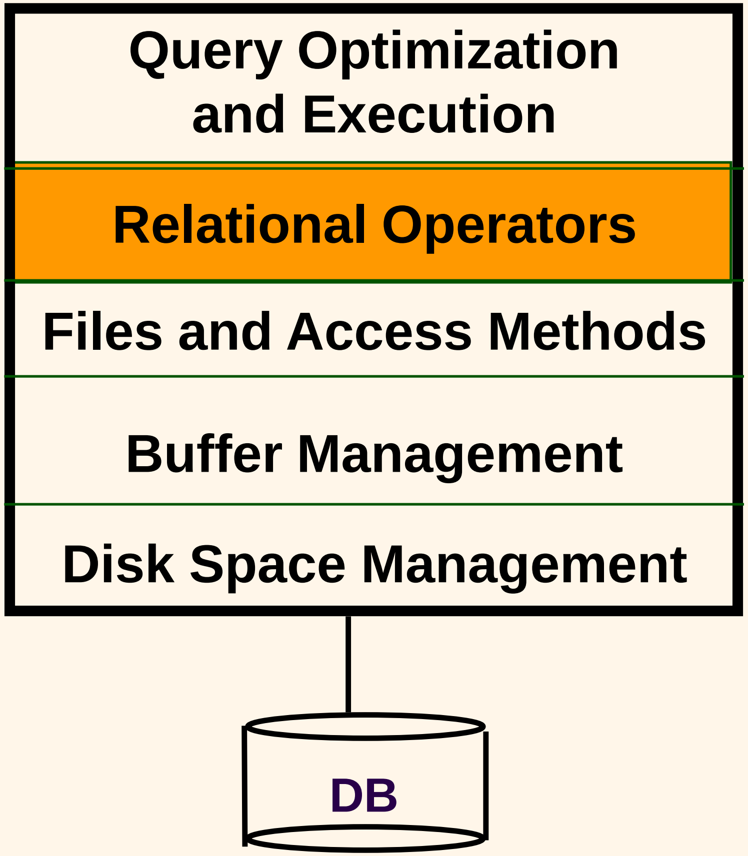
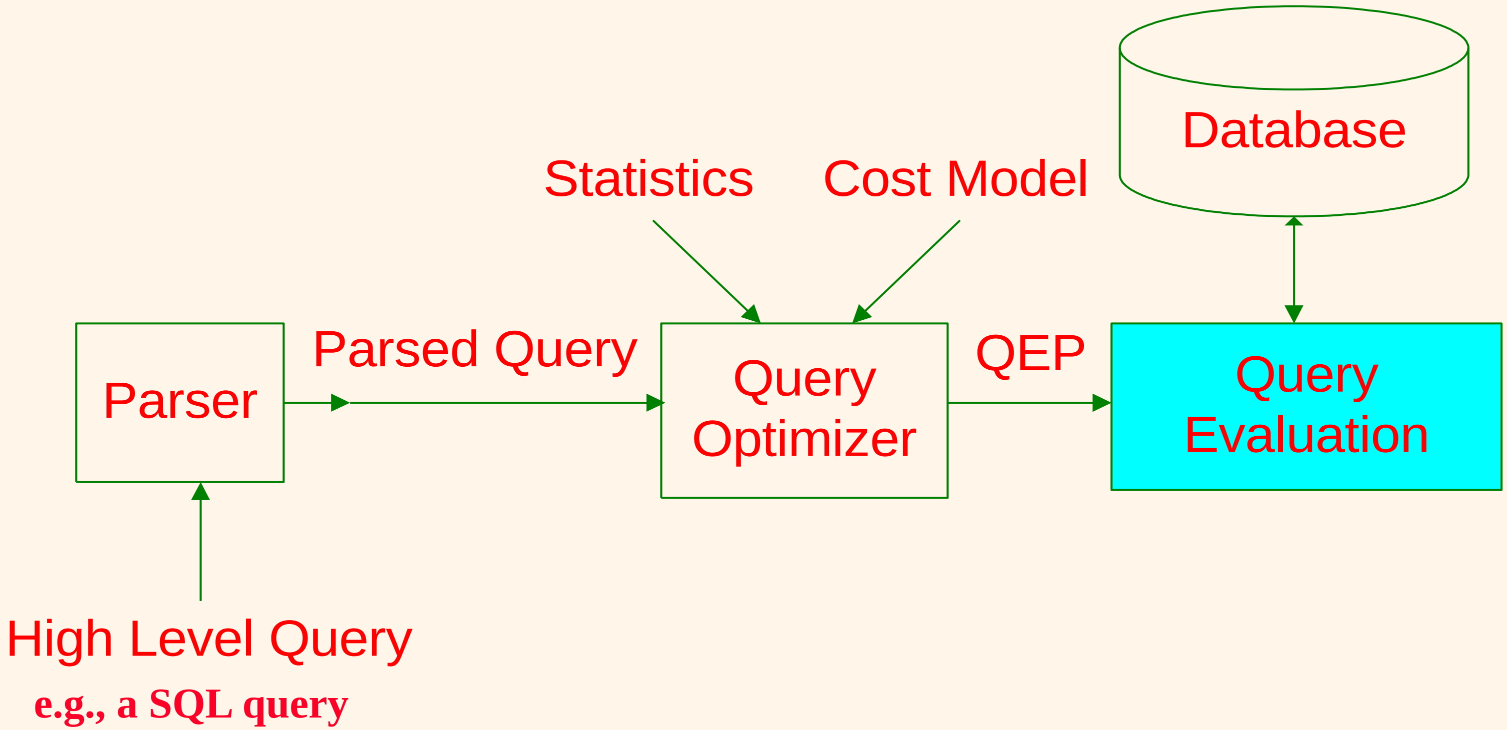
COMP7640
Database Systems & Administration

Query Evaluation

Where Are We Now?



Processing a High-Level Query



- **Query Evaluation**: evaluate basic relational operators (possibly based on indexes available)
- **Query Optimization**: construct a QEP (query evaluation/ execution plan) that minimizes the cost of query evaluation

Rational Operators

- Rational operators are building blocks for query evaluation
  - Selection σ
    - Selects a subset of rows from relation
  - Projection π
    - Deletes unwanted columns from relation
  - Join $\bowtie$
    - Combine two relations

Each relational operator returns a relation!

