CSE 344

APRIL 25TH – DISK I/O
• HW4 Due Tonight
• OQ5 Due Tonight
• HW5 Out Tonight
  • SQL++
  • Due next Wednesday, 11:30
WHICH INDEXES?

The *index selection problem*

- Given a table, and a “workload” (big Java application with lots of SQL queries), decide which indexes to create (and which ones NOT to create!)

**Who does index selection:**

- The database administrator DBA

- Semi-automatically, using a database administration tool

<table>
<thead>
<tr>
<th>ID</th>
<th>fName</th>
<th>lName</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Tom</td>
<td>Hanks</td>
</tr>
<tr>
<td>20</td>
<td>Amy</td>
<td>Hanks</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INDEX SELECTION: WHICH SEARCH KEY

Make some attribute K a search key if the WHERE clause contains:

- An exact match on K
- A range predicate on K
- A join on K
THE INDEX SELECTION
PROBLEM 1

V(M, N, P);

Your workload is this

100000 queries:

SELECT * 
FROM V
WHERE N=?

100 queries:

SELECT * 
FROM V
WHERE P=?
THE INDEX SELECTION PROBLEM 1

V(M, N, P);

Your workload is this

100000 queries:

SELECT *
FROM V
WHERE N=?

100 queries:

SELECT *
FROM V
WHERE P=?

What indexes?
THE INDEX SELECTION PROBLEM 1

V(M, N, P);

Your workload is this

100000 queries:

100 queries:

SELECT *
FROM V
WHERE N=?

SELECT *
FROM V
WHERE P=?

A: V(N) and V(P) (hash tables or B-trees)
THE INDEX SELECTION PROBLEM 2

V(M, N, P);

Your workload is this

100000 queries:

SELECT * FROM V WHERE N>? and N<?

100 queries:

SELECT * FROM V WHERE P=?

100000 queries:

INSERT INTO V VALUES (?, ?, ?)

What indexes?
THE INDEX SELECTION
PROBLEM 2

V(M, N, P);

Your workload is this

100000 queries:

SELECT *
FROM V
WHERE N>? and N<?

100 queries:

SELECT *
FROM V
WHERE P=?

100000 queries:

INSERT INTO V
VALUES (?, ?, ?)

A: definitely V(N) (must B-tree); unsure about V(P)
THE INDEX SELECTION
PROBLEM 3

V(M, N, P);

Your workload is this

100000 queries:
SELECT * FROM V WHERE N=?

1000000 queries:
SELECT * FROM V WHERE N=? and P>?

100000 queries:
INSERT INTO V VALUES (?, ?, ?)

What indexes?
THE INDEX SELECTION PROBLEM 3

V(M, N, P);

Your workload is this

100000 queries: SELECT * FROM V WHERE N=?

1000000 queries: SELECT * FROM V WHERE N=? and P>?

100000 queries: INSERT INTO V VALUES (?, ?, ?)

How does this index differ from:
1. Two indexes V(N) and V(P)?
2. An index V(P, N)?

A: V(N, P)
THE INDEX SELECTION
PROBLEM 4

V(M, N, P);

Your workload is this

1000 queries:

SELECT *
FROM V
WHERE N>? and N<?

100000 queries:

SELECT *
FROM V
WHERE P>? and P<?

What indexes?
THE INDEX SELECTION PROBLEM 4

V(M, N, P);

Your workload is this

1000 queries:

```
SELECT * 
FROM V 
WHERE N>? and N<?
```

100000 queries:

```
SELECT * 
FROM V 
WHERE P>? and P<?
```

A: V(N) unclustered, V(P) clustered index
TWO TYPICAL KINDS OF QUERIES

• Point queries

• Range queries

SELECT * FROM Movie WHERE year = ?

SELECT * FROM Movie WHERE year >= ? AND year <= ?

• What data structure should be used for index?

• What data structure should be used for index?
BASIC INDEX SELECTION GUIDELINES

Consider queries in workload in order of importance

Consider relations accessed by query
  • No point indexing other relations

Look at WHERE clause for possible search key

Try to choose indexes that speed-up multiple queries
TO CLUSTER OR NOT

Range queries benefit mostly from clustering

Point indexes do not need to be clustered: they work equally well unclustered
SELECT * 
FROM R 
WHERE R.K>? and R.K<?
SELECT * 
FROM R 
WHERE R.K>? and R.K<?
Cost

Percentage tuples retrieved

Sequential scan

Clustered index

SELECT * 
FROM R
WHERE R.K>? and R.K<?
**Percentage tuples retrieved**

**Cost**

- **Sequential scan**
- **Unclustered index**
- **Clustered index**

**SQL Query**

```sql
SELECT * 
FROM R 
WHERE R.K>?? and R.K<??
```
CHOOSING INDEX IS NOT ENOUGH

To estimate the cost of a query plan, we still need to consider other factors:

- How each operator is implemented
- The cost of each operator
- Let’s start with the basics
COST PARAMETERS

