CS 143 Discussion Session

Mingda Li

Today's topic

- Midterm review.
  - Important concepts.
  - Some practice.
  - Review some homework questions.

Relational Algebra

Way to ask queries of relations
Operators

Operation	Symbol	Explanation
Selection	σ	Selects rows
Projection	π	Selects columns
Union	U	
Difference	-	
Cross-product	×	
Join	⋈	Join tables on some condition
Rename	ρ	
Aggregate	ϑ	

General SQL

SELECT attributes, aggregates

FROM relations(tables)

WHERE conditions

GROUP BY attributes

HAVING conditions on aggregates

ORDER BY attributes, aggregates

Relational Algebra vs. SQL

- Relational Algebra
  - set semantics, no duplicate tuples

- SQL
  - bag semantics
  - Need to explicitly add keyword DISTINCT for deduplication

RA vs. SQL: sample question

1. Consider the following SQL query on the table R(A, B, C). You may assume that there are no NULLs.

```
SELECT R1.A
FROM   R R1, R R2
WHERE  R1.A=R2.A AND R1.B='UCLA' AND R2.C='blue';
```

Is it possible to write an equivalent relational algebra expression? If yes, write such an expression succinctly. If no, briefly explain why.

ANSWER:

No. When we project on A, there can be multiple tuples that satisfy the condition with the same A values. A relational algebra expression will always remove such duplicates while the above SQL query will not.

Database Integrity

- Key constraints
- Referential Integrity
- Check constraint

Database Integrity

- Key Constraint

```
CREATE TABLE Course (
    dept CHAR(2) NOT NULL,
    cnum INTEGER NOT NULL,
    sec INTEGER NOT NULL,
    unit INTEGER,
    instructor VARCHAR(30),
    title VARCHAR(30),
    PRIMARY KEY(dept, cnum, sec),
    UNIQUE(dept, cnum, instructor),
    UNIQUE(dept, sec, title) )
```

Database Integrity

- Check Constraint

```
CREATE TABLE Class(
    dept char(2),
    unit INT,
    …
    CHECK(dept<>'CS' OR unit > 3))
```

Database Integrity

- Trigger

CREATE Trigger <name>

<event>

<referencing clause>

WHEN (<condition>)

<action>

Views

CREATE VIEW <name> AS

<Query>

Authorization

- GRANT <privileges> ON <R> TO <user> [WITH GRANT OPTION]
- REVOKE <Privileges> ON <R> FROM <user> [CASCADE | RESTRICT]

Disks and Files

- Disk Structure

- Sequential vs. Random I/O
- Access time =
  seek time + rotation delay + data transfer time
- Transfer Rate

Disks and Files

- File
- Fixed-length Tuples:
  spanned vs Unspanned
- Variable-Length Tuple (Slotted Pages)
- Long Tuple
- Sequential Tuples

Index

- B+ Tree

- Very important!!
- n: max # of pointers in a node
- Algorithm in notes/textbook

Index

- Multi-level index
- Primary index
- Secondary index
- Sparse index
- Dense index

Sequential File

- Table sequenced by sid. Find sid=40?

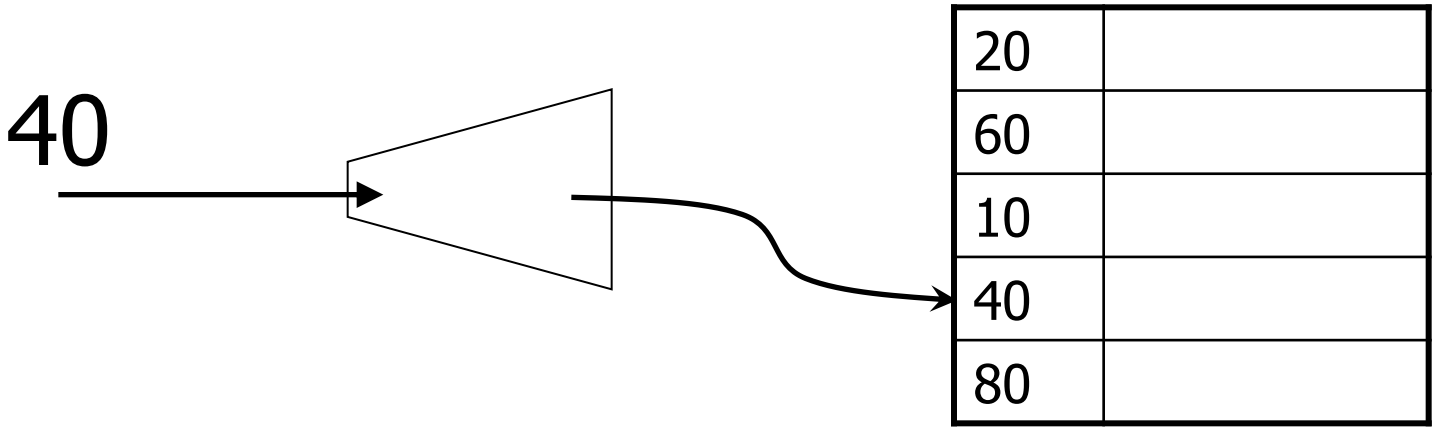
sid	name	GPA
20	Susan	3.5
30	James	1.7
40	Peter	2.6
50	Elaine	3.9
60	Christy	2.9