Example Instances

Students (sid: *integer*, sname: *string*, gpa: *real*, age: *integer*)
CourseEnrolled (sid: *integer*, cid: *string*, day: *date*)

- "Students" and "CourseEnrolled" relations for our examples

S

sid	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21
58	karen	3.5	18

E

sid	cid	day
22	2440	10/01/04
22	3820	10/12/03
58	3820	11/01/04

Projection

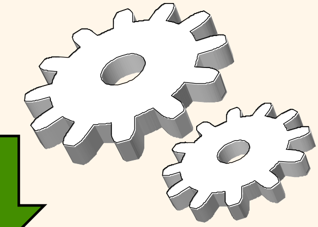
- Deletes attributes that are not in *projection list*.
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the input relation.

sid	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21
58	karen	3.5	18

$$\pi_{sname,\ gpa}(S)$$

sname	gpa
simon	3.6
kelvin	3.5
karen	3.5

Projection



- Deletes attributes that are not in *projection list.*
- *Schema* of result contains exactly the fields in the projection list, with the same names that they had in the input relation.
- *Duplicates*
  - Real systems typically don't do duplicate elimination
  - unless the user explicitly asks for it

<u>sid</u>	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21
58	karen	3.5	18

$\pi_{gpa}(S)$

gpa
3.6
3.5

Selection

- Selects rows that satisfy *selection condition*
- *Schema* of result identical to schema of input relation
- *Result* relation can be the *input* for another relational algebra operation! (***Operator composition***)

sid	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21
58	karen	3.5	18

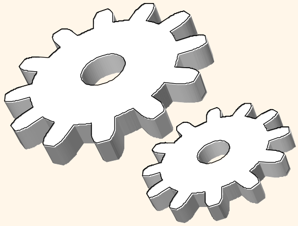
$\sigma_{age > 18}(S)$

sid	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21

$\pi_{sname,\ gpa}(\sigma_{age > 18}(S))$

sname	gpa
simon	3.6
kelvin	3.5

Joins



- *Join*: $R \bowtie_c S = \sigma_c(R \times S)$

sid	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21
58	karen	3.5	18

sid	cid	day
22	2440	10/01/04
22	3820	10/12/03
58	3820	11/01/04

$$S \bowtie_{S.sid = E.sid} E$$

sid	sname	gpa	age	(sid)	cid	date
22	simon	3.6	20	22	2440	10/01/04
22	simon	3.6	20	22	3820	10/12/03
58	karen	3.5	18	58	3820	11/01/04

- ***Result schema*** **same as that of cross-product**
  - **Fewer records than cross-product**

Questions 1-3

S

<u>sid</u>	sname	gpa	age
42	David	4.0	21
15	Louis	2.8	19
98	Amy	1.7	20

- (Question 1) Find the result of this query:

  $$\sigma_{gpa > 2.6 \wedge gpa \leq 4.0} (\sigma_{age \geq 19 \wedge age \leq 20} (S))$$

- (Question 2) Find the result of this query:

  $$\sigma_{age \geq 19 \wedge age \leq 20} (\sigma_{gpa > 2.6 \wedge gpa \leq 4.0} (S))$$

- (Question 3) Find the result of this query:

  $$\sigma_{age \leq 19} (\sigma_{gpa > 2.8} (S))$$

Questions 4 & 5

S

sid	sname	gpa	age
42	David	4.0	21
15	Louis	2.8	19
98	Amy	1.7	20

- (Question 4) Suppose that we reverse the order of two selection operators, can it affect the final answer?
- (Question 5) Find the result of this query:

$$\pi_{age}(\sigma_{gpa > 2.6 \wedge gpa \leq 4.0}(S))$$

Questions 6 & 7

S

sid	sname	gpa	age
42	David	4.0	21
15	Louis	2.8	19
98	Amy	1.7	20

E

sid	cid	day
15	2016	01/09/18
15	2006	12/01/18
42	4035	11/01/20

- (Question 6) Find the result of this query:

$$S \bowtie_{S.sid = E.sid} E$$

- (Question 7) Find the result of this query:

$$S \bowtie_{S.sid = E.sid} (\sigma_{cid = 4035} (E))$$

Measures of Query Cost

- Use the number of page transfers from disk
  - Disk accesses are much slower compared to in-memory operations.
  - Assume that the memory can hold a few pages of data

Evaluation of Relational Operators

- **Access path:** Alternative ways to retrieve records from a relation
  - Example 1: Scan the entire relation
  - Example 2: Use an index (e.g., B+ tree, hash index)

- **Selectivity** of access path
  - Number of pages retrieved for evaluating the query
    - Index pages + data pages
  - **Goal**: minimizes retrieval cost

Selection

- Selection query: $\sigma_{\text{R.attr op value}}(\text{R})$
  - SELECT * FROM Students S

    WHERE S.age > 18

Access Paths for Selection

- Selection query: $\sigma_{\text{R.attr op value}}(\text{R})$
  - SELECT * FROM Students S

    WHERE S.age > 18
- Access Paths
  - No index, unsorted data
    - Scan the whole relation
    - Cost = M (number of pages in S)
  - No index, sorted data
    - Binary search to locate the first record satisfying the selection condition
    - Cost = O ($\log_2$ M) + IOs to retrieve remaining matched records

$=\lceil \log_2 M \rceil$ or $\lfloor \log_2 M \rfloor$

Example: $\sigma_{age > 20}(S)$

- No index, unsorted data (each record takes a page)

	<u>sid</u>	sname	gpa	age
Page 1	22	simon	3.6	20
Page 2	31	kelvin	3.5	21
Page 3	58	karen	3.5	18

full scan

<u>sid</u>	sname	gpa	age
31	kelvin	3.5	21

Cost = 3 Pages

Example: $\sigma_{age > 20}(S)$

- No index, sorted data (each record takes a page)

	sid	sname	gpa	age
Page 1	58	karen	3.5	18
Page 2	22	simon	3.6	20
Page 3	31	kelvin	3.5	21

