Cost of Disk Access

When are Indexes Good? Bad?

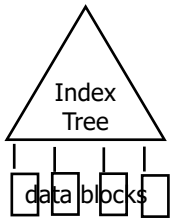
Objective: Understand the implications of disk access in conjunction indexes

- Power of a dense index as a copy of data values
- Clustering
- Overhead

- many slides, Thanks Garcia-Molina,

Indexes and Inserts

- All the data is in the data file, organized in blocks
  - sorted on primary key
  - (if possible) contiguously on disk



Simple I/O models

I/O models:

- Linear:
  - 1 average seek per block
    - n blocks, f(n) = cn, c = average seek time
- Affine:
  - seek first block,
  - weighted average of rotational latency + track to track seek time for each additional block
    - n block f(n) = c + c' (n-1)

In this course

- Always the linear cost model, unless specified otherwise

Simple I/O models:

How they fail

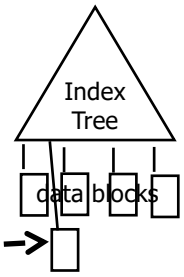
- buffering (hardware and software)
  - main memory
  - disk controller cache
- Prefetch – entire tracks into drive buffer
- blocks/pages aren't, necessarily, where *or* size as they are supposed to be (on board)
  - An indirection in case of bad pages/blocks

What about SSDs?

Data pages can overflow

If so

- fetch a new page from the heap
- new page is not contiguous

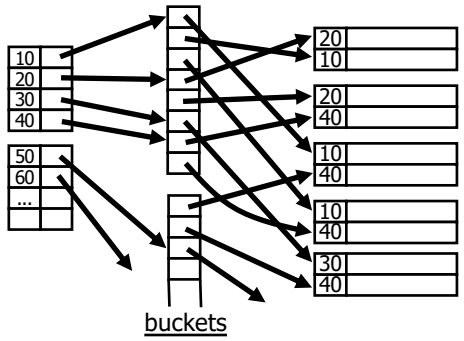


"Clustering"

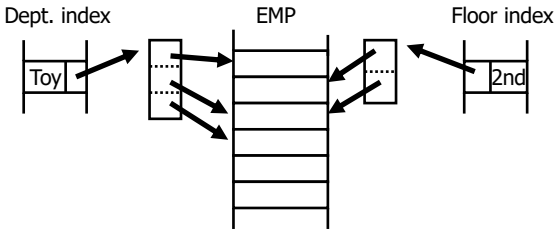
- Many uses/definitions of "clustering".
- How well sequentially ordered blocks appear sequentially on disk.
  - 100% clustered → 100% sequentially accessible
  - 90% clustered → 9 out of 10 accesses is sequential
- Typically only the primary index is clustered - physical reality of disks.

// SQLserver, "clustered index" = primary index

Secondary indexes are dense



Query: Get employees in (Toy Dept) ∧ (2nd floor)



→ Intersect toy bucket and 2nd Floor bucket to get set of matching EMP's

- Didn't have to touch the data file, except to find the output.

When do Secondary Indexes Make Sense?

Consider the example:

```
select *
from table
where city = 'austin'
```

`table` has one million rows

Number of cities =

- 100,000
- 100

Should we build/use a secondary index using a key called city?

Database Statistics are Stored in the Catalog/Dictionary

- 1 million records
- 10,000 data pages → avg 100 records per page
- fanout of a B+-tree = 100 using city as key

How Many Data Pages Do We Need to Read For the Query?

```
select *
from table
where city = 'austin'
```

Number of cities =
- 100,000

Repeat for 100 cities

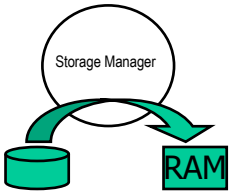
What about primary index?

- Primary index guaranteed to be a [unique] key
- Suppose eid forms primary key
- Consider

```
select *
from table
where eid = 'miranker'      // how many results?
                              → 1 data page access
```

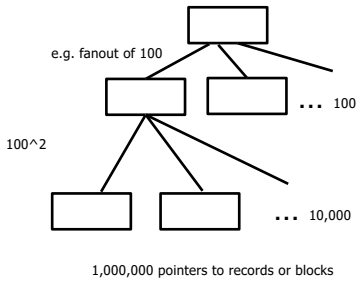

RDBMS Architecture

Storage Manager
- Exploit memory hierarchy to compensate for slow disks.
  - working sets (from OS)
  - *search algorithms*
- Specifics
  - manage a *heap* of disk pages
  - allocation of main memory (buffer management)
  - *LRU-like replacement algorithm*
  - index methods, e.g. B+ tree (access paths)

