Improving Query Performance

陆伟

College of Software

Database Systems

May 19, 2021

Northwestern Polytechnical University

Outline

- Introduction
- Database Storage and Query Performance
- Hash Files
- Indexes
- Query Optimization and Index Choice

Introduction

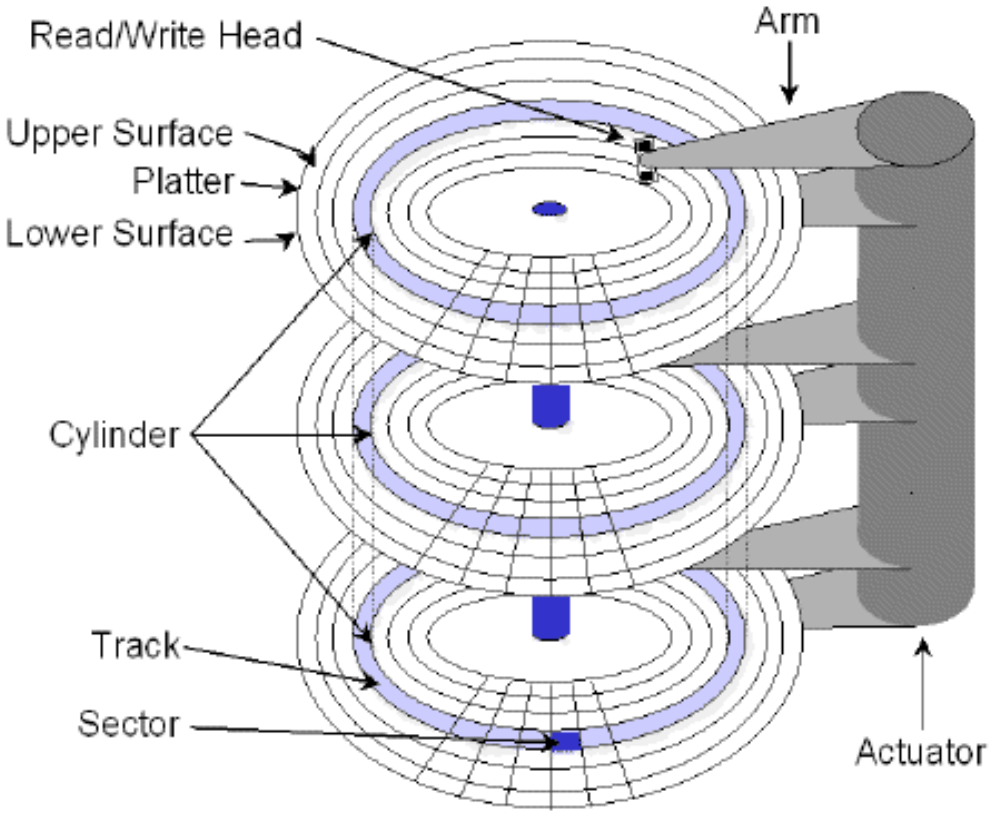
- A DBMS stores a database on a secondary storage, which typically is a disk. As we have learned in our discussion of database recovery, disks are extremely slow devices compared to main memory.
- So …

Introduction

- Attempting to improve the performance in a database system
  - Minimize the data flow between the main memory and the disk. (buffer)
  - Minimize the number of disk accesses required for locating a row. (access paths)
    - File organization
    - Access algorithms
- Indexes are examples of access paths and can greatly reduce the time it takes to process a query.

Database Storage and Query Performance

- Hard Disk Design
- Measuring Hard Disk Performance
  - Seek time: 8 to 14 milliseconds
  - Rotational latency
  - Block transfer time



Database Storage and Query Performance

- File or physical organization and Access Methods
  - The arrangements of the data into records (or tuples, rows, data items, objects) and pages (or blocks) on a disk is called file or physical organization.
  - Each file organization supports specific methods to store and retrieve records from the file.
  - Examples:
    - ordered files - binary search algorithm
    - heap files - linear search

Database Storage and Query Performance

- Improving Query Performance with Indexes
  - About blocking factor
  - Example

Consider a table consisting of 100,000 records and a blocking factor of 100. Sequentially scanning this entire table for a record can require as many as 1,000 disk blocks being read.