binary search

sid	sname	gpa	age
31	kelvin	3.5	21

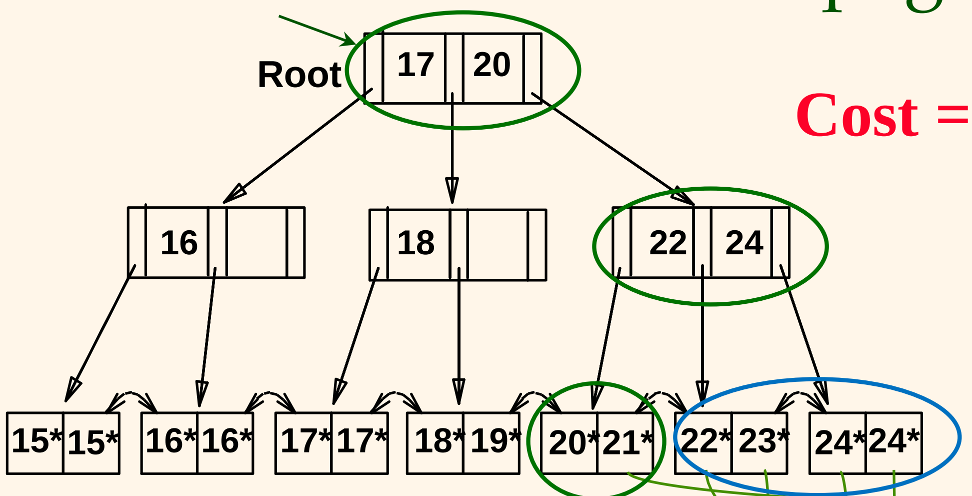
Cost = 2 Pages

Access Paths for Selection

- B+ tree
  - Use index to find first index entry that points to a *matched* record of S
  - Scan leaf pages of index to retrieve all entries in which key value satisfies selection condition
  - Retrieve corresponding data records
  - Cost = generally 2~3 I/Os (tree height) to get *starting* leaf page +
    I/Os to retrieve subsequent *qualifying* leaf pages +
    I/Os to retrieve all *matched* records
- Hash index
  - Equality selection
    - 1 I/O + I/Os to retrieve all *matched* records

Example: $\sigma_{age > 20}(S)$

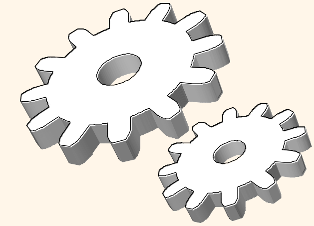
- B+ tree Index (each record takes a page, two index entries take a page)



Cost = 3 Pages + 2 Pages + 5 Pages = 10 pages

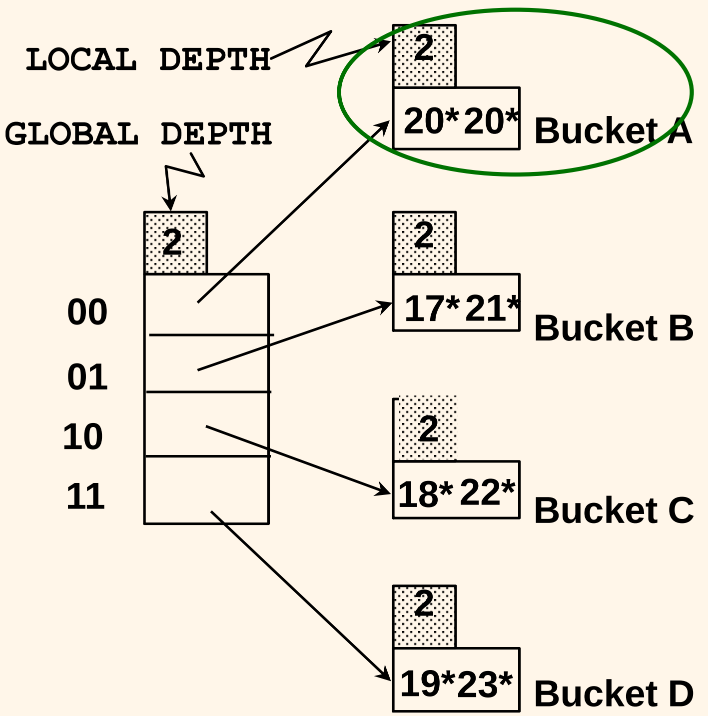
sid	sname	gpa	age
22	simon	3.6	20
31	kelvin	3.5	21
58	karen	3.5	18
62	jack	3.3	22
65	david	3.2	16
71	alex	3.3	24
73	cindy	3.7	23
75	mike	3.6	24
…	…	…	…

14 pages



Example: $\sigma_{age = 20}(S)$

- Hash Index (each record takes a page, two index entries take a page)

