Query Execution

Presenter: Soo Yee Lim

Discussion Lead: Xuechun Cao

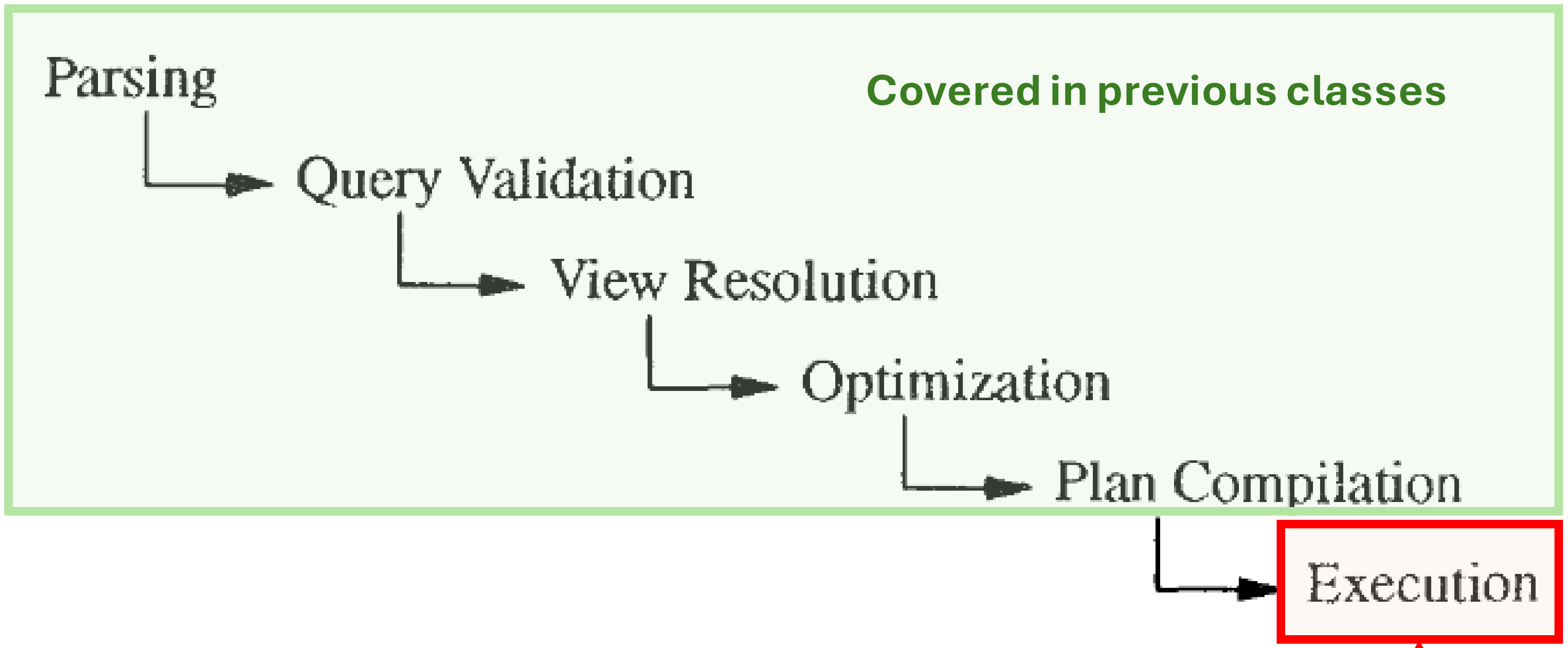


Figure 2. Query processing steps.

Our focus today!

