Iterators:

- Many operations access only one tuple at a time.
  - read tuple.
  - inspect
  - dispose
  - read next tuple.

Open() – initiates the process

GetNext() – return next tuple

close() – ends process

Example:

$\Pi_a \sigma_{b=3} R$

Π_a

| on the fly

$\sigma_{b=3}$

| seq scan of R

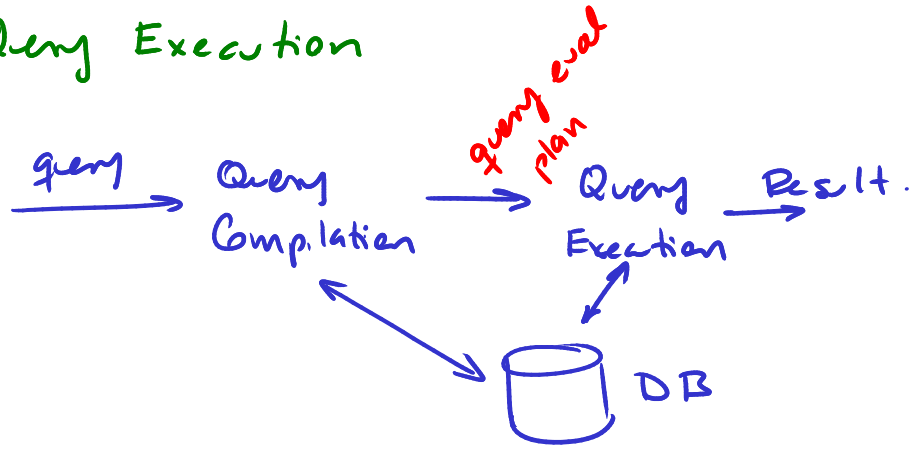
R

Π and σ can be implemented as iterators

σ inspects one tuple at a time, sends one tuple at a time to Π

No need to store any tuple in memory

Query Execution



Query Compilation

a) Parsing. A parse tree is constructed
   - Create an algebraic expression.

b) Query Rewrite:
   - Several equivalent query expression

c) Physical plan generation
   - Each expression is converted to an evaluation plan by indicating the alg. to use.

b) and c) are the query optimizer

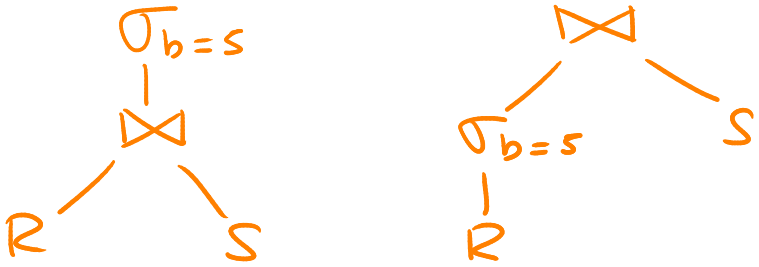
⇒ find best query plan:

1) Which algebraic expression is the one leading to the most efficient alg.
2) For each operation in the expression which alg. will be used to answer it.
3) How should each operation pass data to the next operation.
4) How are the relations going to be accessed.

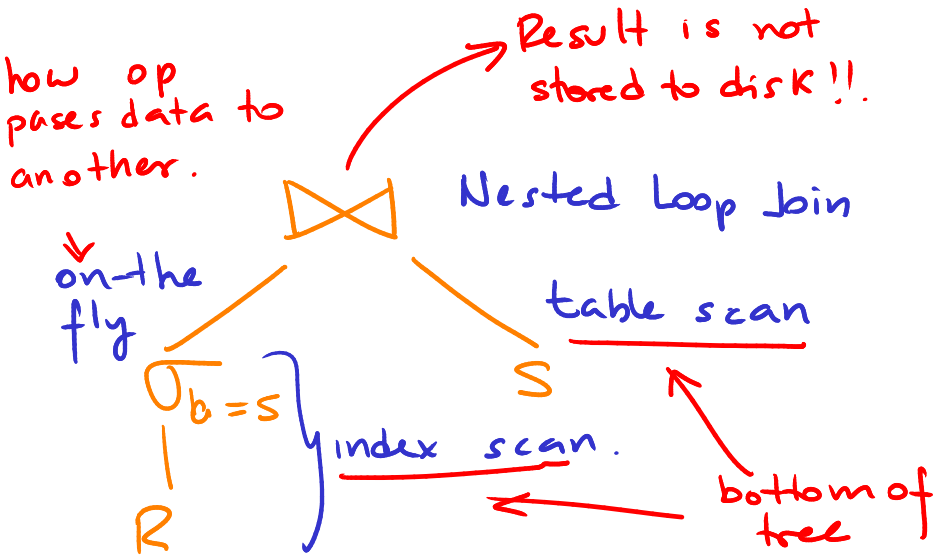
Ex: R(a,b) S(a,c)

SELECT * from R natural join S
WHERE b = 5

Equivalent Expressions



Annotate tree with algorithms and access methods



Estimate cost.
⇒ choose fastest!

Access to tuples:
- Sequential scan of heap of Rel.

or
- Using an index to scan a subset of tuples of R (index scan)

Result of query:
- Kept in memory.