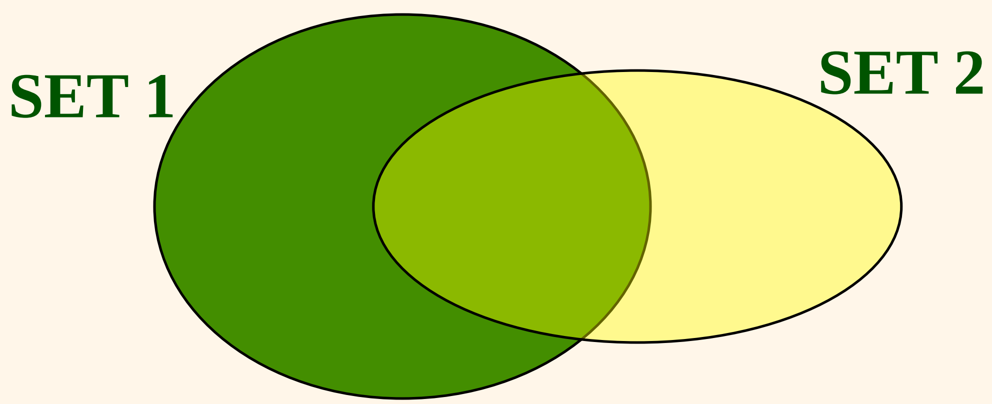


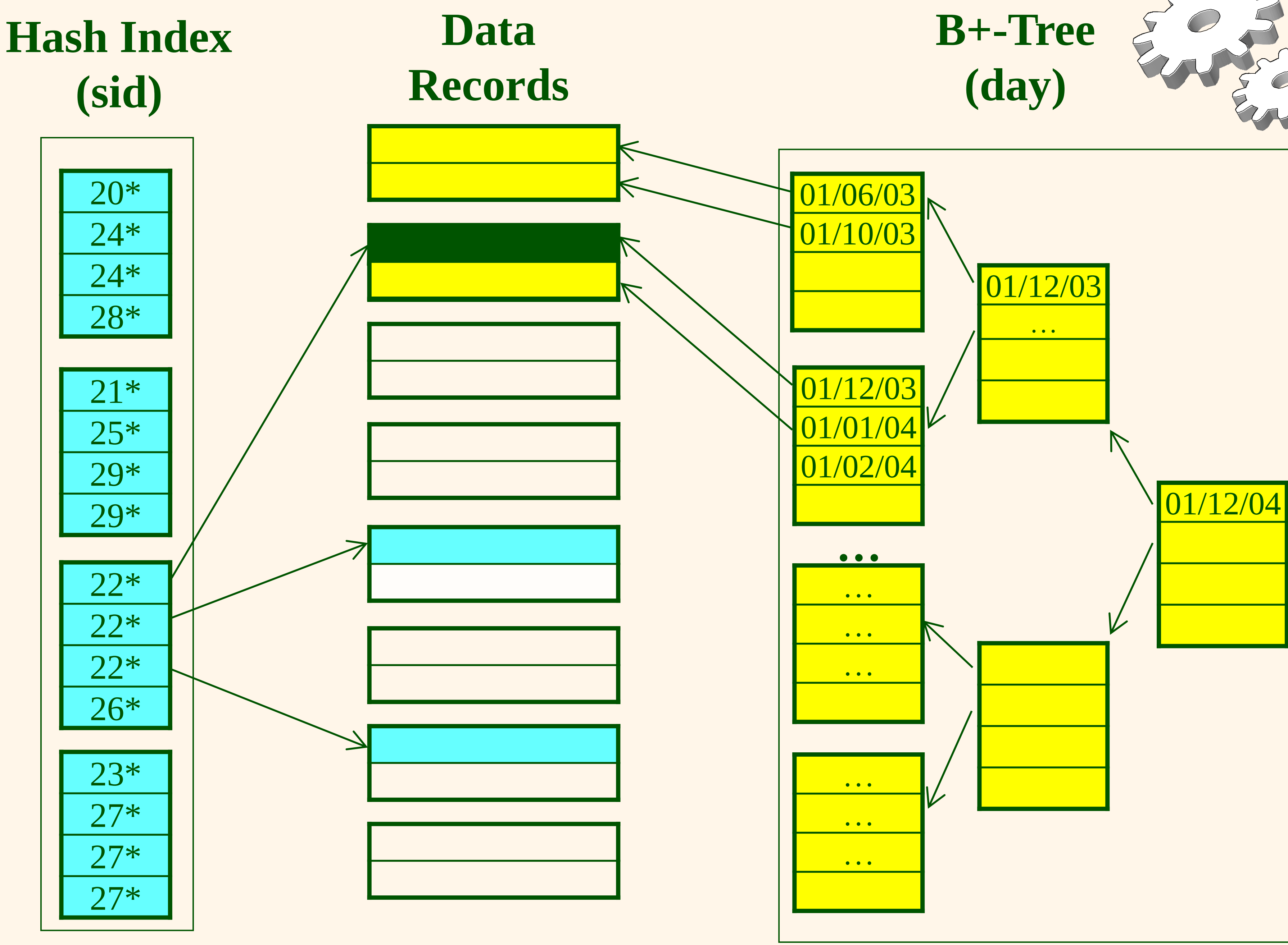
sid	sname	gpa	age
22	bob	3.6	20
31	kathy	3.5	21
58	sam	3.5	18
62	mike	3.3	22
65	dana	3.2	17
71	alen	3.3	20
73	alice	3.7	23
75	jack	3.6	19
…	…	…	…

