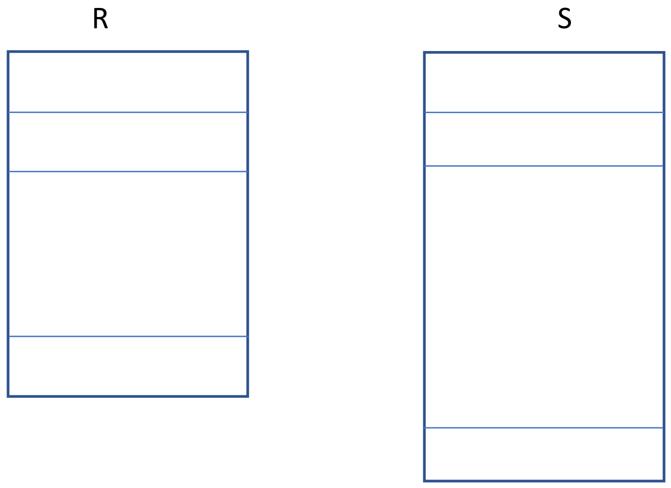
# Some notes on join processing

How does a DBMS evaluate a join query?



$R \bowtie S$

Suppose R has M pages and S has N pages.

Naïve solution: Nested loop join

```
For every row $r \in R$
    For every row $s \in S$
        Join rows r and s
```

Cost:

- I/Os: $M + MN$
- Buffer pages: 3

If the buffer has enough space to store the whole relation R, then

Cost:

- I/Os: M + N
- Buffer pages: M+2

If 100 buffer pages can be allocated for the join and M>100, then
Cost:
I/Os: M + $\left\lceil \frac{M}{98} \right\rceil$N
Buffer pages: 100

In these joins, we call R the outer table and S the inner table (in the nested loop join).

Question. Given two tables of different sizes to be joined, which one should be the outer table and which one should be the inner table?

Now suppose there is an index on the join column of S. What will be an efficientjoin algorithm?

```
Nested loop join using the index on S
  For every row r ∈ R
     Search the index on S using the join column of r; let the search returns a set of rows s
       Join rows r and s
```

Cost: Suppose each page holds 100 rows, and the index is hash where each search takes 1 I/Os.
I/Os: M + 100M*1
Buffer pages: 3