Sequential File - Binary Search

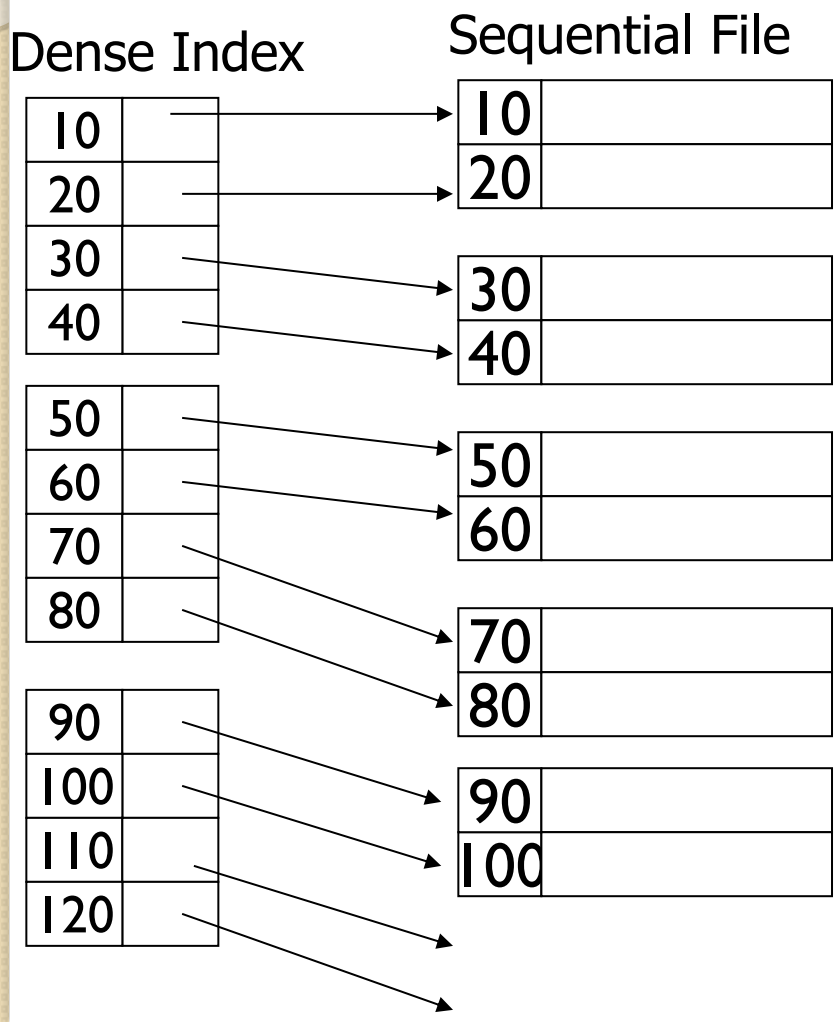
- 100,000 blocks
- Q: How many blocks to read?
  - $100{,}000 <= 2^n$
  - $n >= \log_2 100{,}000 = 16.61$
  - $n = 17$

Better way: index

- Build an "index" on the table
  - An auxiliary structure to help us locate a record given a "key"

