Assignment 4

by Josh Davis Due: Monday, Oct 28

Problem 1

Consider the relation Employee:

- eid an integer
- ename a string
- sal an integer
- title a string
- age an integer

Other properties:

1. Each Employee record is 100 bytes long
2. Total Employee relation uses 10,000 pages
3. Each index data entry is 20 bytes long

Indexes using Alt 2:

1. Hash index on eid
2. Dense, unclustered B+ index on sal
3. Hash index on age
4. Dense, clustered B+ index on (age, sal)

Assumptions:

1. Data page can hold 20 Employee tuples. Each page can hold as many relations as possible. The relation is stored as a heap file.
2. One disk I/O is needed to retrieve a page.
3. Cost for retrieving all relevant internal nodes of a B+ tree from a root to desirable leaf node is 2 disk I/Os.
4. Page size is 2048 bytes, 48 bytes are reserved.
5. 1.2 I/O needed to use hash index to find a data entry that satisfies the selection criterion.
6. As many data entries as possible are stored in a page.
7. Reduction factor is .1

Basic Statistics

```
Number of records
```

```
    = (number of total pages relation uses * size of pages) / (size of
    record)
    = (10,000 * (2048 - 48)) / (100)
    = 200,000 records

Number of matching records
    = (number of records * reduction factor)
    = 200,000 * .1
    = 20,000 matching records

Entries per page
    = (page size / entry size)
    = ((2048 - 48) / 20)
    = 100

Records per page
    = (page size / record size)
    = ((2048 - 48) / 100)
    = 20
```

Part A

Query: `sal > 100`

Solution

Using B+ Index (pg 494)

Total Cost:

1. Cost of traversing from root to leaf +
2. Cost of retrieving pages in sequence +
3. Cost of retrieving pages that contain the data records

Part One

```
Traversing from root to leaf = 2
```

Part Two

```
Retrieving page in sequence
    = (number of matching records) / (entries per page)
    = 20,000 / 100
    = 200
```

Part Three

Now we just need to retrieve the records. According to the given info, for `sal` the index is dense and unclustered. This means that the tuples aren't in the same order as the entries and we might need to read every page once.

```
Retrieving pages that contain data records
    = 20,000
```

B+ Index Cost

`Total Cost = 2 + 200 + 20,000 = 20,202`

Scanning (page 493)

`Total Cost = 10,000 pages`

Final Answer

Since `10,000 < 20,202`, just scanning all the pages and selecting when `sal > 100` is more efficient.

Part B

Query: `age = 20`

Solution

Hash Index

Total Cost:

1. Cost of retrieving matching data entries +
2. Cost of retrieving qualifying tuples

Part One

```
Cost of retrieving matching data entries
= (matching entries * hash index read)
= (20,000 * 1.2)
= 24,000
```

Part Two

Each entry could point to a different page and since there are 20,000 matching records, we might have to read 20,000 pages.

Hash Index Cost

`Total Cost = 24,000 + 20,000 = 44,000`

Using B+ Index (pg 494)

Total Cost:

1. Cost of traversing from root to leaf +
2. Cost of retrieving pages in sequence +
3. Cost of retrieving pages that contain the data records

Part One

```
Cost of traversing from root to leaf
= 2
```

Part Two

```
Retrieving pages in sequence
= (number of matching records) / (entries per page)
= (20,000 / 100)
= 200
```

Part Three

Since our index is clustered on (age, sal), we will have multiple records on a page. Thus

```
Retrieving pages that contain data records
= (number of matching records) / (records per page)
= 20,000 / 20
= 1,000
```

B+ Index Cost

`Total Cost = 2 + 200 + 1,000 = 1,202`

Scanning

`Total Cost = 10,000 pages`

Final Answer

Since `1,202 < 10,00 < 44,000`, using the B+ index is the fastest way to go.

Part C

Query: `sal > 200` and `age > 30` and `title = "CFO"`

Solution

There is no index on title, thus we must scan all the pages.