Assuming you have an index which is stored on a single disk block, on the other hand, may require as few as two disk blocks being read to find a record in this table (one disk block read to load the index and one disk block read to load the data).

Hash Files

- Hashing: Basic Organization
- About Collisions
- Properties of a Good Hash Function
  - It should be computed easily and efficiently.
  - It should minimize the number of collisions by spreading keys around the file as evenly and uniformly as possible.
- Pros and Cons of Hashing

Indexes

- An index is an auxiliary file that makes it more efficient to search for a record in a data file. It is usually defined on one field of the data file, called the indexing field.
- Each index entry is a pairing of an indexing value with a pointer to the page in which the value appears as part of a record.
- For fast search, the entries are sorted by the indexing field values.

Indexes

- According to the different properties of the indexing fields involved, indexes can be classified as
  - Primary Index
  - Secondary Index
  - Cluster Index
- Indexes are also classified as
  - Sparse
  - Dense

Indexes

- Indexed Sequential Access Method (ISAM)
  - An ISAM is a sparse, primary index.
  - It includes one index entry for each page in the data file.
  - The first record in a data page is called the page anchor.
  - The field value of an index entry is the key value of the page anchor.

Indexes

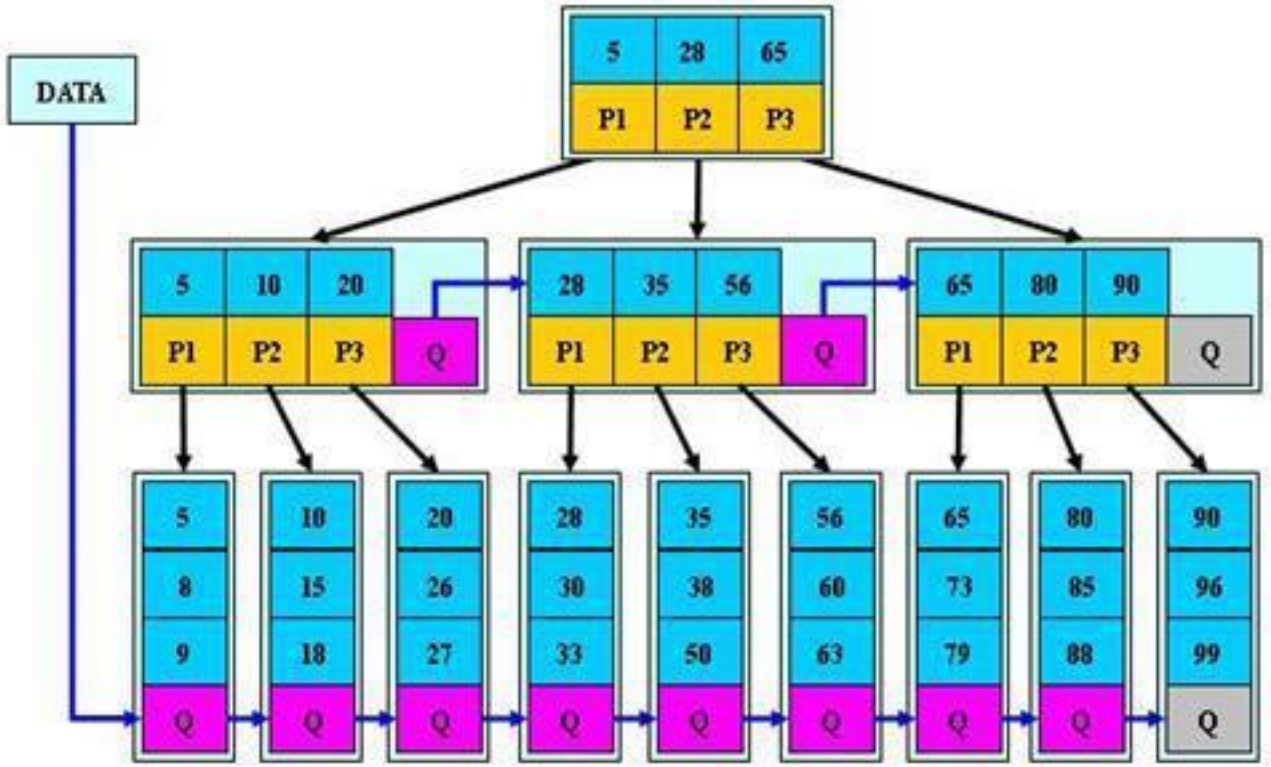
- Cluster Index
- Secondary ISAM
- Multi-Level ISAM