Cost = 1 Page + 2 Pages
= 3 pages

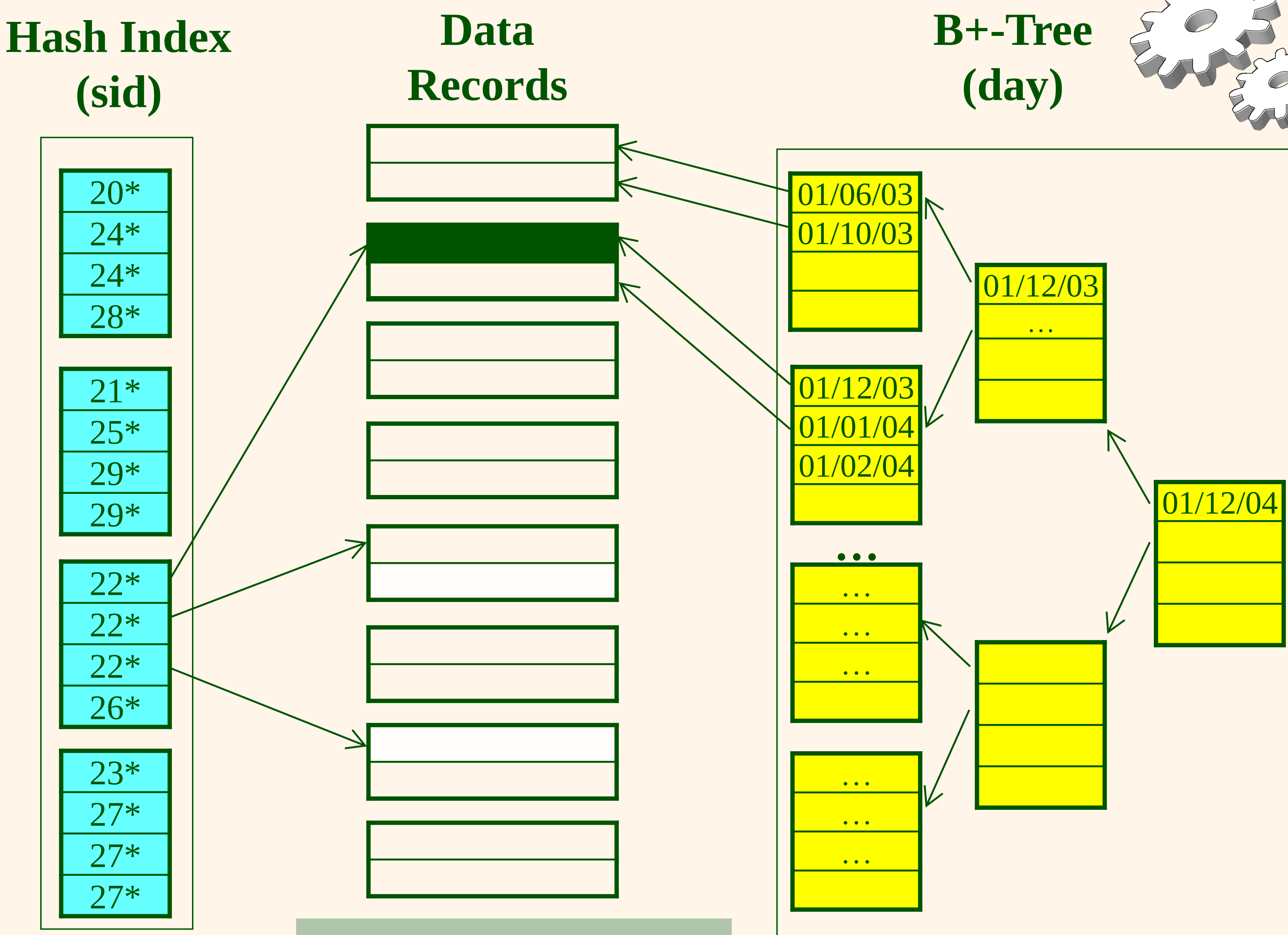
Access Paths for Selection

- Example: **sid = 22 AND cid = 4035 AND day<15/01/04**
  - Hash index on search key **<sid>**
    B+ tree index on **<day>**
  - Retrieve rids satisfying **sid = 22** using the hash index
  - Retrieve rids satisfying **day<15/01/04** using the B+ tree index
  - Intersect the set of rids, retrieve records, check **cid = 4035**





Hash Index
(sid)
Data
Records
B+-Tree
(day)
20*
24*
24*
28*
21*
25*
29*
29*
22*
22*
22*
26*
23*
27*
27*
27*
01/06/03
01/10/03
01/12/03
01/01/04
01/02/04
01/12/03
...
01/12/04
...
...
...
...
...
...



check cid = 4035

Join

- Join query: S $\bowtie$ E
  - SELECT *

    FROM Students S, CourseEnrolled E

    WHERE S.sid = E.sid
- Join operation can be implemented by
  - Cross-product: S $\times$ E
  - Followed by selections and projections
  - Inefficient
    - Cross-product much larger than result of a join

Access Paths for Join

- Techniques to implement join
  - Iteration
    - Simple Nested Loops
    - Page-oriented nested loops join
  - Partition
    - Sort Merge Join
  - Indexing
    - Index Nested Loops

Join

- Assume
  - ***M*** pages in **S**, p_S records per page
  - ***N*** pages in **E**, p_E records per page
  - If **S** is **Students** and **E** is **CourseEnrolled**
    - $M = 1000$, $p_S = 100$
    - $N = 500$, $p_E = 400$
- Cost metric
  - Number of I/Os

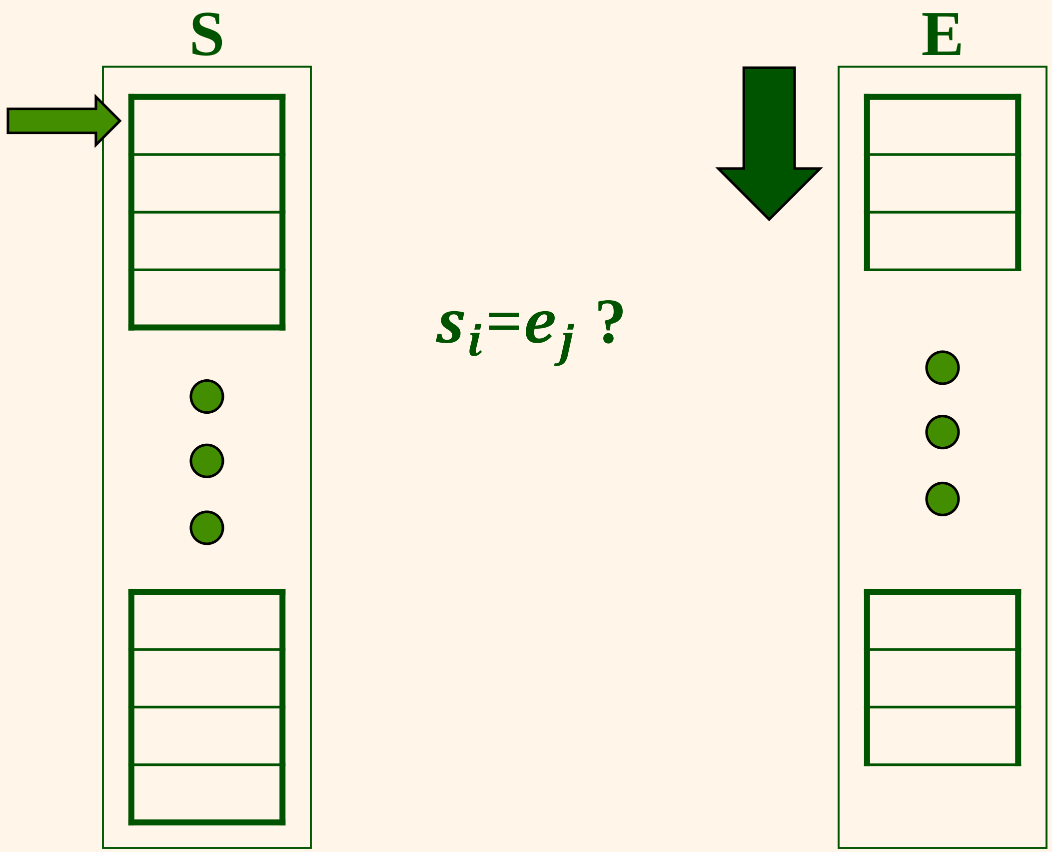
Access Path 1: Simple Nested Loops Join

foreach record s_i in S do
    foreach record e_j in E do
      if $s_i == e_j$ then add $< s_i, e_j >$ to result

- Scan outer relation *S*
- For each record *s* in *S*, scan entire inner relation *E*