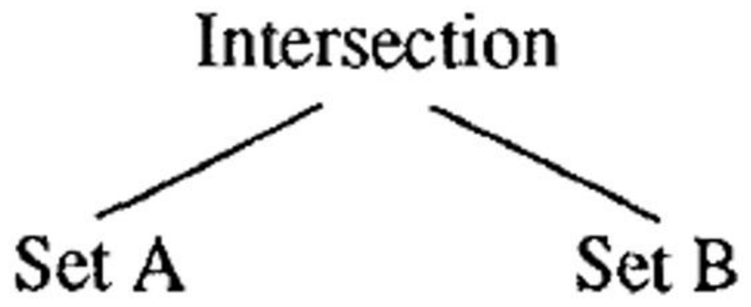
Discussion #1 – Group of 3

The author states that DBMSs have not been used in many areas for two reasons:

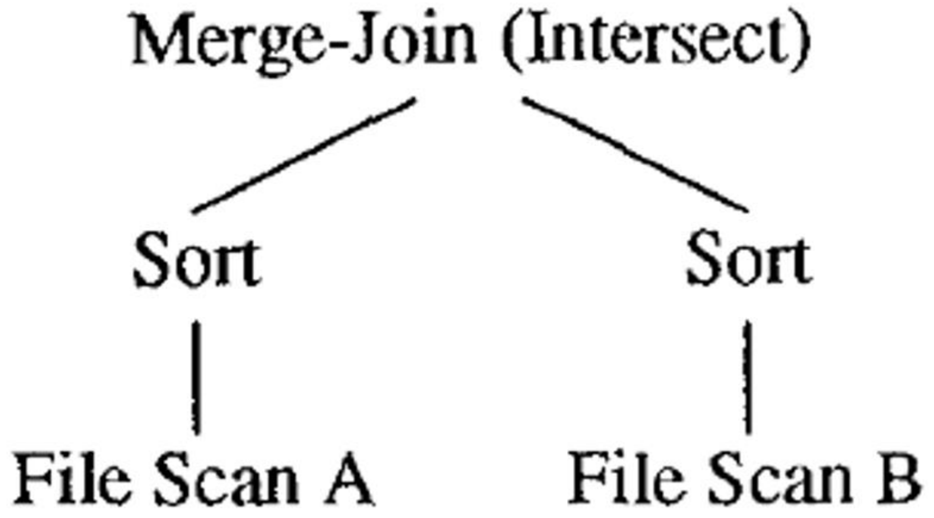
- application development and maintenance is difficult.
- the data in those areas is SO big, that speed trumps all, and people would rather hand-code.

Why do you think databases aren't used more? Why don't "you" use them on "your" data?

Logical Algebra



Physical Algebra



Logical Algebra

- Closely related to data model.
- Defines what queries can be expressed in the data model.
- E.g., relational algebra

Physical Algebra

- Query processing algorithms.
- Can vary across systems.
- Cost functions are associated only with physical operators.

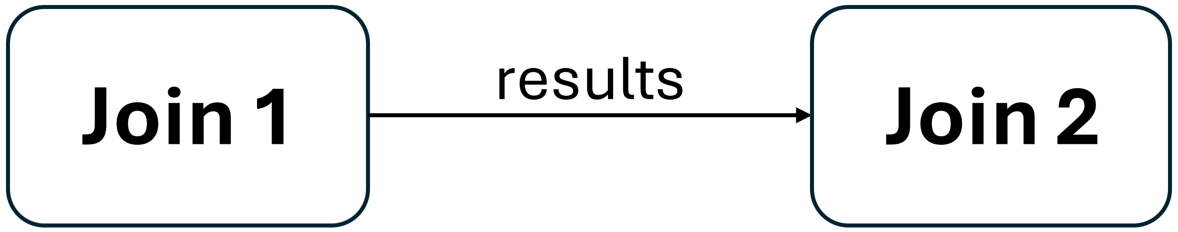
Logical -> Physical Mapping

- Why?
  - Due to the lack of algorithm specification, logical algebra expressions is not directly executable.
- Why is it non-trivial?
  - Involves algorithmic choices.
  - Logical and physical operators often do not map directly into one another.

DBMS Overview

- Query Optimization
  - Mapping from logical to physical operations.
- Query Execution Engine
  - Implementation of operations on physical representation types (e.g., file).
  - Implementation of mechanisms for coordination and cooperation among multiple operations in a complex query.