Cost = I/O + CPU + Network BW

- We will focus on I/O in this class

Parameters (a.k.a. statistics):

- \( B(R) \) = # of blocks (i.e., pages) for relation \( R \)
- \( T(R) \) = # of tuples in relation \( R \)
- \( V(R, a) \) = # of distinct values of attribute \( a \)
COST PARAMETERS

Cost = I/O + CPU + Network BW

• We will focus on I/O in this class

Parameters (a.k.a. statistics):

• $B(R)$ = # of blocks (i.e., pages) for relation $R$
• $T(R)$ = # of tuples in relation $R$
• $V(R, a)$ = # of distinct values of attribute $a$

When $a$ is a key, $V(R,a) = T(R)$
When $a$ is not a key, $V(R,a)$ can be anything $\leq T(R)$
COST PARAMETERS

Cost = I/O + CPU + Network BW

- We will focus on I/O in this class

Parameters (a.k.a. statistics):

- $B(R) = \#$ of blocks (i.e., pages) for relation $R$
- $T(R) = \#$ of tuples in relation $R$
- $V(R, a) = \#$ of distinct values of attribute $a$

When $a$ is a key, $V(R,a) = T(R)$
When $a$ is not a key, $V(R,a)$ can be anything $\leq T(R)$

DBMS collects statistics about base tables
must infer them for intermediate results
SELECTIVITY FACTORS FOR CONDITIONS

\[ A = c \]

\[ /* \sigma_{A=\text{c}(R)} */ \]

- Selectivity \( = \frac{1}{V(R,A)} \)

\[ A < c \]

\[ /* \sigma_{A<\text{c}(R)} */ \]

- Selectivity \( = \frac{c - \min(R, A)}{\max(R, A) - \min(R, A)} \)

\[ c_1 < A < c_2 \]

\[ /* \sigma_{c_1<\text{A}<c_2}(R) */ \]

- Selectivity \( = \frac{c_2 - c_1}{\max(R, A) - \min(R, A)} \)
COST OF READING
DATA FROM DISK

Sequential scan for relation R costs $B(R)$

Index-based selection

- Estimate selectivity factor $f$ (see previous slide)
- Clustered index: $f^*B(R)$
- Unclustered index $f^*T(R)$

Note: we ignore I/O cost for index pages
INDEX BASED SELECTION

Example:

\[
\begin{align*}
B(R) &= 2000 \\
T(R) &= 100,000 \\
V(R, a) &= 20
\end{align*}
\]

Table scan:

Index based selection:

\[ \text{cost of } \sigma_{a=v}(R) = ? \]
INDEX BASED SELECTION

Example:

\[
\begin{align*}
B(R) &= 2000 \\
T(R) &= 100,000 \\
V(R, a) &= 20
\end{align*}
\]

cost of \( \sigma_{a=v}(R) = ? \)

Table scan: \( B(R) = 2,000 \) I/Os

Index based selection:
INDEX BASED SELECTION

Example:

\[
\begin{align*}
B(R) &= 2000 \\
T(R) &= 100,000 \\
V(R, a) &= 20
\end{align*}
\]

Table scan: \(B(R) = 2,000\) I/Os

Index based selection:

- If index is clustered:
- If index is unclustered:

\[
\text{cost of } \sigma_{a=v}(R) = ?
\]
INDEX BASED SELECTION

Example:

\[ B(R) = 2000 \]
\[ T(R) = 100,000 \]
\[ V(R, a) = 20 \]

Table scan: \( B(R) = 2,000 \) I/Os

Index based selection:

- If index is clustered: \( B(R) \times 1/V(R,a) = 100 \) I/Os
- If index is unclustered:

\[ \text{cost of } \sigma_{a=v}(R) = ? \]
INDEX BASED SELECTION

Example:

B(R) = 2000
T(R) = 100,000
V(R, a) = 20

Table scan: B(R) = 2,000 I/Os

Index based selection:

• If index is clustered: B(R) * 1/V(R,a) = 100 I/Os
• If index is unclustered: T(R) * 1/V(R,a) = 5,000 I/Os

cost of \( \sigma_{a=v}(R) = ? \)
INDEX BASED SELECTION

Example:

| B(R) = 2000 |
| T(R) = 100,000 |
| V(R, a) = 20 |

cost of \( \sigma_{a=v}(R) = ? \)

Table scan: B(R) = 2,000 I/Os

Index based selection:

- If index is clustered: \( B(R) \times 1/V(R,a) = 100 \) I/Os
- If index is unclustered: \( T(R) \times 1/V(R,a) = 5,000 \) I/Os

Lesson: Don’t build unclustered indexes when \( V(R,a) \) is small!
OUTLINE

Join operator algorithms

- One-pass algorithms (Sec. 15.2 and 15.3)
- Index-based algorithms (Sec 15.6)

Note about readings:

- In class, we discuss only algorithms for joins
- Other operators are easier: read the book
JOIN ALGORITHMS

Hash join

Nested loop join

Sort-merge join
**HASH JOIN**

Hash join: R $\bowtie$ S

Scan R, build buckets in main memory

Then scan S and join

Cost: B(R) + B(S)

Which relation to build the hash table on?
**HASH JOIN**

Hash join:  \( R \bowtie S \)

Scan \( R \), build buckets in main memory

Then scan \( S \) and join

Cost: \( B(R) + B(S) \)

Which relation to build the hash table on?