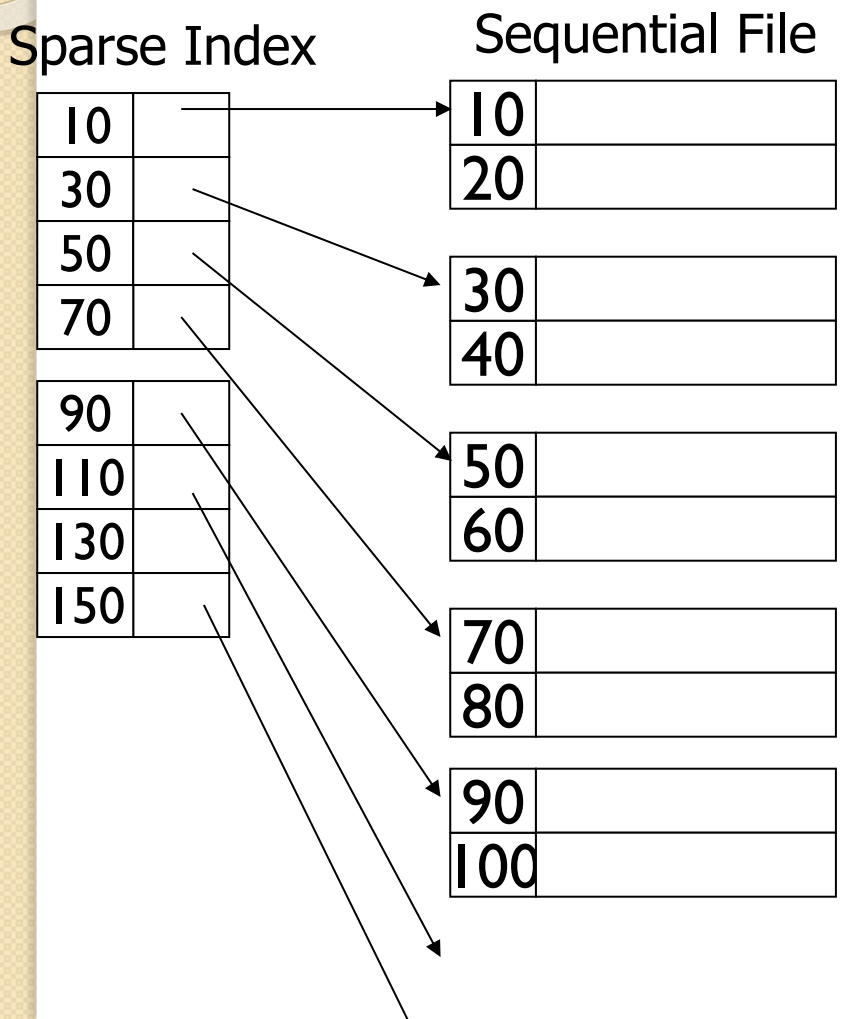


Dense, Primary Index



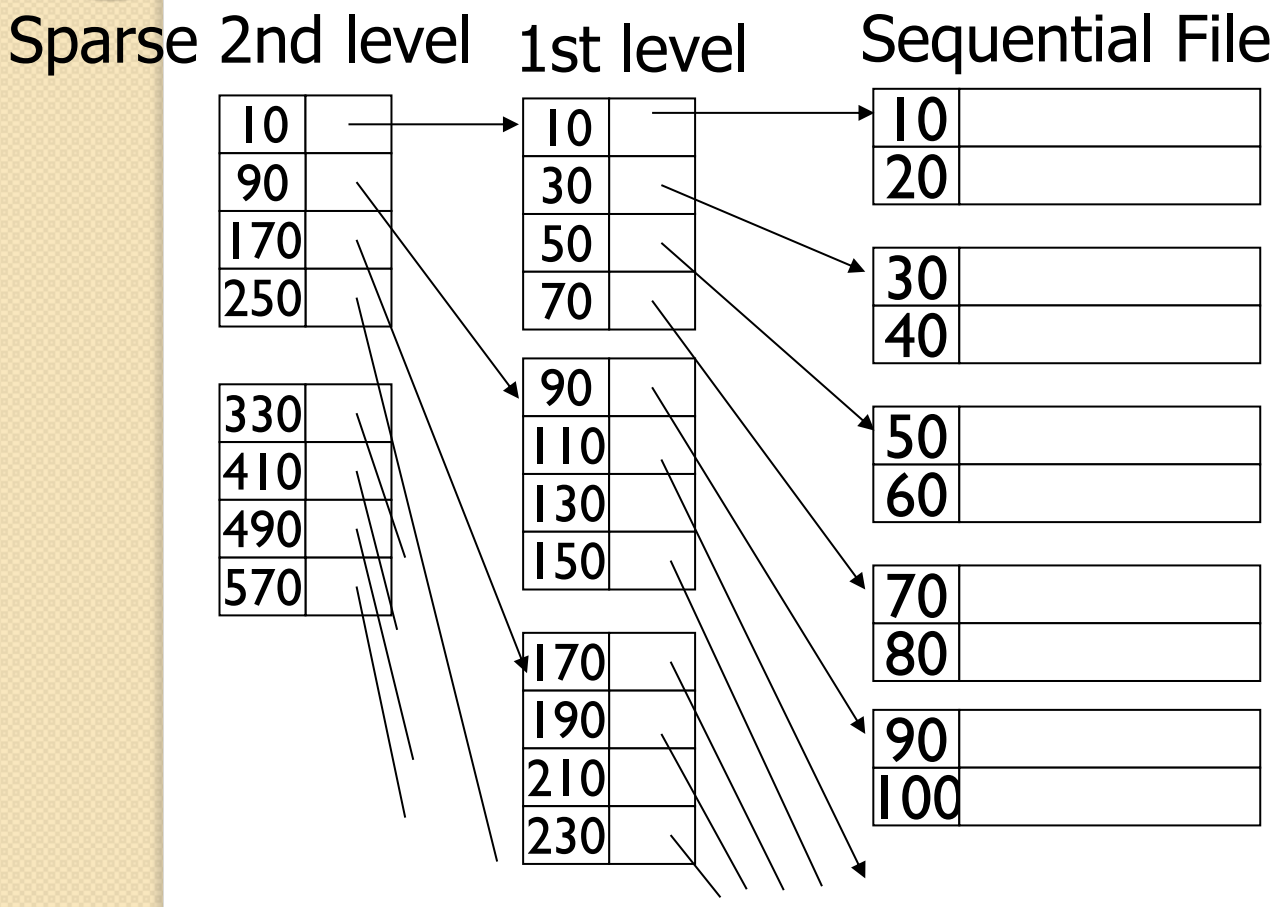
- Primary index (clustering index)
  - Index on the search key
- Dense index
  - (key, pointer) pair for **every record**
- Find the key from index and follow pointer
  - Maybe through binary search
- Q: Why dense index?
  - Isn't binary search on the file the same?
  - Index require less space and could be load into memory

Sparse, Primary Index



- Sparse index
  - (key, pointer) pair per **every "block"**
  - (key, pointer) pair points to the first record in the block
- Q: How can we find 60?
  - Binary search on sparse index - 50
  - Load the block
  - Find 60

Multi-level index



- What if index size is big?
  - Build multi-level
- 1st level: dense/sparse
- 2nd level or higher level: sparse only

Primary Index vs. Secondary Index

Sequential File

10	
20	

30	
40	

50	
60	

70	
80	

90	
100	

30	
50	

20	
70	

80	
40	

100	
10	

90	
60	

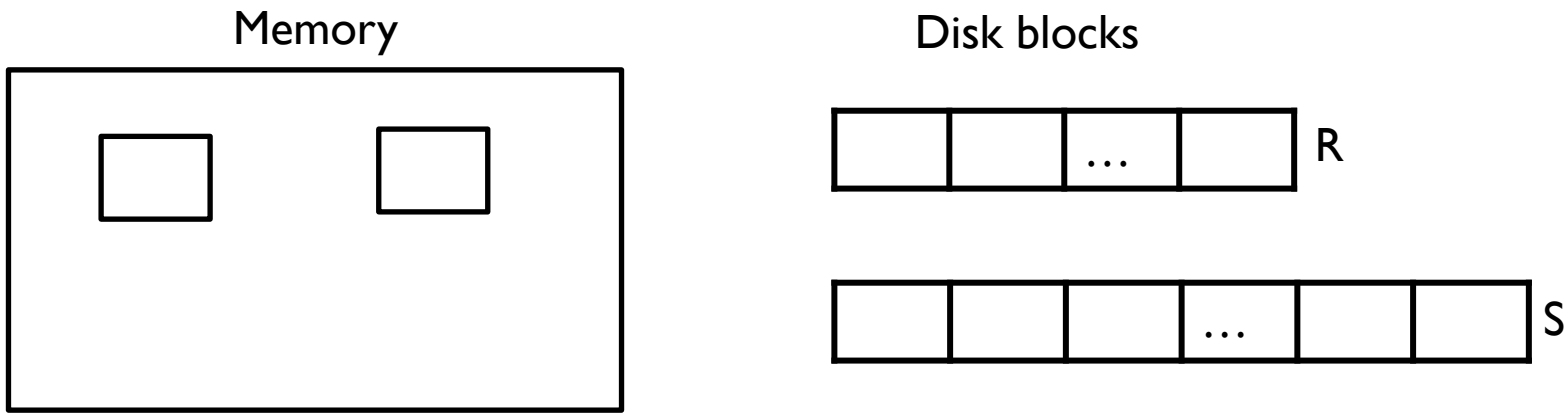
Primary Index vs. Secondary Index

- Primary index
  - Clustering
  - Tuples in the table are ordered by the index search key
- Secondary index
  - Non-clustering
  - Tuples in the table are not ordered by the index search key
    - Index on a non-search-key for sequential file
    - Unordered file

A sparse index can only be built on clustered data and should be primary index.