Synchronization & Data Transfer



Synchronization & Data Transfer

Temporary Files

- Execute Join 1 completely before starting Join 2.
- No pipelining, hence, performance loss.

Pipes

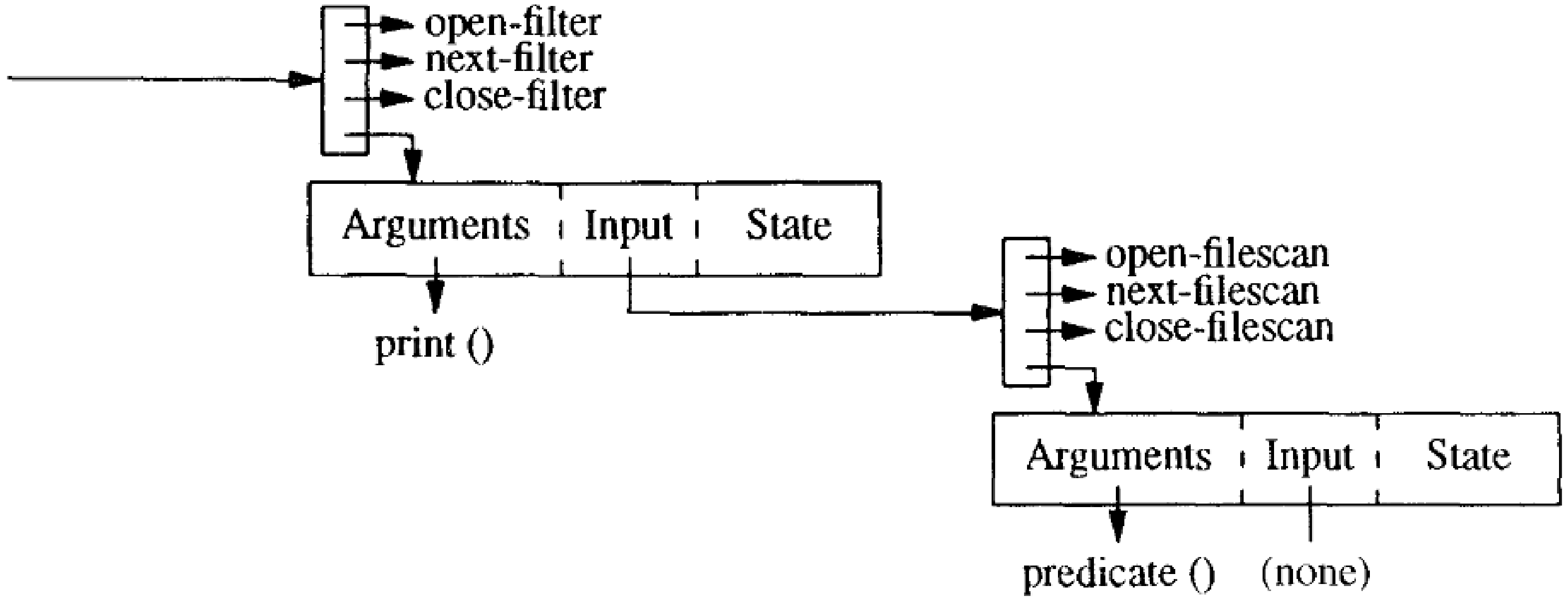
- Transfer data via interprocess communication mechanisms.
- Introduces new overhead (scheduling and IPC).

Synchronization & Data Transfer

Better Alternative?

Let the operators schedule each other within a single operating system process.

Iterators



filter function invokes the *filescan* function as needed and can pace the *filescan* according to the needs of the *filter*.

Benefits of Iterators

- The entire query plan is executed within a single process.
- Operators produce one item at a time on request.
- Items never wait in a temporary file or buffer between operators.
- Efficient in its time-space-product memory costs.
- No operator is affected by the complexity of the whole plan.

