

# Evaluating Relational Operations: Part I

---

(From Chapter 14)

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Relational Operators

---

- **Select**
- Project
- Join
- Set operations (union, intersect, except)
- Aggregation

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Select Operator

---

```
SELECT *  
FROM Sailor S  
WHERE S.Age = 25  
      AND S.Salary > 100000
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Select Operator

---

Three cases:

- Case 1:
- Case 2:
- Case 3:

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Case 1:

---

- Assume that select operator is applied over a relation with  $N$  tuples stored in  $P$  data pages
- What is the cost of select operation in this case (in terms of # I/Os)?

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Select Operator

---

Three cases:

- Case 1:
- Case 2:
- Case 3:

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Case 2: Example

---

```
SELECT *
FROM Sailor S
WHERE S.Age = 25
      AND S.Salary > 100000
```

Matching index?

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

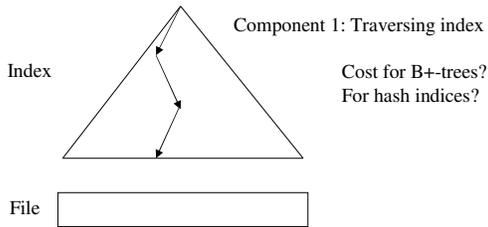
---

---

---

## Case 2: Cost Components

---



Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

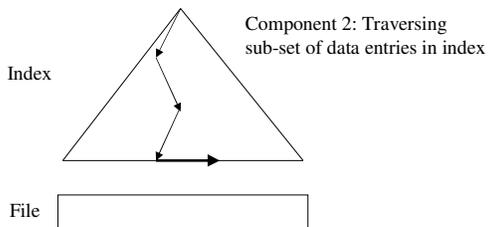
---

---

---

## Case 2: Cost Components

---



Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

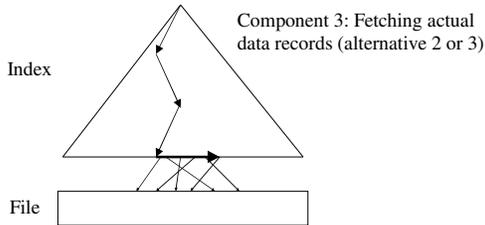
---

---

---

## Case 2: Cost Components

---



Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Cost of Component 1

---

- D is cost of reading/writing one page to disk (using random disk I/O)
- Hash index
  - Cost = \_\_\_\_\_
- B+-tree
  - Cost = \_\_\_\_\_

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Cost of Component 2

---

- N data entries (= # data tuples if alternative 2)
- Hash index
  - Linear hashing
  - B hash buckets
  - Average cost = \_\_\_\_\_
- B+ tree index
  - L = average number of entries per leaf page
  - S = **Selectivity** (fraction of tuples satisfying selection)
  - Average cost = \_\_\_\_\_

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Cost of Component 3

---

- S\*N data entries satisfy selection condition
  - S is selectivity, N is total number of data entries
- T is number of data tuples per page
- Hash index
  - Worst-case cost = \_\_\_\_\_
  
- B+ tree index
  - Worst-case cost = \_\_\_\_\_

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Putting it all together

---

- Total cost of select operations using unclustered B+ tree index  
\_\_\_\_\_
- Should we always use index in this case?
  - Depends on selectivity of selection condition!

❖ What about a clustered index?

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Component 3: Optimization

---

- Alternative 2 or 3, unclustered index
- Find qualifying data entries from index
  - Sort the rids of the data entries to be retrieved
  
  - Fetch rids in order

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Select Operator

---

Three cases:

- Case 1:
- Case 2:
- Case 3:

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Case 3: Example

---

```
SELECT *  
FROM   Sailor S  
WHERE  S.Age = 25  
       AND S.Salary > 100000
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Alternatives

---

- Option 1
  - Use available index (on Age) to get *superset* of relevant data entries
  - Retrieve the tuples corresponding to the set of data entries
  - Apply remaining predicates on retrieved tuples
  - Return those tuples that satisfy all predicates
- Option 2
  - Sequential scan! (always available)

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Case 3: Example

---

```
SELECT *
FROM   Sailor S
WHERE  S.Age = 25
       AND S.Salary > 100000
```

- Have Hash index on Age
- Have B+ tree index on Salary

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Alternatives

---

- Option 1
  - Choose **most selective** access path (index)
    - Could be index on Age or Salary, depending on selectivity of the corresponding predicates
  - Use this index to get **superset** of relevant data entries
  - Retrieve the tuples corresponding to the set of data entries
  - Apply remaining predicates on retrieved tuples
  - Return those tuples that satisfy all predicates

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Alternatives

---

- Option 2
  - Get rids of data records using each index
    - Use index on Age and index on Salary
  - **Intersect** the rids
    - We'll discuss intersection soon
  - Retrieve the tuples corresponding to the rids
  - Apply remaining predicates on retrieved tuples
  - Return those tuples that satisfy all predicates

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Alternatives

---

- Option 3
  - Sequential scan!