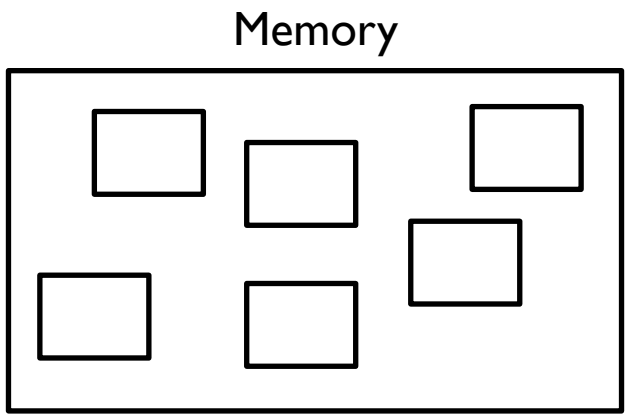
Join Algorithm: Cost Model

- Total number of disk blocks that have been read/written
  - Data are load from disk to memory block by block
  - Only count it when a disk block is being loaded into memory
  - Ignore the time of in-memory operations

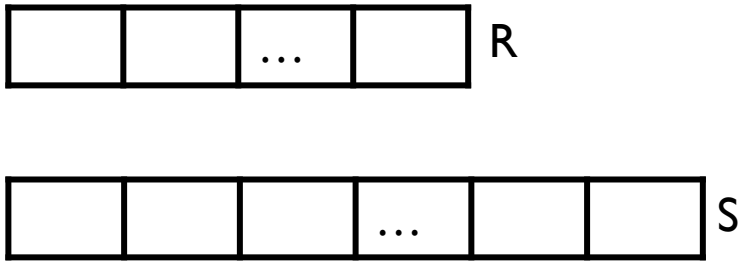


Example to Use

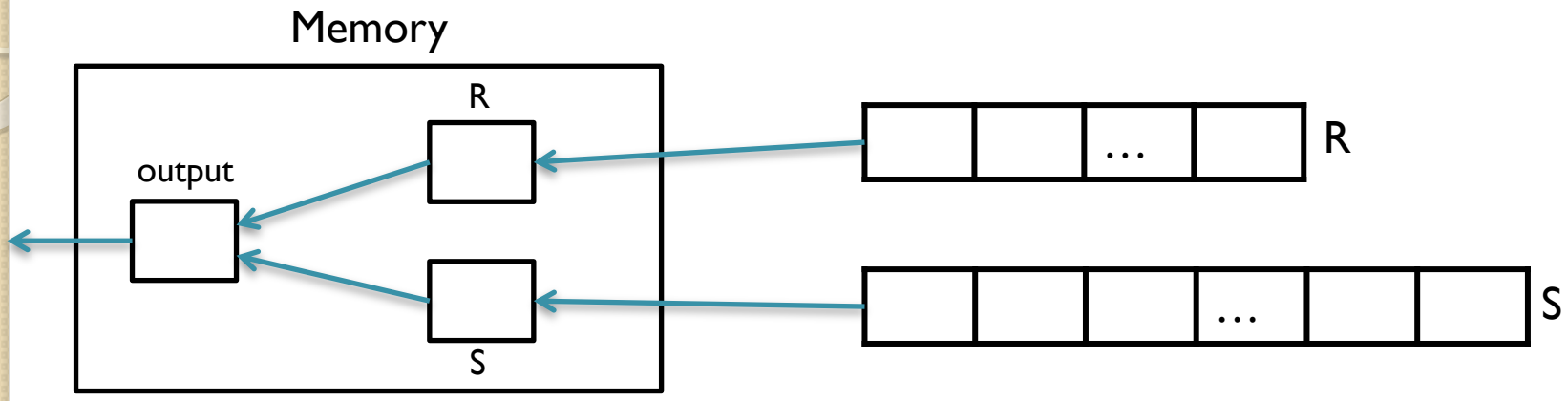
- Two tables R, S
- |R|=1,000 tuples, |S|=10,000 tuples
- 10 tuples/block
- b_R=100 blocks, b_s=1,000 blocks
- Memory buffer: M=22 blocks