Discussion #2: Group of 3

- How do database systems' needs for dealing with large amounts of data differ from other applications? How much overlap is there with what is taken care of by the OS or other system changes?

Sorting

- All sorting algorithms in DBMS use merging:
  - Input data is written into initial sorted runs and then merged into larger runs until only one run is left -> the sorted output.
  - E.g., merge sort, quicksort.
- What happens if input is larger than main memory?

Quicksort

- Each run will have the size of allocated memory.
- The number of initial runs = input size / memory size.

Replacement Selection

- Recursively fill buffers in order from smallest, one at a time.
- The number of runs is about half of quicksort.

Hashing

- An alternative to sorting.
- Expected complexity of set algorithms:
  - Hashing, O(N)
  - Sorting, O(N log N)

Hash Table Overflow

- Hash-based query processing algorithms use an in-memory hash table of database objects to perform matching.
- If the required hash table is larger than memory, hash table overflow occurs.

- Input is divided into multiple partitions; process each independently.
- Concatenate the results of each partition to get the result of the entire operation.

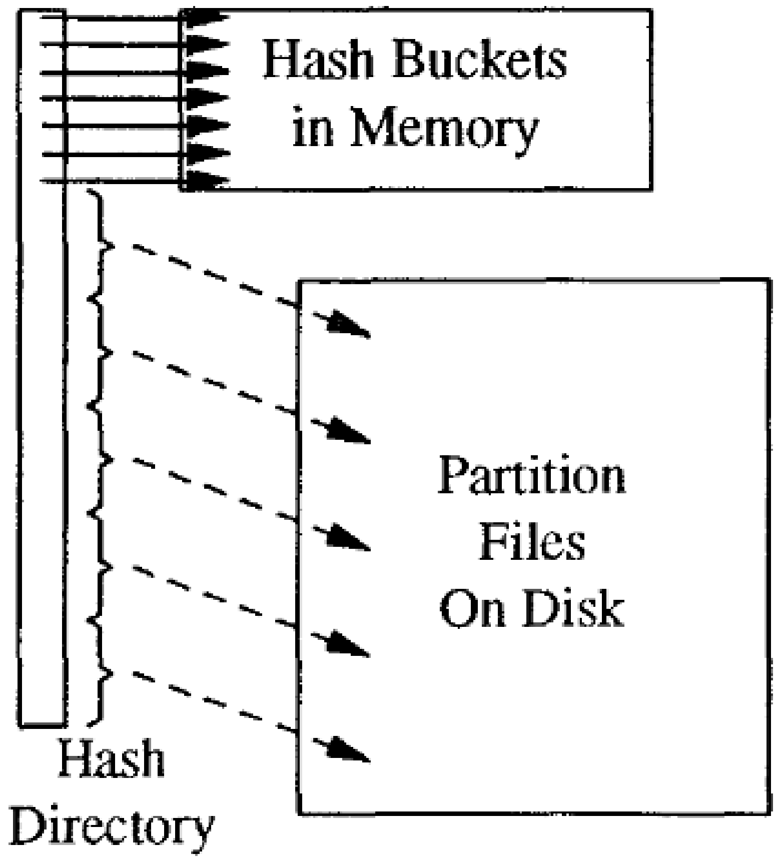


Figure 8. Hybrid hashing.

Indices

- Indices map keys or attribute values to locator information with which database objects can be retrieved.
- Reduce the number of accesses to secondary storage (which is slower compared to main memory).

Characteristic of Indices

- Clustering
  - Can be used to cluster the actual data items.
  - The order of index entries determines the order of items in the data file.
- Sparse
  - One index entry for each page of the primary file.
- Dense
  - There are the same number of entries in the index as there are items in the primary file.

Buffer Management

- Motivation
  - I/O cost can be reduced by caching data in an I/O buffer.
- Fixed page is not subject to replacement in the buffer pool.
- If all buffer frames are fixed, but a page needs to be replaced:
  - Dynamically grow the buffer pool
  - Abort transaction

Discussion #3 – Group of 3

Do you think that in the time since then the issues would have gotten better, because memories have gotten larger, or worse because there is a bigger gap between the time it takes to access memory and the time to access things on disk?

