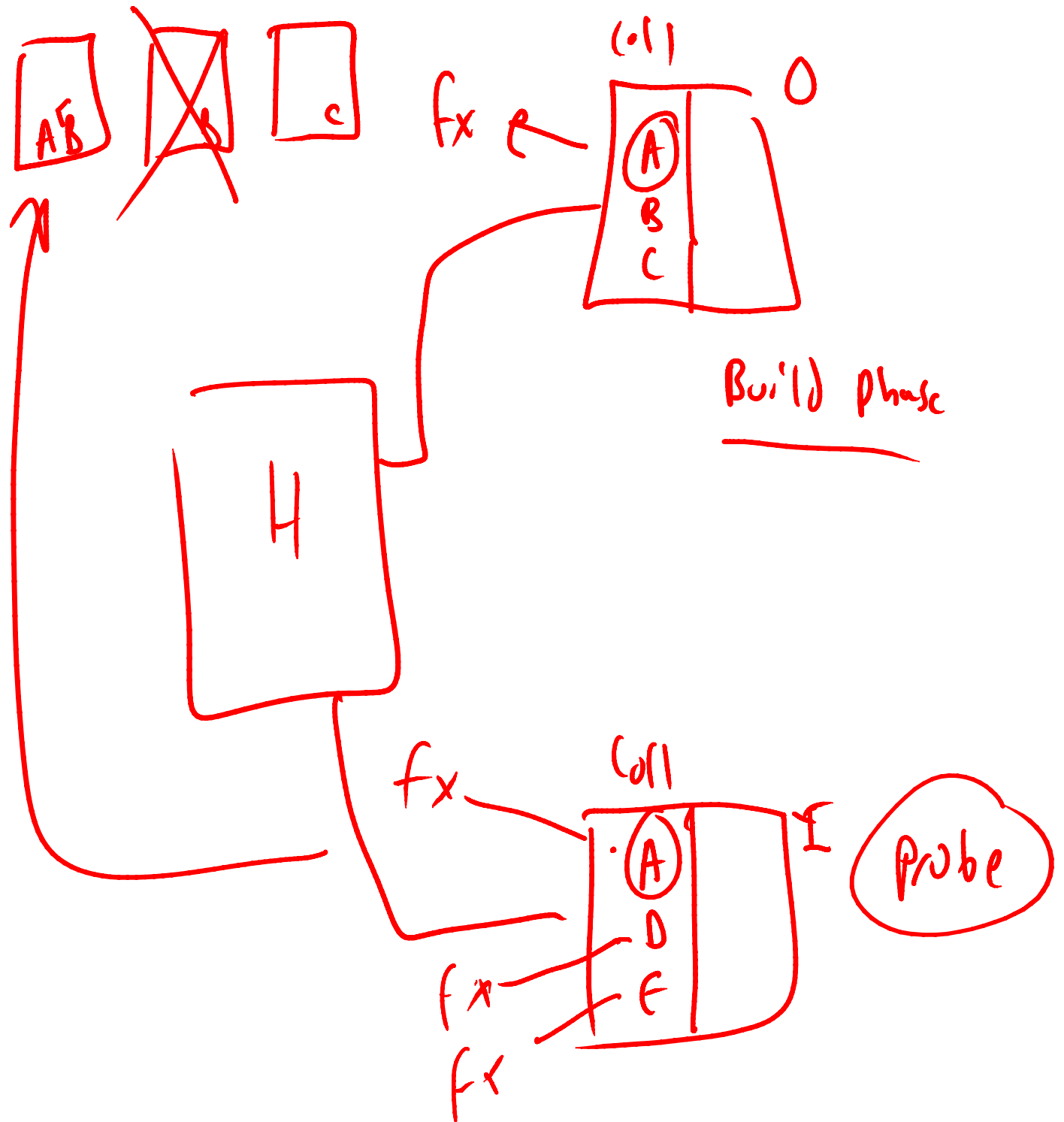


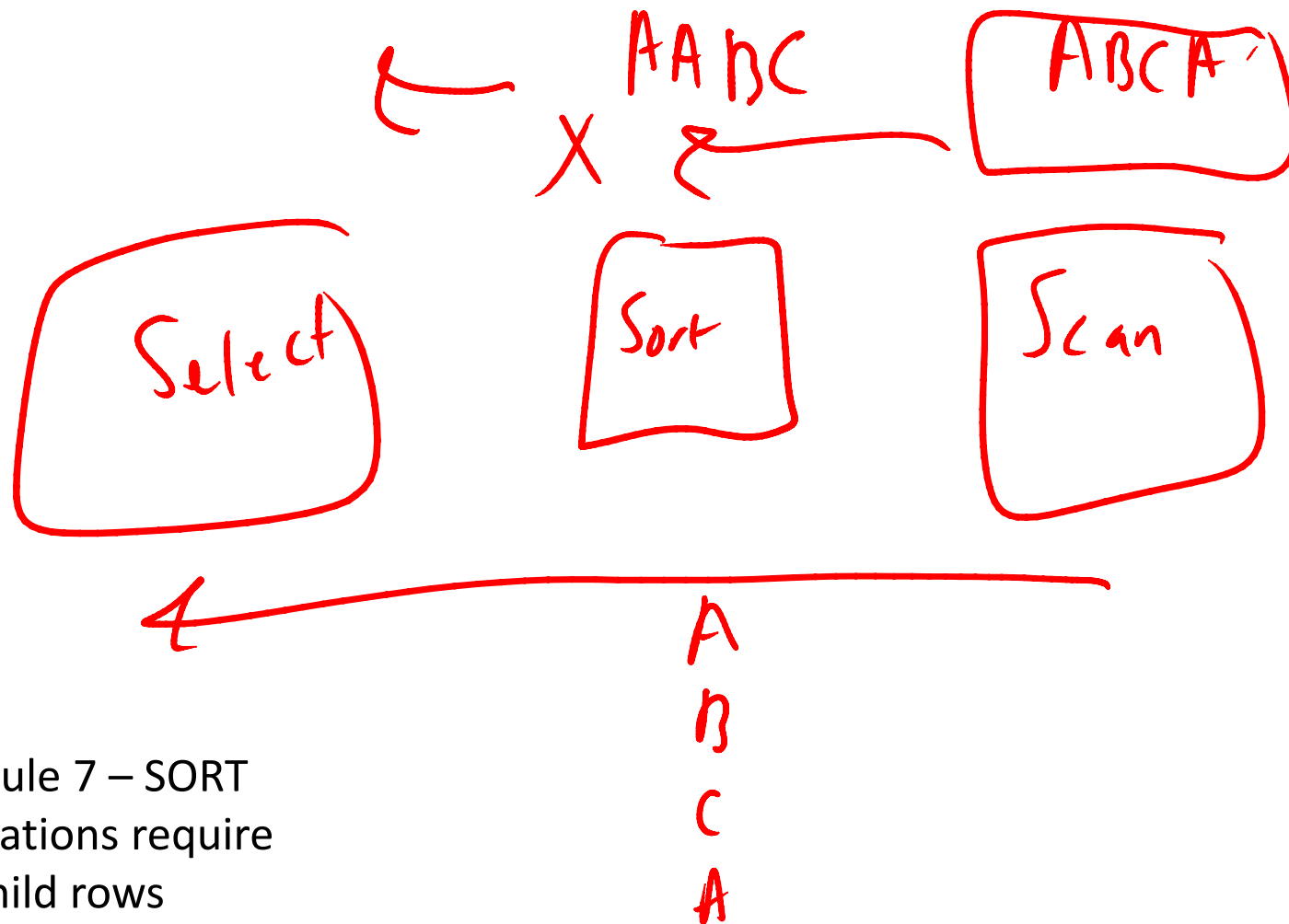


Module 7 – comparing control flow vs. data flow (the TOP shows that control flow moves from left to right on a graphical plan)

The diagram illustrates the operation of a Bloom filter in two phases:

- Build phase:** A set of elements $\{A, B, C\}$ is hashed by a function H . The results are stored in a bit array (represented as a table with indices 0, 1, 2). Element A is hashed to index 0, B to index 1, and C to index 2. The bit array is shown with bits 0, 1, and 2 set to 1.
- Probe:** To check if an element A is in the set, the same hash function H is applied. The resulting indices (0, 1, 2) are checked in the bit array. If all bits at these indices are 1, the element is likely in the set.