SELECT * FROM R, S WHERE R.C=S.C

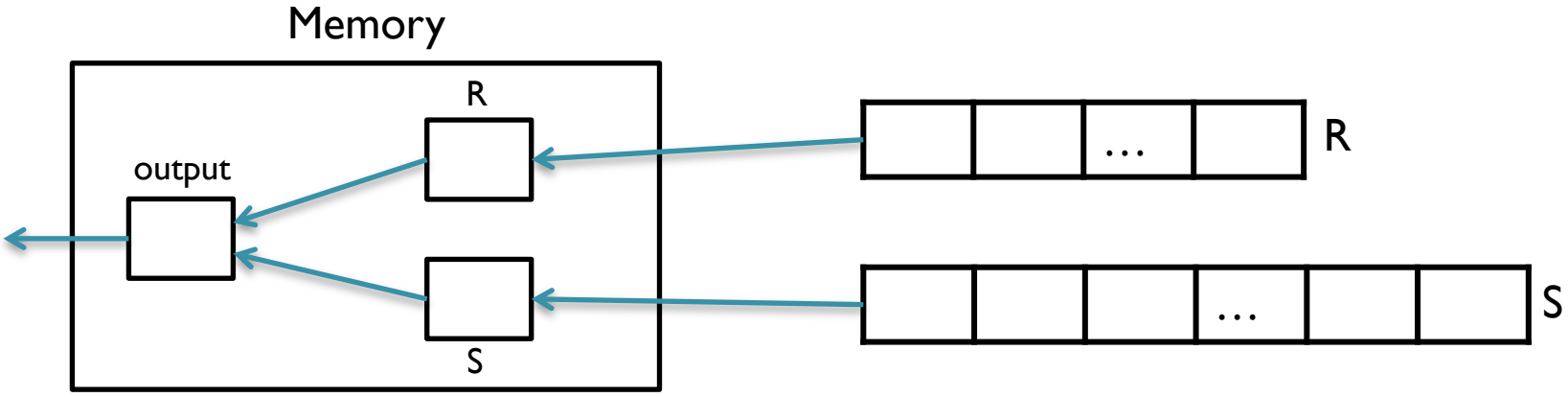


Nested-Loop Join



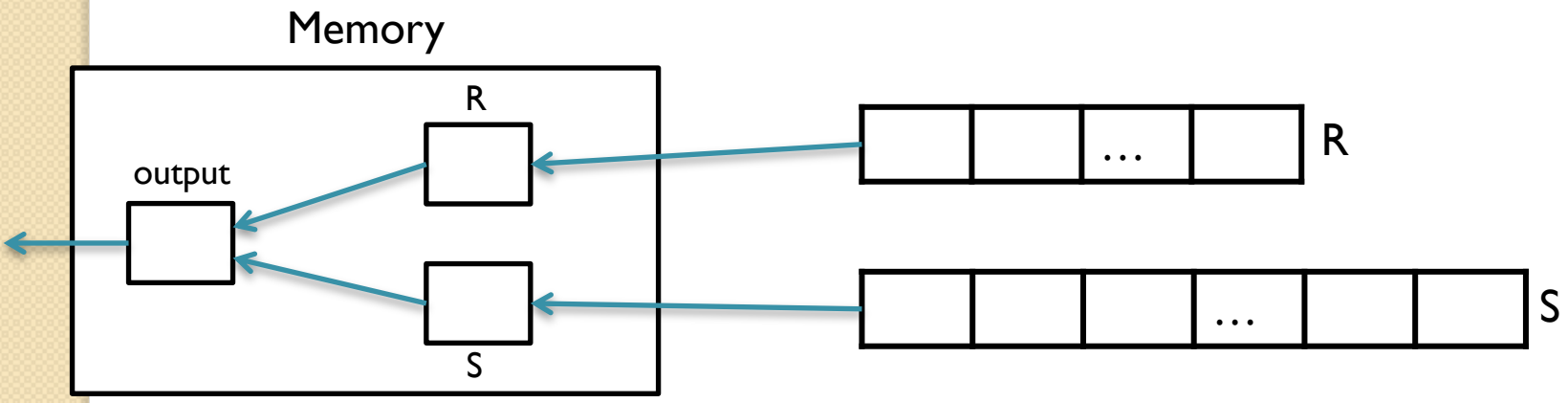
- Read a block from R at a time
  - For **each tuple** in R, compare it with **each tuple** in S
- $b_R + |R| * b_S$
- $= 100 + 1000 * 1000 = 1000100$

Nested-Loop Join



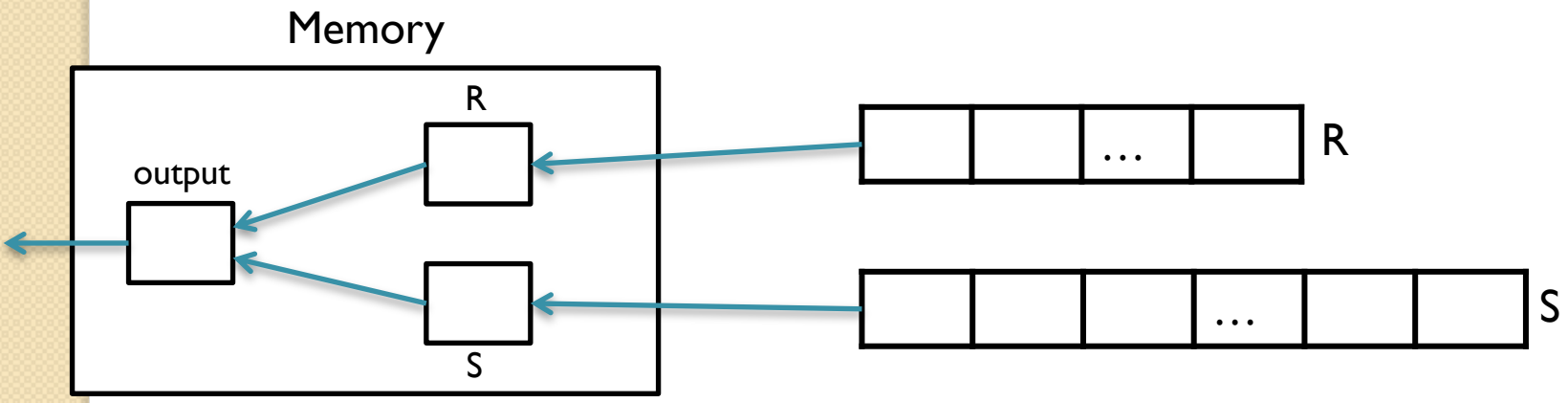
- What if we read S first?
- $b_S + |S| * b_R$
- $= 1000 + 10000 * 100 = 1001000$
-

Block Nested-Loop Join



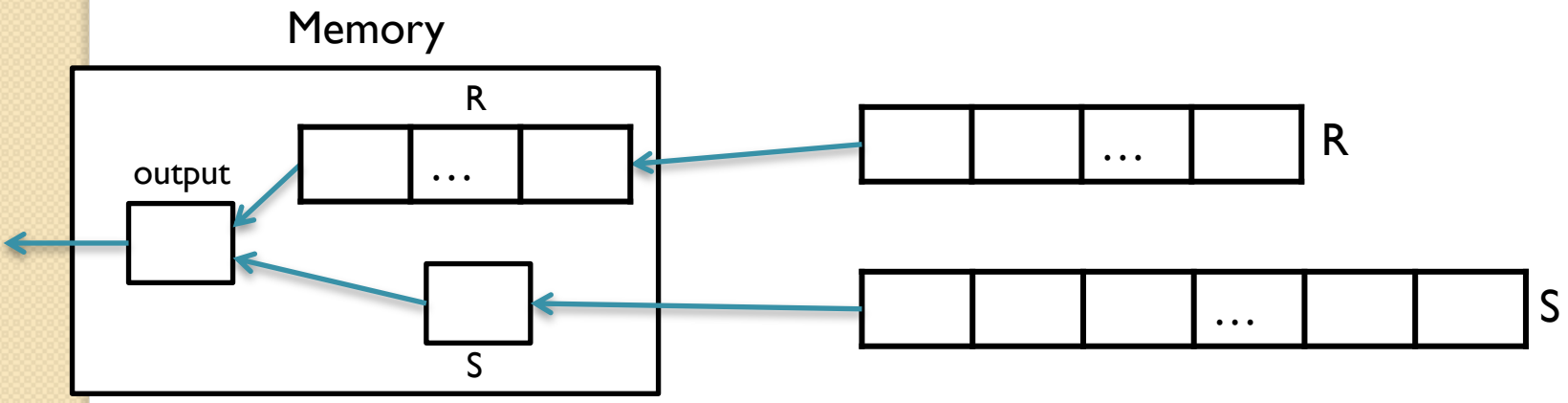
- Read a block from R at a time
  - For **each block** in R, compare it with **each block** in S
- $b_R + b_R * b_S$
- =100+100*1000=100100