One-pass algorithm when \( B(R) \leq M \)

- \( M = \) number of memory pages available
**HASH JOIN EXAMPLE**

Patient(pid, name, address)
Insurance(pid, provider, policy_nb)

Patient $\bowtie$ Insurance

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>Blue</td>
</tr>
<tr>
<td>Seattle</td>
<td>123</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Ela</td>
<td>Prem</td>
</tr>
<tr>
<td>Everett</td>
<td>432</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Jill</td>
<td>Prem</td>
</tr>
<tr>
<td>Kent</td>
<td>343</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Joe</td>
<td>GrpH</td>
</tr>
<tr>
<td>Seattle</td>
<td>554</td>
</tr>
</tbody>
</table>

Two tuples per page
HASH JOIN EXAMPLE

Patient □ Insurance

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2</td>
<td>2  4</td>
</tr>
<tr>
<td>3  4</td>
<td>4  3</td>
</tr>
<tr>
<td>9  6</td>
<td>2  8</td>
</tr>
<tr>
<td>8  5</td>
<td>8  9</td>
</tr>
</tbody>
</table>

Memory M = 21 pages

Showing pid only

Some large-enough #

This is one page with two tuples
HASH JOIN EXAMPLE

Step 1: Scan Patient and **build** hash table in memory
Can be done in method open()

Memory M = 21 pages
Hash h: pid % 5

Disk

Patient Insurance

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input buffer
**HASH JOIN EXAMPLE**

Step 2: Scan Insurance and **probe** into hash table
Done during calls to next()

Memory $M = 21$ pages
Hash $h: \text{pid} \% 5$

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 4</td>
</tr>
<tr>
<td>3 4</td>
<td>4 3</td>
</tr>
<tr>
<td>9 6</td>
<td>2 8</td>
</tr>
<tr>
<td>8 5</td>
<td>8 9</td>
</tr>
</tbody>
</table>

Input buffer

| 2 4 |

Output buffer

| 2 2 |

Write to disk or pass to next operator
HASH JOIN EXAMPLE

Step 2: Scan Insurance and **probe** into hash table

Done during calls to next()
HASH JOIN EXAMPLE

Step 2: Scan Insurance and probe into hash table
Done during calls to next()

Memory M = 21 pages
Hash h: pid % 5

Input buffer

Output buffer

Cost: B(R) + B(S)
NESTED LOOP JOINS

Tuple-based nested loop $R \bowtie S$

$R$ is the outer relation, $S$ is the inner relation

\[
\text{for each tuple } t_1 \text{ in } R \text{ do}
\]
\[
\text{for each tuple } t_2 \text{ in } S \text{ do}
\]
\[
\text{if } t_1 \text{ and } t_2 \text{ join then output } (t_1, t_2)
\]

What is the Cost?
NESTED LOOP JOINS

Tuple-based nested loop $R \bowtie S$

$R$ is the outer relation, $S$ is the inner relation

```
for each tuple $t_1$ in $R$ do
  for each tuple $t_2$ in $S$ do
    if $t_1$ and $t_2$ join then output ($t_1$, $t_2$)
```

Cost: $B(R) + T(R) B(S)$

Multiple-pass since $S$ is read many times
for each page of tuples r in R do
   for each page of tuples s in S do
      for all pairs of tuples t₁ in r, t₂ in s
         if t₁ and t₂ join then output (t₁,t₂)

Cost: B(R) + B(R)B(S)
PAGE-AT-A-TIME REFINEMENT

Disk

Patient

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

Insurance

| 2 | 4 |
| 6 | 6 |
| 1 | 3 |
| 2 | 8 |
| 8 | 9 |

Input buffer for Patient

| 1 | 2 |

Input buffer for Insurance

| 2 | 4 |

Output buffer

| 2 | 2 |
PAGE-AT-A-TIME REFINEMENT

Disk

Patient  Insurance
1 2 3 4 9 6 8 5

Input buffer for Patient
1 2

Input buffer for Insurance
4 3

Output buffer

2 4 6 6
4 3 1 3
2 8
8 9
PAGE-AT-A-TIME REFINEMENT

Disk

**Patient**  **Insurance**

1 2

3 4

9 6

8 5

Input buffer for Patient

1 2

2 8

Input buffer for Insurance

Keep going until read all of Insurance

Then repeat for next page of Patient… until end of Patient

Output buffer

2 2

Cost: \( B(R) + B(R)B(S) \)
BLOCK-NESTED-LOOP RENEFEMENT

Cost: \( B(R) + \frac{B(R)B(S)}{(M-1)} \)