Indexes

- Drawbacks of ISAM
  - ISAMs are static structures and can offer great performance guarantees for searching, as long as insertions, deletions, and updates are very infrequent.
  - But in dynamic database system environments, …

Indexes

- B+ Trees



Indexes

- Indexes in SQL
  - Most existing DBMSs use fixed physical design and automatically build the best possible access path, which usually is some B+ tree.
- The general syntax of the CREATE INDEX statement

```
CREATE [UNIQUE] INDEX  index-name
      ON  table (indexing-attribute-order-list) [CLUSTER];
```

Query Optimization and Index Choice

- Query optimization is a process by which programmers and database administrators make use of access methods to ensure that an application accesses data as efficiently as possible.
  - One of the steps in optimizing queries is to examine the physical storage characteristics of tables accessed by those queries.
  - The next step in query optimization is analyzing the run-time characteristics of queries. (Find bottlenecks)

Query Optimization and Index Choice

- PostgreSQL Analysis Tools
  - The **pg_class** table stores information and statistics about each table

```
SELECT relname, reltuples, relpages FROM
pg_class WHERE relname = 'student';
```

```
relname   | reltuples | relpages
----------+-----------+----------
student   | 249999    | 4425
```

Query Optimization and Index Choice

- PostgreSQL Analysis Tools
  - It is important for the query planner to have accurate statistics in the **pg_class** table.
  - The analyze command updates the statistics in the **pg_class** table.
  - PostgreSQL does not update the data stored in the **pg_class** table for each update to a user table.
  - To allow the query planner to make the best possible plan, the database administrator should run the analyze command often.

Query Optimization and Index Choice

- PostgreSQL Analysis Tools
  - PostgreSQL can report a query's execution plan using **explain** command.

```
explain select distinct course_id from course where course_term = 'Fal02';
```

```
NOTICE:  QUERY PLAN:
Unique  (cost=12223.09..12339.76 rows=4667 width=4)
        -> Sort (cost=12223.09..12223.09 rows=46666 width=4)
        ->  Seq Scan on course  (cost=0.00..8279.99 rows=46666 width=4)
```

Query Optimization and Index Choice

- PostgreSQL Analysis Tools
  - It is important to note that the explain command does not actually execute a query, it only gets the plan for a query's execution and the estimated cost of running the query from PostgreSQL's query planner.

Query Optimization and Index Choice

- PostgreSQL Analysis Tools
  - The **explain analyze** command will execute a query and display the plan of execution for a query as well as the actual run-time in milliseconds and the actual number of disk page fetches for a query.

Query Optimization and Index Choice

- PostgreSQL Analysis Tools

```
explain analyze select distinct course_id
from course
where course_term = 'Fal02';
```

```
NOTICE:  QUERY PLAN:
Unique  (cost=12223.09..12339.76 rows=4667 width=4)
            (actual time=1643.87..1797.34 rows=41803 loops=1)
  ->  Sort  (cost=12223.09..12223.09 rows=46666 width=4)
            (actual time=1643.86..1706.05 rows=41803 loops=1)
  ->  Seq Scan on course  (cost=0.00..8279.99 rows=46666 width=4)
            (actual time=184.83..1075.11 rows=41803 loops=1)
Total runtime: 1899.24 msec
```

Query Optimization and Index Choice

- PostgreSQL Indexes
  - PostgreSQL supports both hash and b-tree indexes. PostgreSQL also supports clustering indexes.

create index *index_name* on *table* using *index_type(column[,column[,...]]);*

cluster *index_name* on *table;*

Query Optimization and Index Choice

- PostgreSQL Indexes

```
create index assessment_point_index on assessment
using btree(total_points);
```

```
cluster assessment_point_index on assessment;
```

```
create index assessment_id_index on assessment
using hash(assessment_id);
```

Query Optimization and Index Choice

- Application-Readings
  - Exercise

Query Optimization and Index Choice

- Index Choice and Optimization Heuristics

Summary

- In this chapter you should have learned:
  - Factors affecting query performance
  - Principles and types of index
  - Query Optimization and Index Choice

Quenstions