Block Nested-Loop Join



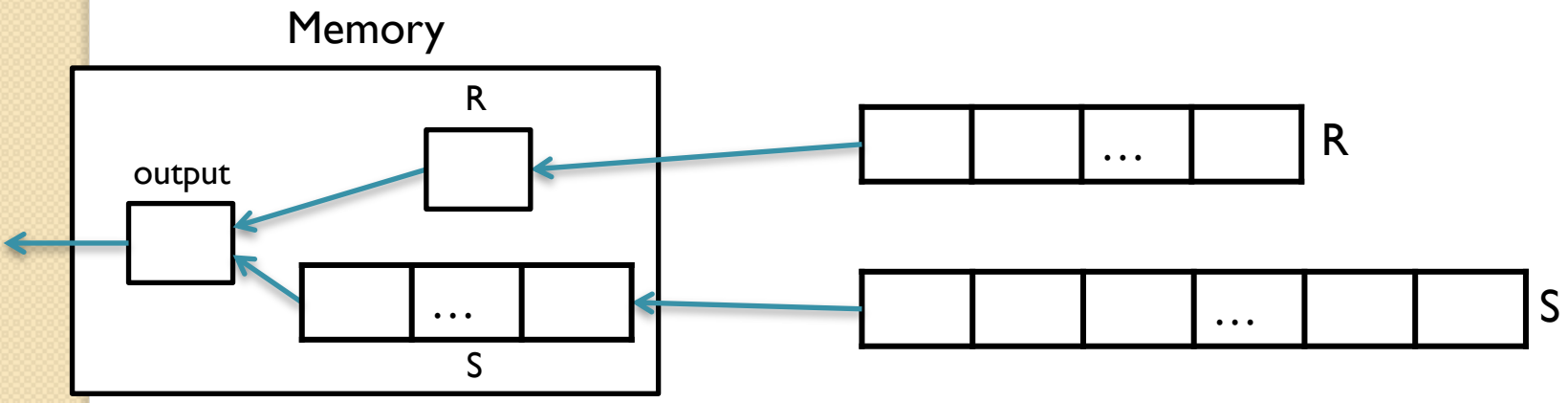
- What if we read S first?
- $b_S + b_S * b_R$
- =1000+1000*100=101000
-

Bulk Block Nested-Loop Join



- Read several blocks(memory can hold) from R at a time
  - For **each 20 blocks** in R, compare them with **each block** in S
- $b_R+[b_R/(M-2)]*b_S$
- =100+100/20*1000=5100

Bulk Block Nested-Loop Join



- What if we read S first?
- $b_S+[b_S/(M-2)]*b_R$
- =1000+1000/20*100=6000
-

Nested-loop Join Summary

- Performance: bulk load > block load > tuple load

SQL Practice

Consider the following schema about students and courses. Primary keys are underlined.

Students (sid, sname, street, city, age, gender)
Registered (sid, cid, grade)
Courses (cid, cname, profname)

Note that all of the SQL queries in parts a and b (next two slides) are syntactically valid

a) Which of the following queries produces the CIDs of courses that have no registered students? (One or more options are correct)

A. SELECT cid FROM Registered
EXCEPT
SELECT cid FROM Courses;

B. SELECT cid FROM Courses
WHERE cid IN
(SELECT cid FROM Registered
GROUP BY cid HAVING COUNT(*) = 0);

C. SELECT c.cid FROM Courses c
WHERE NOT EXISTS
(SELECT cid FROM Registered r WHERE r.cid = c.cid);

D None of the above

Answer: C

Schema
Students (sid, sname, street, city, age, gender)
Registered (sid, cid, grade)
Courses (cid, cname, profname)

b) Which of the following queries are equivalent to the query: SELECT DISTINCT profname FROM Courses (One or more options are correct)

A. SELECT profname FROM Courses GROUP BY profname;

B. SELECT profname FROM Courses
UNION SELECT profname FROM Courses;

C. SELECT DISTINCT profname FROM Courses
UNION ALL SELECT profname FROM Courses

D. SELECT DISTINCT profname FROM Courses WHERE NULL = NULL

E. None of the above

Answer: A, B

Schema
Students (sid, sname, street, city, age, gender)
Registered (sid, cid, grade)
Courses (cid, cname, profname)

c) When would the following two queries return different results for a given database instance? A one sentence answer should be sufficient!!!

```
SELECT s.sname FROM Students s LEFT OUTER JOIN
    Registered r ON s.sid = r.sid
SELECT s.sname FROM Students s, Registered r WHERE s.sid =
    r.sid
```

If a student is not registered for a course.

Schema
Students (sid, sname, street, city, age, gender)
Registered (sid, cid, grade)
Courses (cid, cname, profname)

d) In the space below, write a SQL query that returns the name and SID of every student enrolled in the class 'CS186' whose age is greater than the average age of all the students enrolled in that class. ('CS186' is a CID.)

One solution with subqueries (more in slide notes):

```
SELECT S.sname, S.sid
FROM Students S, Registered R
WHERE S.sid = R.sid AND R.cid = 'CS186' AND
      S.age > (SELECT AVG(S2.age)
               FROM Students S2, Registered R2
               WHERE S2.sid = R2.sid AND R2.cid = 'CS186')
```

Schema

Students (sid, sname, street, city, age, gender)
Registered (sid, cid, grade)
Courses (cid, cname, profname)

Relational Algebra Practice

Recall the schema about students and courses from the previous question:

Students (sid, sname, street, city, age, gender)

Registered (sid, cid, grade)

Courses (cid, cname, profname)

a) In the space below, write a Relational Algebra expression that returns the sid and sname of all students who received an "A" in a course taught by "Hilfinger".

$$\pi_{sid,sname}(\sigma_{grade='A' \wedge profname='Hilfinger'}(S \bowtie R \bowtie C))$$

Schema
Students (sid, sname, street, city, age, gender)
Registered (sid, cid, grade)
Courses (cid, cname, profname)

Objective Questions

[5 points] Consider two relations with the same schema: R(A,B) and S(A,B). Which one of the following relational algebra expressions is not equivalent to the others?

$\Pi_{R.A}((R \cup S) - S)$

$\Pi_{R.A}(R) - \Pi_{R.A}(R \cap S)$

$\Pi_{R.A}(R - S) \cap \Pi_{R.A}(R)$

2nd is different!

E.g.
R {(2, 'hi') (2, 'bye')}
S {(2, 'hi')}

Can intersection be expressed using set difference?

Yes: $R \cap S = R - (R - S) = S - (S - R)$

Can intersection be expressed using cartesian product and projection?

No, neither operator can remove elements!

Disk and Indexing

We have 1 million employee tuples with unique key `Eno`. The length of each tuple is 50 bytes. These are stored as fixed-length unspanned records in a file consisting of blocks of size 2048 bytes.