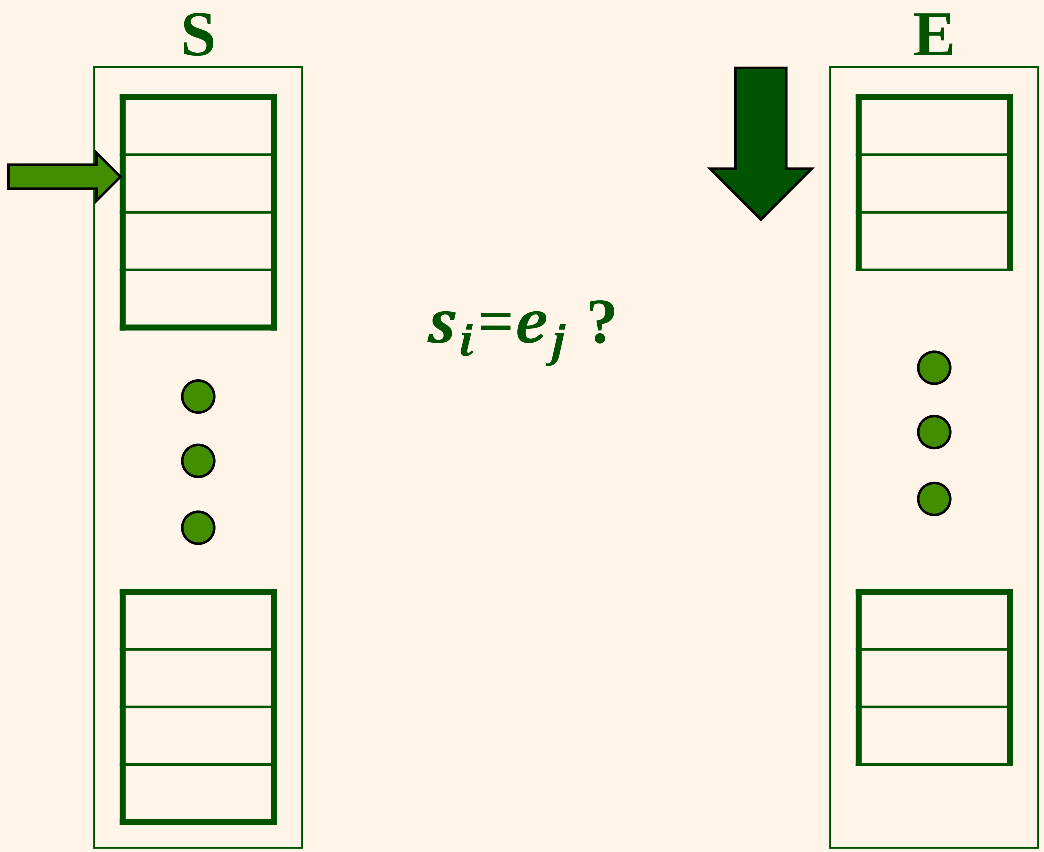
Access Path 1: Simple Nested Loops Join

foreach record s_i in S do
    foreach record e_j in E do
      if $s_i == e_j$ then add $< s_i, e_j >$ to result



Access Path 1: Simple Nested Loops Join

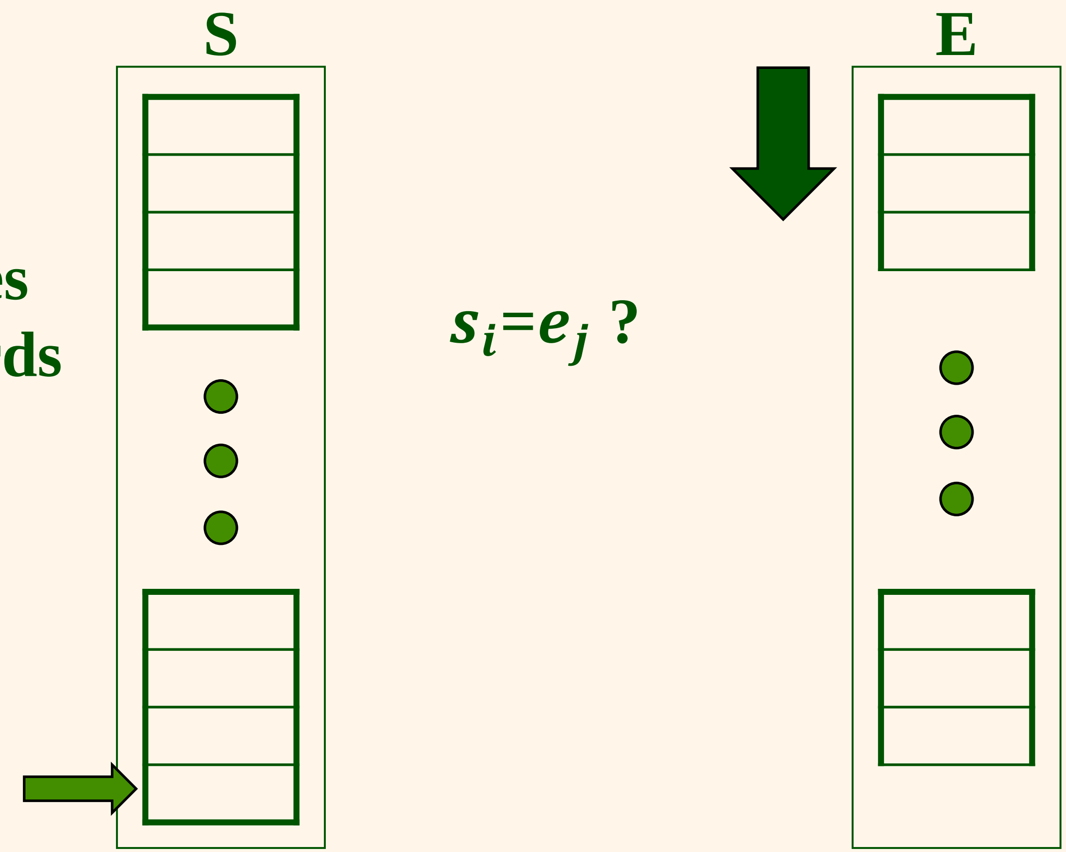
foreach record s_i in S do
    foreach record e_j in E do
        if $s_i == e_j$ then add $< s_i, e_j >$ to result



Access Path 1: Simple Nested Loops Join

foreach record s_i in S do
    foreach record e_j in E do
        if $s_i == e_j$ then add $< s_i, e_j >$ to result

