# Query Processing

Given a query,

```
SELECT B.manufacturer
FROM Beers B, Sells S
WHERE B.name = S.beer
  AND S.price < 20
```

We execute it in two levels:

① Logical Plan: Relational Alg., many variations exist

② Physical Plan: Implementation of Logical Plan

## Implementations of Operations

### 1. SELECTION

① Logical Level: $\sigma_{subject = "DB"}$ (Textbook)

② Physical Level
- Table Scan: Read R block by block
- Index Scan: Use Index to find blocks (efficient)

### 2. JOIN

Can be extended to implement other operators

① Logical Level: $R \bowtie S$

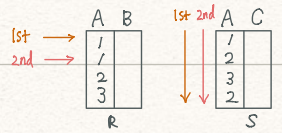
② Physical Level:

Internal Memory Join

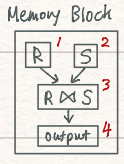
#### Case 1. Nested Loop Join

↳ If both R and S can fit into the main memory

↳ Time: $O(|R||S|)$




Basically, for $i$ in $R(A)$ for $j$ in $S(A)$ — Nested for Loop




↳ Memory Requirement: 4

↳ Cost: $B(R) + |R| B(S)$ — Improved!

Read blockwise: $B(R) + B(R) \cdot B(S)$ — Improved w/ bigger M

If M = 200 (large), You can fill 197 blocks w/ R and join w/ S

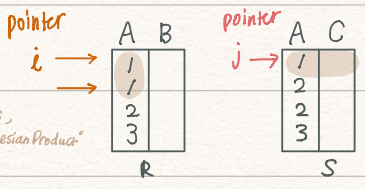
$$B(R) + \left[\frac{B(R)}{M-2}\right] B(S) \approx \frac{B(R)B(S)}{M}$$

M-2 because 2 blocks for output

#### Case 2. Sort-Merge Join

↳ If both R and S can fit into the main memory

↳ $O(|R| \log |R|)$ if $|R| > |S|$

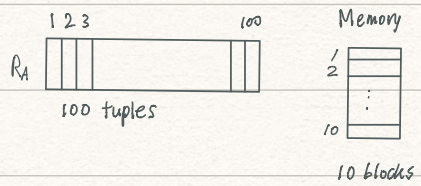



for duplicates, we use "Cartesian Product"

Since R(A) and S(A) are sorted, we can search and match.

External Memory Join

* Problem: how to sort R if it's too big to fit in memory?

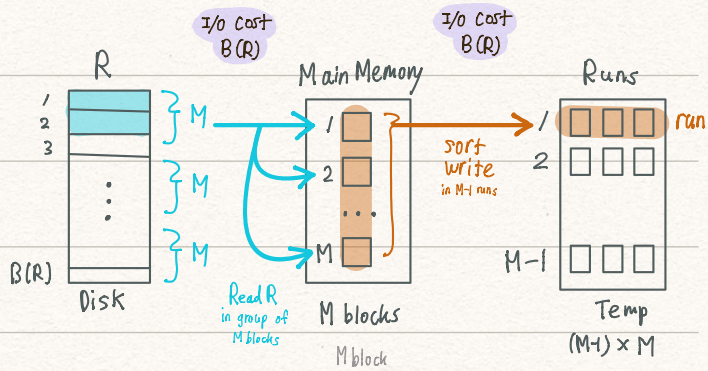



#### Case 3. Two Pass (Phase) Multiway Merge Sort

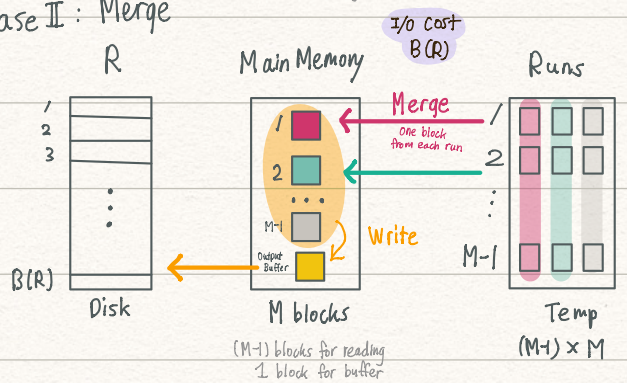
↳ At least one relation doesn't fit into main memory.