Merge Join

- Requires both inputs to be sorted.
- Merging the two inputs is like the merge process in sorting.
- Nested-Loop Join + Merge Join = Heap-Filter Merge-Join
  - Compared to block nested loops, the number of scans of the inner input is reduced by one half.
  - Compared to merge join, it saves writing and reading temporary files for the larger outer input.

Hash Join

- Build an in-memory hash table on the *build* input, and then probe the hash table using items from the *probe* input.
- Hybrid-hash allows extensive pipelining.
- If the build input fits into memory, no temporary files needed.
- Requires overflow avoidance/resolution for larger build input.

Nested-Loop Join

- For each item in the *outer* input:

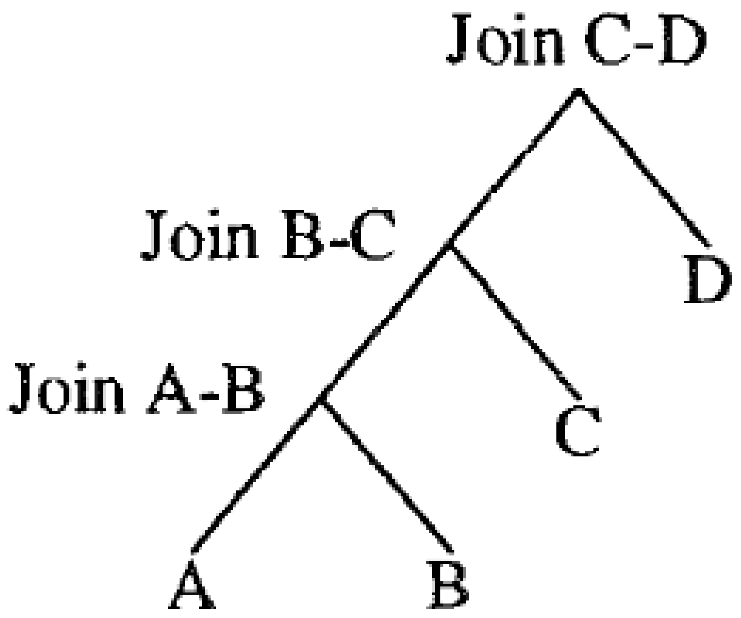
  Scan the entire *inner* input and find matches.

- Simple.

- If the inner input is produced by a complex subplan, it must be stored in a temporary file.

Benefits of Left-Deep Join

- Efficient; no need to scan inner (right) multiple times.
- Can be fully pipelined.
- Recall: System R considers only left-deep join trees.



Discussion #4: Group of 3

- Does the # of generally used joins seem large or small to you? Why?
- Are you surprised by any of the joins that are used?

Summary

- Sort-based or hash-based query processing algorithms?
  - Neither algorithm type outperforms the other in all situations.
  - Both should be available in query execution engines, for a choice to be made by the query optimizer.
  - Choices should depend on:

    - sizes of the two inputs

    - performance impairments due to skewed data or hash distributions

Discussion #5: Group of 2 (different research interests)

From the summary: "Query processing has been explored extensively in the last 20 years in the context of relational database management systems and is slowly gaining interest in the research community for extensible and object-oriented systems. This is a very encouraging development, because if these new systems have increased modeling power over previous data models and database management systems but cannot execute even simple requests efficiently, they will never gain widespread use and acceptance."

What tradeoff would you accept for better functionality with your data? Is this comparable to the tradeoff in programming languages, e.g., the slowness of Java vs its becoming an accepted programming language?

Discussion #6: Group of 3

- Do you find survey papers useful?
  - What kind of person do you think would be beneficial for reading and writing this kind of paper?
  - Who do you think can be able to write these papers?

- Compared to this old style of survey paper, what are advantages or disadvantages of having more recent SoK (Systemization of Knowledge) papers?