M pages
p_S records



N pages

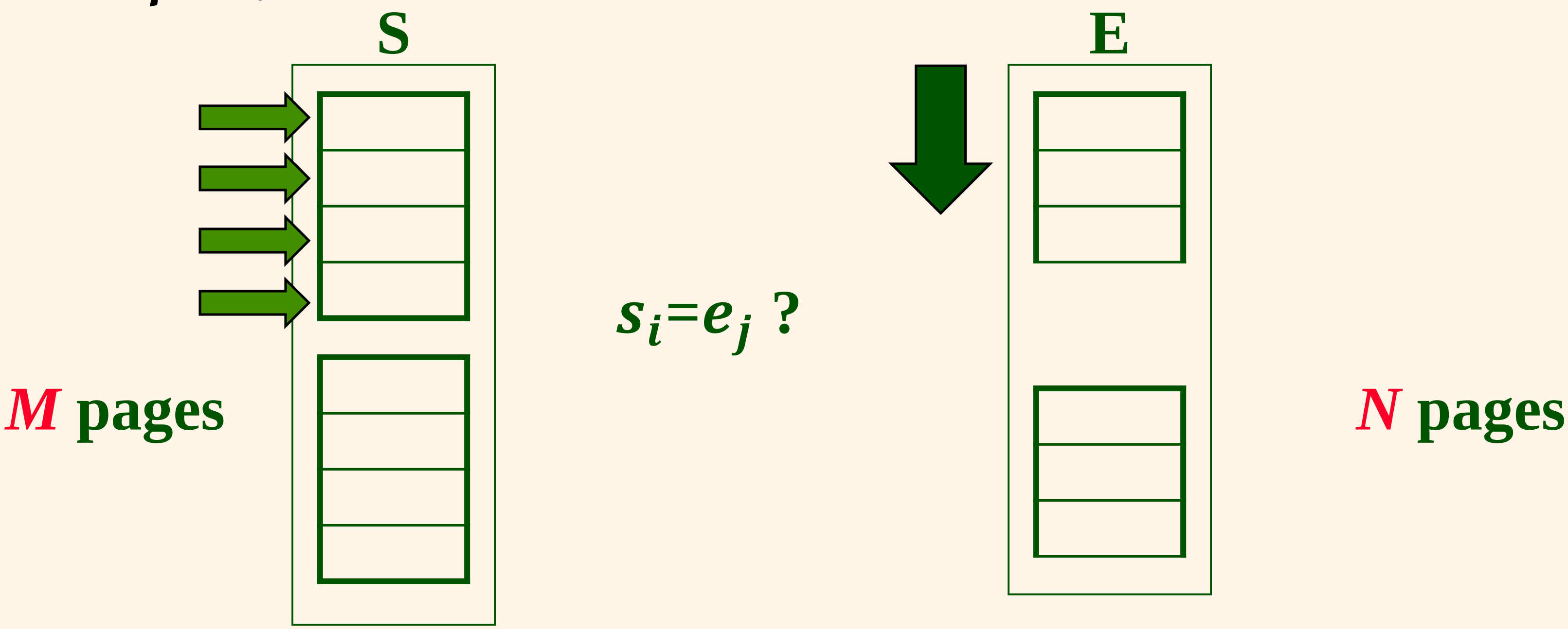
$M = 1000,\ p_S = 100$
$N = 500,\ p_E = 400$

Total cost = $M + p_S * M * N$ = 1000 +100 * 1000 * 500 I/Os

Access Path 2: Page-oriented Nested Loops Join

- Page-oriented nested loops join
  - For each page of *S*, retrieve each page of *E*
  - Write out matching pairs of records <*s*, *e*>

Access Path 2: Page-oriented Nested Loops Join



- Page-oriented nested loops join
  - Total cost = *M* + *M* * *N*

    = 1000 + 1000 * 500
  - Choose smaller relation to be outer relation
    - E.g., *E* be outer relation
    - Total cost = 500 + 500 * 1000

Example: $S \bowtie_{S.sid=E.sid} E$

- Access Path 1: Simple Nested Loops Join

S

	sid	sname	gpa	age
Page 1	22	simon	3.6	20
	31	kelvin	3.5	21
Page 2	58	karen	3.5	18
	66	kelly	3.4	19

$M = 2, p_S = 2$

E

sid	cid	day	
22	2440	10/01/04	**Page 1**
66	2440	09/01/04	
22	3820	10/12/03	**Page 2**
58	3820	11/01/04	
58	4035	11/01/04	**Page 3**
66	4035	12/01/04	

$N = 3, p_E = 2$

Total cost = $M + p_S * M * N = 2 + 2*2*3 = 14$ ***pages***

Example: $S \bowtie_{S.sid=E.sid} E$

- Access Path 2: Page-oriented Nested Loops Join

S

	sid	sname	gpa	age
Page 1	22	simon	3.6	20
	31	kelvin	3.5	21
Page 2	58	karen	3.5	18
	66	kelly	3.4	19

$M = 2, p_S = 2$

E

sid	cid	day	
22	2440	10/01/04	**Page 1**
66	2440	09/01/04	
22	3820	10/12/03	**Page 2**
58	3820	11/01/04	
58	4035	11/01/04	**Page 3**
66	4035	12/01/04	

$N = 3, p_E = 2$

Total cost = *$M + M * N = 2 + 2*3 = 8$ pages*

Question 8

- Consider the join operation R $\bowtie_{R.a=S.b}$ S. Given the following information about these two relations.
  - Relation R contains 10,000 records and has 10 records per page.
  - Relation S contains 2,000 records and has 10 records per page.
  - No index structure has been built for these two relations.

1) What is the lowest cost of joining R and S using a page-oriented nested loop join?
2) What is the lowest cost of joining R and S using a simple-nested loop join?

[Remark: You can ignore the I/O cost for writing the result back to the disk.]

Projection

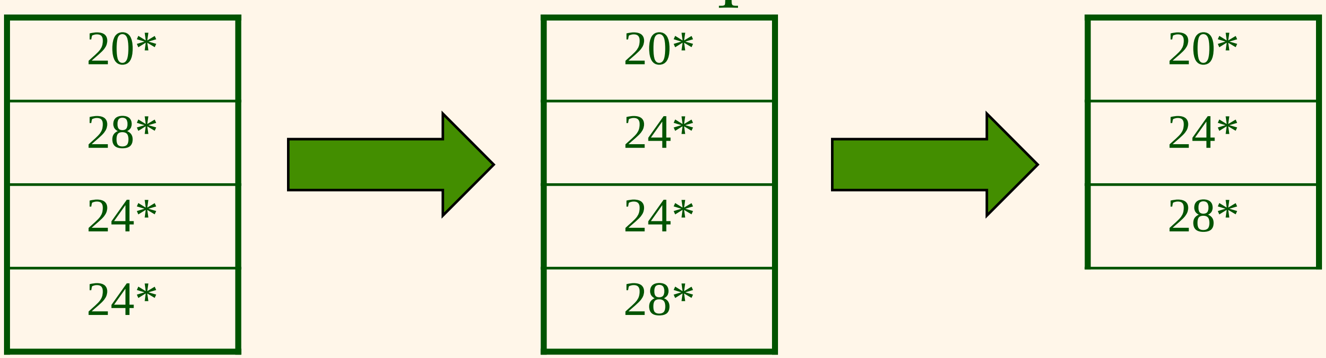
- Projection query: $\pi_{\text{R.attr(s)}}(R)$
  - SELECT S.gpa
    FROM Students S
  - SELECT DISTINCT S.gpa
    FROM Students S
- To implement projection
  - Remove *unwanted* attributes
  - Eliminate any *duplicate* records