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

---

---

## ICE: Choose the best for each query!

---

R(a,b,c,d,e):  
5,000,000 records, 10 records/page  
stored as sorted file by R.a (candidate key in [0,4999999])  
What is best?  
a) access sorted file for R directly  
b) use clustered B+tree index on R.a  
c) use linear hashing index on R.a  
d) use clustered B+tree index on (R.a, R.b)  
e) use linear hashing index on (R.a, R.b)  
f) use unclustered B+tree index on R.b  
Queries: SELECT \* FROM R WHERE ...  
1. a < 50,000 AND b < 50,000  
2. a = 50,000 AND b < 50,000  
3. a > 50,000 AND b = 50,000  
4. a = 50,000  
5. a <> 50,000 AND b = 50,000  
6. a < 50,000 OR b = 50,000

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

---

---

## Relational Operators

---

- Select
- Project
- Join
- Set operations (union, intersect, except)
- Aggregation

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

---

---

## Example

---

```
SELECT DISTINCT S.Name, S. Age
FROM   Sailor S
```

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Alternatives

---

- Option 1
- Option 2
- Option 3

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Example

---

```
SELECT DISTINCT S.Name, S. Age
FROM   Sailor S
```

- Have B+ tree index on (Name, Age)

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Using “Covering” Index

---

- Simply scan leaf levels of index structure
  - No need to retrieve actual data records
- Works so long as the index search key includes all the projection attributes
  - Extra attributes in search key are \_\_\_\_\_
  - Best if projection attributes are \_\_\_\_\_ search key
    - Can eliminate duplicates in single pass of index-only scan
- Other examples
  - Hash index on (SSN, Name, Age)
  - B+ tree index on (Age, # Dependents, Name)

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Example

---

```
SELECT DISTINCT S.Name, S. Age
FROM Sailor S
```

- Have Hash index on Name
- Have B+ tree index on Age
- Sailor relation has 100 other attributes!

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Using RID Joins

---

- Retrieve (SearchKey1, RID) pairs from first index
- Retrieve (SearchKey2, RID) pairs from second index
- Join these based on RID to get (SearchKey1, SearchKey2, RID) triples
- Project out the third column to get the desired result

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Evaluation Alternatives

---

- Option 1
  - Using Indices
- Option 2
  - Based on sorting
- Option 3
  - Based on hashing

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Example

---

```
SELECT DISTINCT S.Name, S. Age
FROM Sailor S
```

- No indices

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

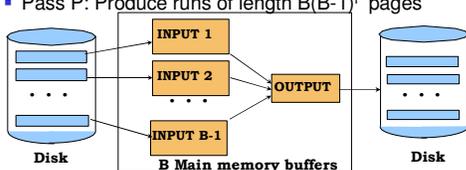
---

---

## General External Merge Sort

---

- Phase 2: Make multiple passes to merge runs
  - Pass 1: Produce runs of length  $B(B-1)$  pages
  - Pass 2: Produce runs of length  $B(B-1)^2$  pages
  - ...
  - Pass P: Produce runs of length  $B(B-1)^P$  pages



Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

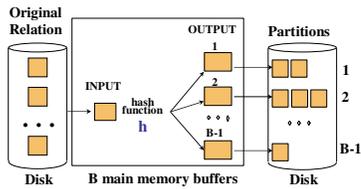
---

---



## Projection Based on Hashing

- Assume relation does not fit in memory
- Phase 1
  - Divide relation into partitions



Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Phase 1: Analysis

- Number of data pages =  $N$ 
  - Assume all attributes are projected out
- Cost of reading/writing disk page =  $D$
- Number of Partitions = \_\_\_\_\_
- Length of each partition = \_\_\_\_\_
- Cost of Phase 1 = \_\_\_\_\_

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

## Two Cases for Each Partition

- Case 1
  - Partitions fits in memory
- Case 2
  - Partition does not fit in memory

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

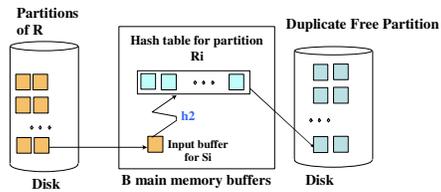
---

---

---

## Case 1: Partition Fits in Memory

- Use  $h_2 \leftrightarrow h_1$ !



- $R$  is number of pages in result
  - After eliminating duplicates
- Cost =  $D * (N + R)$

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

---

---

## Case 2: Partition Doesn't fit in Memory

- Recursively apply Phase 1 algorithm on the partition!
  - Use hash function  $h_2 \leftrightarrow h_1$ !
- Analysis
  - Size of each partition after  $P$  partitioning phases = \_\_\_\_\_
  - Stop partitioning when : \_\_\_\_\_
  - # Partitioning phases = \_\_\_\_\_
  - Total cost of Phase 1 = \_\_\_\_\_

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

---

---

## Comments on Projection

- Sort-based approach vs. hash-based approach
  - Which one would you choose?
  - Why?

Database Management Systems, R. Ramakrishnan and Johannes Gehrke

---

---

---

---

---

---

---

---

---

---