On this file, we build a **sparse** index on `Eno`. The index is is organized as a B+ tree, where each key takes 18 bytes and each pointer takes 22 bytes (the leaf nodes are chained together as in the textbook). The B+ tree blocks contain 2048 bytes.

A. How many blocks are needed to store all the records in the file.

$$2048/50 = 40 \textit{ records per block. } 10^6/40 = 25000.$$

B. Compute the blocks used at each level of the B+ tree, for the best case and the worst case.

N=51 pointers (50 at the leaf level.) Worst case: 25 at bottom level, 26 at other levels

B+ tree, best case:

Leaf level: sparse index. One pointer per block: $25.000/50 = 500$

Next Level: 10

The root:1

B+ tree, worst Case:

Leaf nodes: $25.000/25 = 1000$

Next Level: $1000/26 = 38.4$ *take the FLOOR (not the ceiling). 38!*

Next Level: $38/26$. *Cannot split in two blocks. This is the root!*

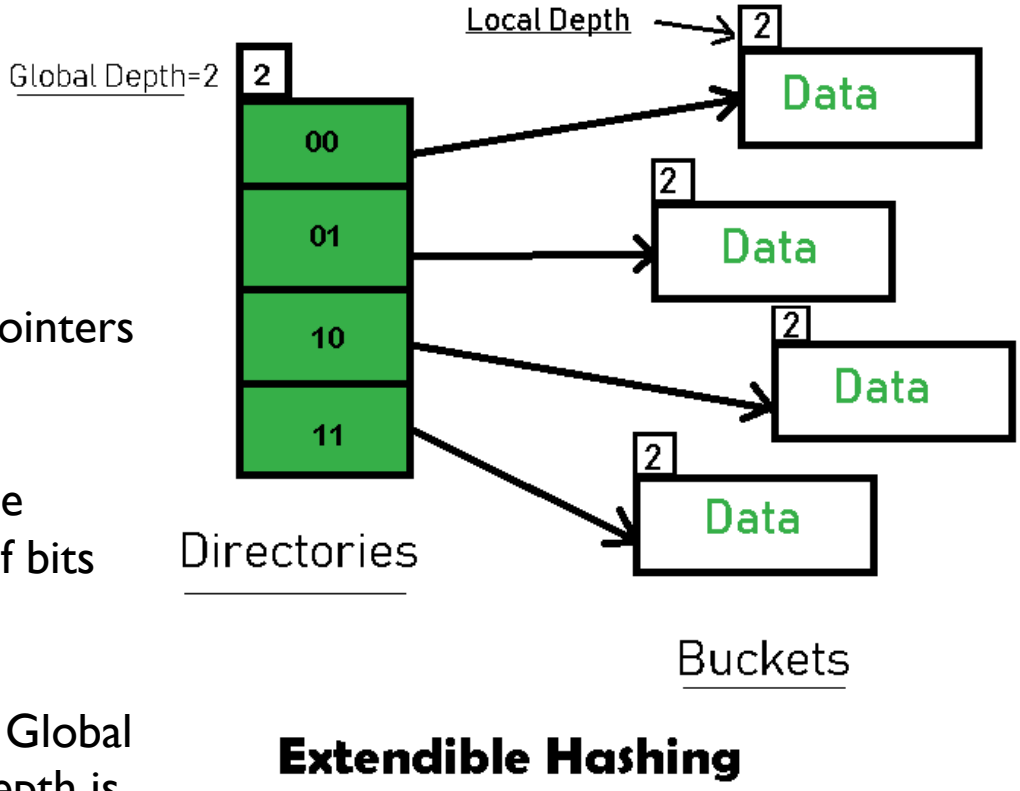
Suppose n keys max: 18 *n + 22*(n+1) <= 2048
After getting n, from the formula we can get the other max or min info.

C. Is this index a primary index or a secondary one?

A sparse index can only be built on clustered data. Thus this is a primary index.

Extendible Hashing

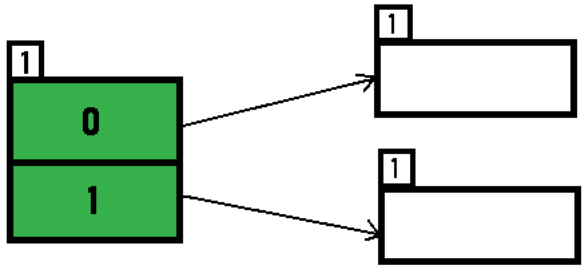
- Exp.

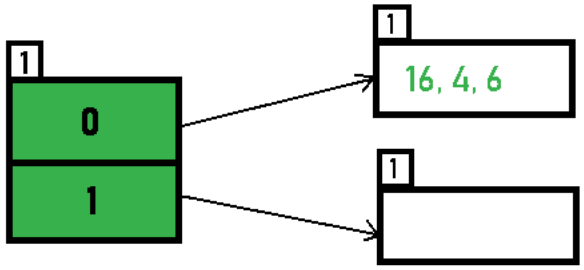


Extendible Hashing

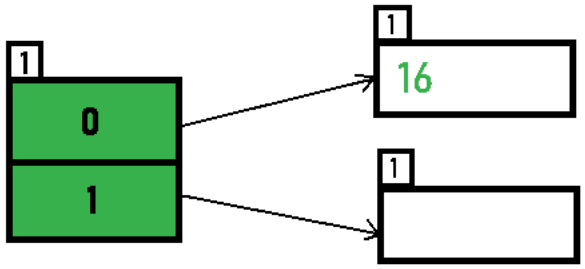
Directories: These containers store pointers to buckets
Buckets: They store the hashed keys
Global Depth: It is associated with the Directories. They denote the number of bits which are used by the hash function to categorize the keys
Local Depth: It is the same as that of Global Depth except for the fact that Local Depth is associated with the buckets and not the directories