Access Paths for Projection

- Projection query: $\pi_{\text{R.attr(s)}}(R)$
  - SELECT S.gpa
    FROM Students S
  - SELECT DISTINCT S.gpa
    FROM Students S
- To implement projection
  - Remove unwanted attributes
  - Eliminate any duplicate records
- Access Paths:
  - Sort-based projection

Sort-Based Projection

- **Step 1** – Scan S to produce a set of records containing only the desired attributes
- **Step 2** - Sort records using combination of attributes as key
- **Step 3** – Scan sorted result, compare adjacent records and discard duplicates



Sort-Based Projection

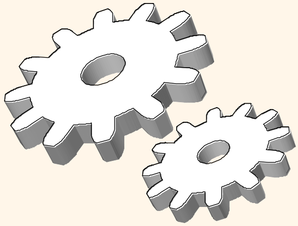
- **Step 1** – Scan S to produce a set of records containing only the desired attributes
  - *M* is # pages of S, *T* is # pages of temp relation S′
  - Cost = *M* I/Os to scan S + *T* I/Os to write S′
- **Step 2** - Sort records using a combination of attributes as key (Using external sorting with *B* buffer pages in the main memory)
  - Cost = $2T * \lceil \log_{B-1} T \rceil$
- **Step 3** – Scan sorted result, compare adjacent records and discard duplicates
  - Cost = *T*

Example

- Projection on Students relation
  - Each record is 40 bytes long, 100 records per page, 1000 pages.
- Sort-based projection
  - Step 1:
    - Scan Students with 1000 I/Os
    - If a record in S′ is 10 bytes, 250 I/O to write out S′
  - Step 2:
    - Given 20 buffer pages, sort S′ in 2 passes at a cost of (2*250 *2) I/Os
  - Step 3:
    - 250 I/Os to scan for duplicates
  - Total cost: 2500 I/Os

Question 9

- Consider the relation S(a, b, c, d, e) with 5,000,000 records, where a, b, c, d and e take 3 bytes, 1 byte, 2 bytes, 4 bytes, and 2 bytes, respectively. Suppose that each page can store 4,096 bytes and the query processor uses the *sort-based algorithm* as an access path to evaluate every projection operation. Assume that we have 101 buffer pages in the main memory. What is the I/O cost for evaluating $\pi_{a,b,c}(S)$?



Summary

- Queries are composed of basic operators.
- Access paths are the alternative ways to retrieve records from a relation.
- Index matches selection condition if it can be used to only retrieve records that satisfy selection condition.
- Selectivity of an access path with respect to a query is the total number of I/O costs that is needed to solve this query.

Solutions

- Question 1:

<u>sid</u>	sname	gpa	age
15	Louis	2.8	19

- Question 2:

<u>sid</u>	sname	gpa	age
15	Louis	2.8	19

- Question 3:

<u>sid</u>	sname	gpa	age

Solutions

- Question 4: It cannot.

- Question 5:

age
21
19

Solutions

- Question 6:

<u>sid</u>	sname	gpa	age	cid	day
42	David	4.0	21	4035	11/01/20
15	Louis	2.8	19	2016	01/09/18
15	Louis	2.8	19	2006	12/01/18

- Question 7:

<u>sid</u>	sname	gpa	age	cid	day
42	David	4.0	21	4035	11/01/20

Solution to Question 8

- Consider the join operation $R \bowtie_{R.a=S.b} S$. Given the following information about these two relations.
  - Relation R contains 10,000 records and has 10 records per page.
  - Relation S contains 2,000 records and has 10 records per page.
  - No index structure has been built for these two relations.

1) What is the lowest cost of joining R and S using a page-oriented nested loop join?

The number of pages in R is: 10,000/10 = 1,000

The number of pages in S is: 2,000/10 = 200

In order to achieve the lowest cost of joining these two relations, we need to put the table with the smaller number of pages in the outer loop.

The I/O cost of the page-oriented nested loop join is:

$$200 + 200 \times 1{,}000 = 200{,}200 \text{ I/Os}$$

Solution to Question 8

- Consider the join operation $R \bowtie_{R.a=S.b} S$. Given the following information about these two relations.
  - Relation R contains 10,000 records and has 10 records per page.
  - Relation S contains 2,000 records and has 10 records per page.
  - No index structure has been built for these two relations.

2) What is the lowest cost of joining R and S using a simple-nested loop join?

The number of pages in R is: 10,000/10 = 1,000

The number of pages in S is: 2,000/10 = 200

There are two options when using simple-nested loop join:

When R is outer relation: I/O cost is 1,000 + 10×1000×2,00

When S is outer relation: I/O cost is 200 + 10×200 ×10,00

So, the minimum I/O cost is:

min(1,000 + 10,000×2,00, 200 + 2,000 ×10,00) = 2,000,200 I/Os

Solution to Question 9

- We need to scan this relation S. The cost is

  $\lceil (5{,}000{,}000 \times (3 + 1 + 2 + 4 + 2))/4{,}096 \rceil$=14,649 I/Os
- Then, we need to write the temporary results to the disk. The cost is is

  $\lceil (5{,}000{,}000 \times (3 + 1 + 2\,))/4{,}096 \rceil$ = 7,325 I/Os
- After that, we need to sort these records. The cost is:

  $2 \times 7325 \times \lceil \log_{100} 7325 \rceil$= 29,300 I/Os.
- Lastly, we scan the sorted results and discard the duplicates. The cost is: 7,325 I/Os.
- The total I/O cost is: 14,649 + 7,325 + 29,300 + 7,325 = 58,599 I/Os.