↳ Dominant cost is I/O access

- Phase I: Read & Sort




- Phase II: Merge



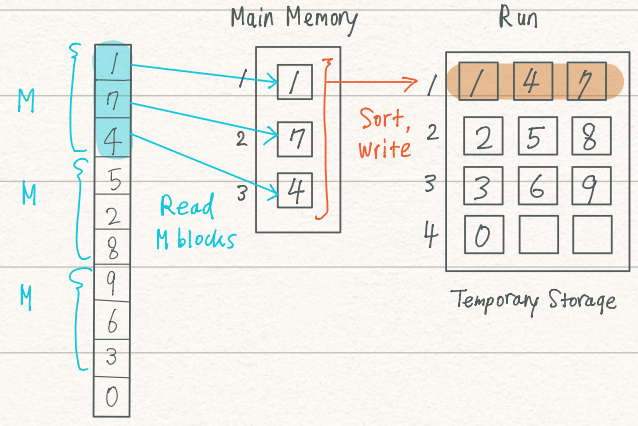

↳ cost: $2B(R)$ in Phase I + $B(R)$ in Phase II $\rightarrow 3B(R)$

↳ Limitation: $B(R) \leq M(M-1)$ or $B(R) \leq M^2$

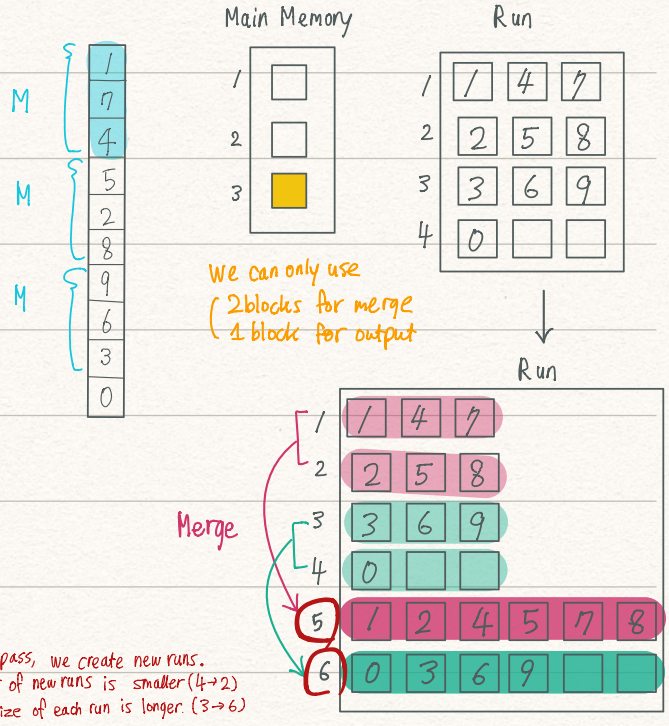
## case 4. General Multipass Multiway Merge Sort

ex> M : 3 blocks, each block holding one number

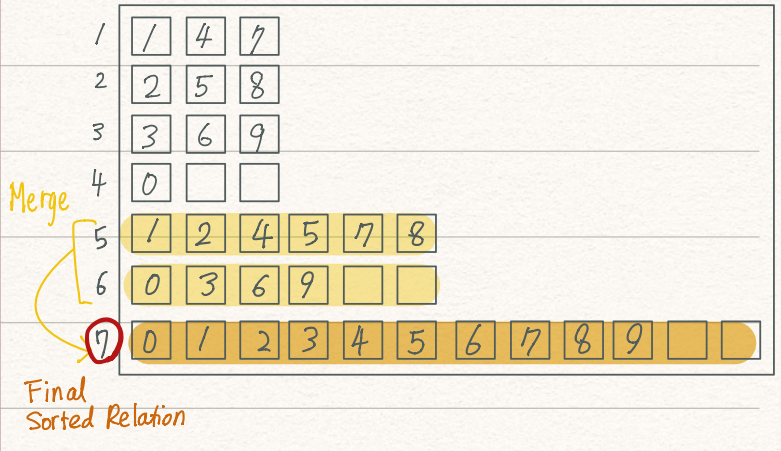
- Phase I (Pass 0) : Read & Sort




- Phase II (Pass 1) : Merge




- Phase III (Pass 2) : Merge




---

Can handle more than $B(R) > M^2$

The key to apply Multipass Multiway Merge Sort is

① For each pass, merge as much as you can w/ given M

② Create new runs (Merge & Sort) w/ each run # runs ↓, |runs| ↑

③ Finish ② when everything is sorted.

④ Return the one sorted relation.

---

## Generalization of MMMS :

- Pass 0 : Read a relation, R, in group of M blocks & Sort them
  - ↳ $\lceil \frac{B(R)}{M} \rceil$ runs in level 0.
- Pass k : Merge (M−1) runs in level(k−1) & write them in level k
  - why? (M−1) blocks for input, 1 block for output
  - ↳ In level k, the number of runs = $\frac{\text{The number of runs in level (k-1)}}{(M-1)}$
- Final Pass : results in one sorted run.

---

## Analysis of MMMS :

- # of Passes : $\lceil \log_{M-1} \lceil \frac{B(R)}{M} \rceil \rceil + 1$
  - pass 0 for reading R
- Cost
  - In each pass, we read and write R
    - ↳ # of passes · 2B(R)
  - In the final pass, we simply write, so disregard
  - ∴ cost of MMMS = $O(B(R) \cdot \log_M B(R))$
- Memory Requirement : M