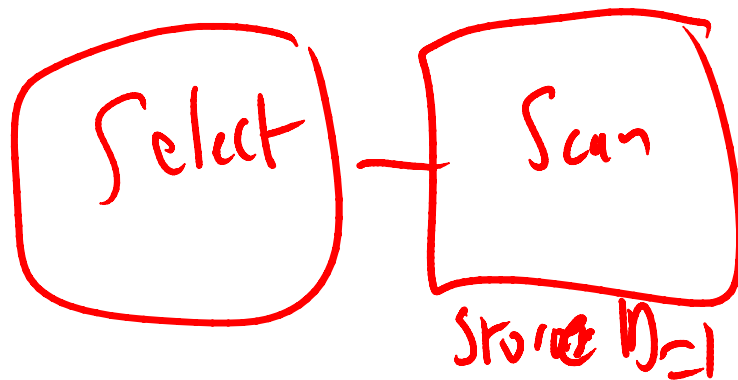
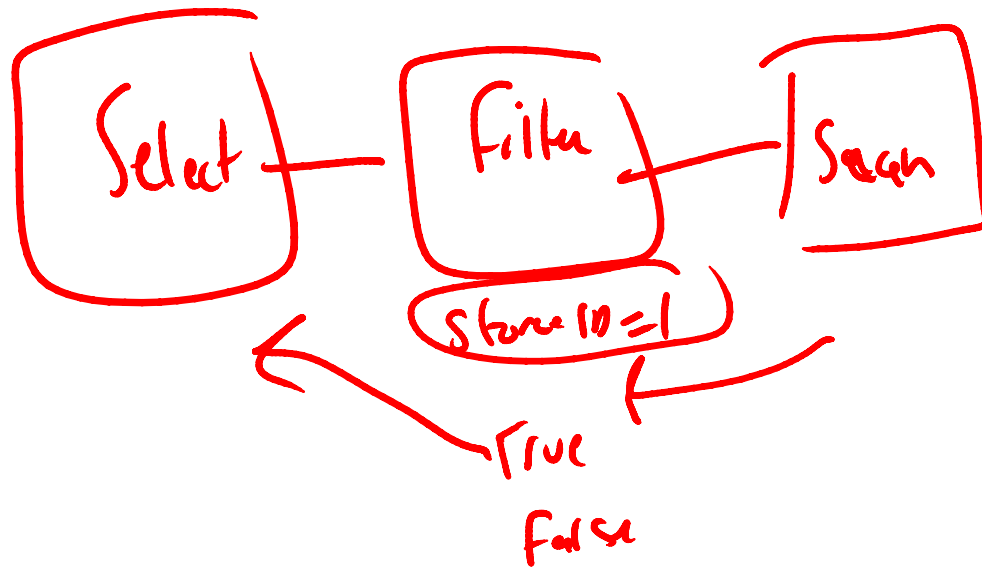




Module 7 – SORT
operations require
all child rows
before consecutive
rows flow to the
parent operator.

Where StoreID = 1

Stores table



Latch page

Row

~~Evaluate~~

Push row up

Release latch

Evaluate Row

Repeat

Latch page

eval row

True False row

row

Release latch

Module 7 – Pushed predicates can help avoid additional page latch operations per row, as the predicate is evaluated at the SE level and non-matching rows don't flow up the tree.

Module 7 – bitmap operators are an additional optimization that leverages the stop-and-go nature of the build-phase of a hash operation – taking the bitmap and then pushing it prior to the probe phase.