Extendible Hashing





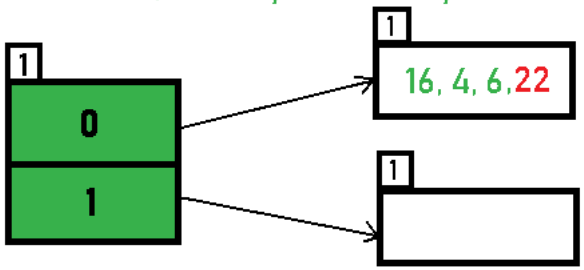
Hash(4)=100
Hash(6)=110



Hash(16)= 10000

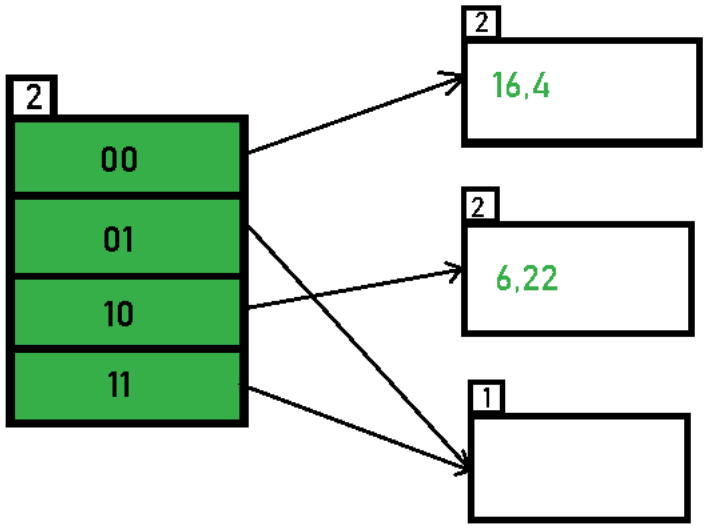
OverFlow Condition

Here, Local Depth=Global Depth



Hash(22)=10110

After Bucket Split and Directory Expansion



HW 4 Prob 4

Suppose you have 2 relations, $R(\overline{A}, B)$ and $S(\overline{B}, C)$, with the following characteristics:

- The size of one disk block is 1000 bytes.
- Attributes A, B are of length 10 bytes. Attribute C is of length 180 bytes. [50R tiuple per block]
- The tuples are not spanned across disk blocks [5 tuples per block]
- $|R| = 5,000$ (number of tuples of R) [100 blocks]
- $|S| = 500$ (number of tuples of S) [B+C= 90: 5 S tuples per block: 100 blocks]
- We have 30 blocks of memory buffer
- We use one disk block for one B+tree node
- Each pointer in a B+tree index (both a record pointer and a node pointer) uses 10 bytes.

1. (2 points) What is the minimum number of blocks needed to store R and S?

2. (2 points) We want to construct a dense B+tree on attribute B of table S by scanning the S table. What is the maximum possible n for the B+tree given the above parameters?

n: pointers 10n +10(n-1) <= 100

3. Assume the numbers computed in the previous problems. Assume that each node in the B+tree contains the minimum number of keys and pointers (as long as it is allowed in our parameter setting).

   (a) (4 points) How many nodes does the constructed B+tree have?

Layer by layer compute.
Leaf: 500/25 = 20 leaf nodes
Then, only need a root.

(b) (4 points) How many disk IOs would be incurred during the construction of the B+tree on `S.B`? Assume that you use the main memory buffer in the most efficient way to minimize the number of disk IOs. *In your answer, please include the cost of reading the tuples from S and writing the constructed B+tree to the disk.*

I/O cost: # blocks

Block of S +write back the B+ tree blocks.

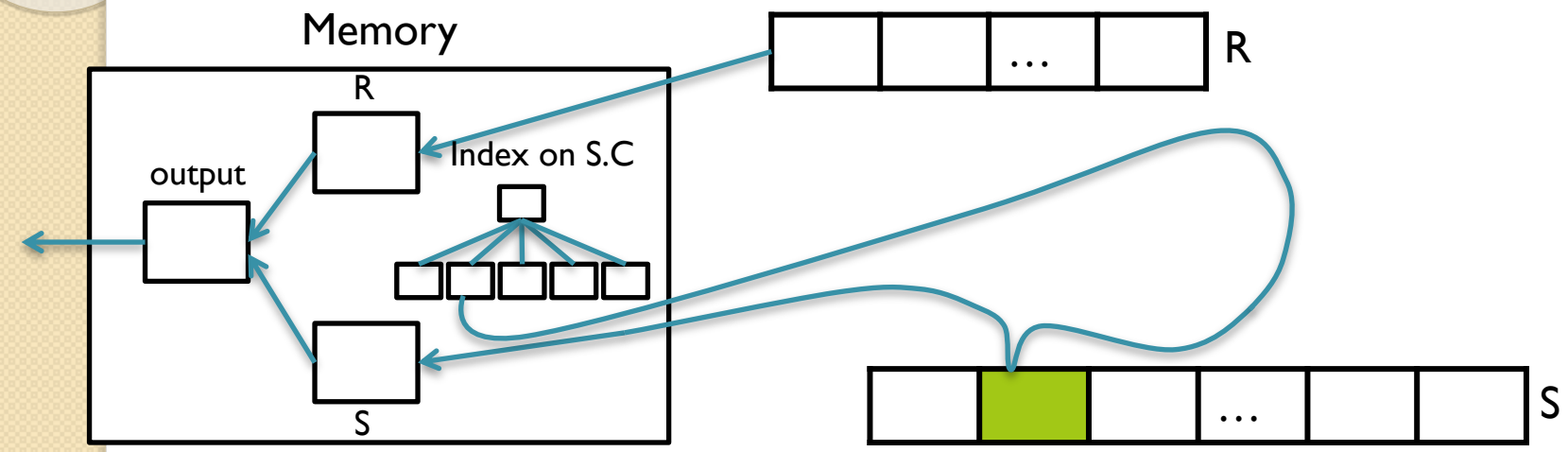
4. (4 points) How many disk IOs would have been incurred if we used block nested loop join? Is it worthwhile to construct an index on `S.B` to perform the join using the above algorithm considering the index construction overhead?

Index join vs nested loop join.

Index join

- **SELECT * FROM R, S WHERE R.C=S.C**
- What if we have an index built on S.C?
- Index join cost:
  - IO for R scanning
  - IO for index look up
  - IO for tuple read from S

Index Join: An example



Example 1
15 blocks for index (1 root, 14 leaf), read index into memory first
On average, one matching S tuples per an R tuple

How many IO?
index: 15
Cost of searching within B+ tree: 0 (already in memory) $15+b_R+|R|*(0+1)$
Average mach for each R tuple in S: One $15+100+1000*(0+1)=1115$

- Due to conflict schedule, I will not proctor the midterm.
- Good luck for your midterm! See you next week.