CS150A Homework 1 – Writing

School of Information Science and Technology
March 22, 2024

1 SQL (40 PTS)

The following relations keep track of airline flight information:

Flights (flno: `integer`, distance: `integer`, price: `integer`, aid: `integer`)
Aircraft (aid: `integer`, aname: `string`, cruisingrange: `integer`)
Certified (eid: `integer`, aid: `integer`)
Employees (eid: `integer`, ename: `string`, salary: `integer`)

Note that the underlined attributes are the primary keys; the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly; each aircraft may operate multiple flights. Answer the following questions.

1. Write a SQL to find the eid, name and salary of employees who are not pilots and sort them from highest to lowest salary.
2. Write a SQL to find the employee ID (i.e. eid), name (i.e. ename) and salary of all pilots whose salary is greater than the average salary of all pilots, and sort the query results from low to high salary after removing duplicates.
3. Which of the following queries returns the flight number (i.e. flno) of the flights with the highest price per kilometer (i.e. price / distance)? (The correct answer may be more than one)

   A. SELECT F.flno
   FROM Flights F
   WHERE (F.price/F.distance) = MAX(F.price/F.distance);

B. SELECT F.flno
FROM Flights F
WHERE (F.price/F.distance) >= ALL (SELECT F.price/F.distance FROM Flights F) ;

C. SELECT F.flno
FROM Flights F
WHERE (F.price/F.distance) = (SELECT MAX(F.price/F.distance) FROM Flights F);

D. SELECT F.flno, MAX(F.price/F.distance)
FROM Flights F
GROUP BY F.flno

E. None of the above

2 INDEX AND B+ TREES (30 PTS)

1.Consider the B+ tree index of order d = 2 shown below:

1. Show the tree that would result from inserting a data entry with key 3 into this tree.
2. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the right sibling is checked for possible redistribution.

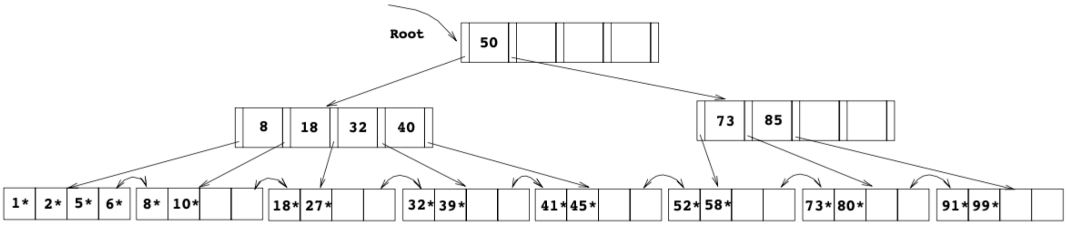

Figure 2.1: B+ Tree

2. Suppose that all nodes in our B+ tree have an order of 1500. What's the MAXIMUM number of records we can index with a B+ tree of height 2? (Assume our B+ trees are laid out as in lecture.)

3 FILE ORGANIZATION (30 PTS)

1. Consider the following relations:

Emp($\underline{eid:\ \text{integer},}$ $ename:$ varchar, $sal:$ integer, $age:$ integer, did: integer)

Dept($\underline{did:\ \text{integer},}$$budget:$ integer, $floor:$ integer, mgr_eid: integer)

Salaries range from $10,000 to$100,000, ages vary from 20 to 80, each department has about five employees on average, there are 10 floors, and budgets vary from \$10,000 to \$1 million. You can assume uniform distributions of values.

For each of the following queries, which of the listed index choices would you choose to speed up the query? If your database system does not consider index-only plans (i.e., data records are always retrieved even if enough information is available in the index entry), how would your answer change? Explain briefly.

1. Query: *Print ename, age, and sal for all employees.*
   - A. Clustered hash index on *⟨ename, age, sal⟩* fields of Emp.
   - B. Unclustered hash index on *⟨ename, age, sal⟩* fields of Emp.
   - C. Clustered B+ tree index on *⟨ename, age, sal⟩* fields of Emp.
   - D. Unclustered hash index on *⟨eid, did⟩* fields of Emp.
   - E. No index.
2. Query: *Find the dids of departments that are on the 10th floor and have a budget of less than \$15,000.*
   - A. Clustered hash index on the *floor* field of Dept.
   - B. Unclustered hash index on the *floor* field of Dept.
   - C. Clustered B+ tree index on *⟨floor,budget⟩* fields of Dept.
   - D. Clustered B+ tree index on the *budget* field of Dept.
   - E. No index.

2. Assume that each page in our system can hold 128 KB (1 KB = 1024bytes), integers are 32-bits wide, and bytes are 8-bits wide.

Consider the following relation:

```
CREATE TABLE Submissions(
    record_id integer UNIQUE,
    assignment_id integer,
    student_id integer,
    time_submitted integer,
    grade_received byte,
    PRIMARY KEY (assignment_id, student_id)
);
```

Assume the column record_id corresponds to the row’s actual record ID.

1. How large (in bytes) is a record?

2. Suppose we begin each page with a 32-bytes header plus a bitmap. At most, how many records can fit in an unpacked page?