B+ Index

We know this from Part A:

```
Total Cost = 2 + 200 + 20,000 = 20,202
```

Hash Index

We know this from Part B:

```
Total Cost = 24,000 + 20,000 = 44,000
```

Scanning (page 493)

`Total Cost = 10,000 pages`

B+ Index and B+ Index

Total Cost:

1. Cost of retrieving entries from first index +
2. Cost of retrieving entries from second index
3. Cost of retrieving records

The first index we use is the B+ tree that is indexed on sal The second index that we use is the B+ tree that is indexed on (age, sal)

Part One

```
Cost of retrieving entries for first index
= (cost to tree leaf) + (cost to read each entry)
= 2 + (number of matching records / entries per page)
= 2 + (20,000 / 100)
= 2 + 200
= 202
```

Part Two

```
Cost of retrieving entries for second index
= (cost to tree leaf) + (cost to read each entry)
```

```
= 2 + (number of matching records / entries per page)
= 2 + (20,000 / 100)
= 2 + 200
= 202
```

Part Three

```
Cost of retrieving records
= (number of matching records from after retrieving from index) / (records per page)
= (20,000 * .1) / (20)
= 100
```

Total Cost

`Total Cost = 202 + 202 + 100 = 504`

Final Answer

Since `504 < 20,202 < 10,000 < 44,000`, we know that using the combined B+ tree indexes gives us the fastest lookup time.

Problem 2

Consider join of R and S where R.a = S.b.

Info:

- R contains 10,000 tuples, each tuple is 400 bytes long.
- S contains 2,000 tuples, each tuple is 400 bytes long.
- Page size is 4096 bytes (96 unusable), unpacked, bitmap page format is used.
- Attribute b of S is the primary key for S.
- Both relations are stored as simple heap files, no indexes.
- Available memory is 52 pages.
- Fudge factor is 1.1.

Calculations:

```
Number of R pages
    = (number of tuples) * (tuple size) / (page size)
    = 10,000 * 400 / 4000
    = 1000 pages

Number of S pages
    = (number of tuples) * (tuple size) / (page size)
    = 2,000 * 400 / 4000
```

```
    = 200
```

Part A

What is the cheapest cost of joining R and S using block nested loops join for the given amount of memory space?

What should the number of memory pages be to minimize the cost?

Solution

R, S

```
B = 52
R = 1000
S = 200

Cost
    = R + ceil(R/(B - 2)) * S
    = 1000 + ceil(1000 / 50) * 200
    = 5000
```

S, R

```
B = 52
R = 1000
S = 200

Cost
    = S + ceil(S/(B - 2)) * R
    = 200 + ceil(200/50) * 1000
    = 4200
```

Final Answer

Since `4200 < 5000`, using S as the outer and R as the inner is the most efficient.

Like the notes say, it is optimal if `S = B - 2`. Therefore if we have 202 pages of buffer memory, we will optimize the cost of the join.

Part B

What is the cheapest cost of joining R and S using a GRACE hash join?

Solution

```
R = 1000
S = 200

Cost to partition
    = 2 * R + 2 * S
    = 2 * 1000 + 2 * 200
    = 2400

Cost to probe
    = R + S
    = 1000 + 200
    = 1200

Cost
    = (cost to partition) + (cost to probe)
    = 2400 + 1200
    = 3600
```

Part C

What is the cheapest cost of joining R and S using a sort-merge join?

Solution

```
R = 1000
S = 200

Cost to sort R
    = 2 * 2 * R
    = 2 * 2 * 1000
    = 4000

Cost to sort S (upper bound)
    = 2 * 2 * S
    = 2 * 2 * 200
    = 800

Cost to merge S and R
    = 1000 + 200

Total Cost
    = (cost to sort R) + (cost to sort S) + (cost to merge)
    = 4000 + 800 + (1000 + 200)
    = 6000
```