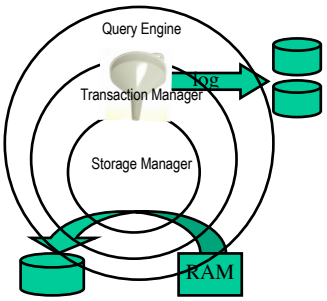


Locality of Reference at Top of the Tree

- *top of the index, usually pinned in memory*
- number of leaves dominate the total number of nodes



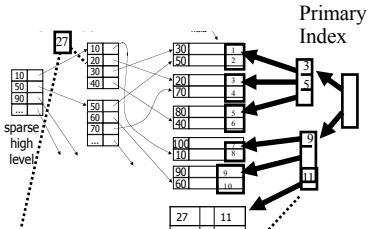
Transactions and Secondary Indexes



- recall, all writes/updates get written to a log
- atomicity - all or nothing
- *includes index records*

Transactions and Secondary Indexes

What Happens On Insert?



1. record is inserted
   insert data file
2. update primary index (sometimes)
3. update secondary index (always)

Properties of Secondary Indexes

- They don't usually cluster well
- They must be dense - which can have it's own benefit
- Increase cost of dynamic operations
  - insert, delete, update

  big impact on transaction performance
- ***Need to be careful about when to use them*** (text Ch. 8.4)
  - consider workload wrt:
    - transactions
    - database statistics (selectivity)
    - updates vs. reads

What About SSDs?

		9300 PRO (Read-Intensive, 1 Drive Write Per Day)			9300 MAX (Mixed-Use, 3 Drive Writes Per Day)		
Capacity[1]		3.84 TB	7.68 TB	15.36 TB	3.2 TB	6.4 TB	12.8 TB
Performance	Seq Read (MB/s)[2]	3500	3500	3500	3500	3500	3500
	Seq Write (MB/s)[2]	3100	3500	3500	3100	3500	3500
	Rand Read (K IOPS)[3]	835	850	850	835	850	850
	Rand Write (K IOPS)[3]	105	145	150	210	310	310
Endurance (Terabytes Written in PB)		8.4	16.8	33.6	18.6	37.3	74.7

- Sequential access speed >> random access

SSD Internals

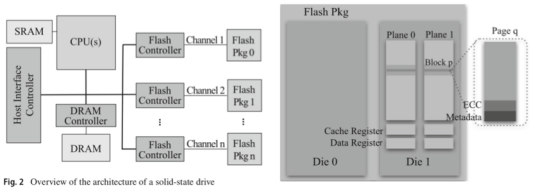
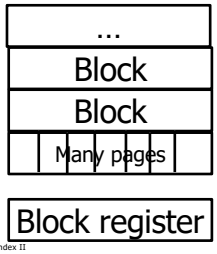


Fig. 2 Overview of the architecture of a solid-state drive

Fig. 1 The anatomy of a flash package

PLANE



Block read → many pages read in parallel

[like many word in RAM form a cache line]

But Also

1. Many planes can be read in parallel

→ planes x pages/block = parallel page reads

2. Block is primitive unit for writing.

3. Block level wear.
4. Block level wear leveling

→ map logical blocks to physical blocks

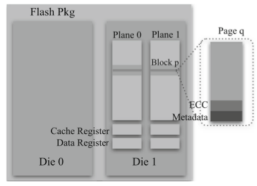


Fig. 1 The anatomy of a flash package

DBA Moment:
a word on transactions & real-world DBA heuristics

When developing database schema

1) Avoid secondary indexes on transactional data (e.g. point-of-sales)
   - why?

2) Separate transactional data from static data
   - e.g. (name,address) vs. balance

3) Minimize width of transactional rows
   - why?

A Practical (accepted) Dirty Trick

Account			
AID (pk)	name	address	balance
1234	dan	austin	1.00
6789	joe	houston	5.00

- start with good model
- then transform to denormalized model

•secondary indexes on "static" data,
•no impact on updates to dynamic data

AccountStatic		
AID	name	address
1234	dan	austin
6789	joe	houston

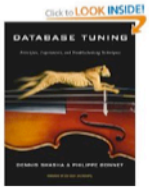
AccountDynamic	
AID	balance
1234	1.00
6789	5.00

A word on good design practice

- do not consider the above at logical design time.
  - I.e. get the logic right, then optimize
    - good for getting system done on time
      - (separation of concerns)

    &
    - long-term maintenance

If this is ever your job – money well spent:



Database Tuning: Principles, Experiments, and Troubleshooting Techniques (The Morgan Kaufmann Series in Data Management Systems) (Paperback)

~ Dennis Shasha (Author), Philippe Bonnet (Author) "Tuning rests on a foundation of informed common sense..." (more)
Key Phrases: nonclustering index, multipoint queries, summation query, Morgan Kaufmann, Tuning the Guts, International Conference (more...)
(4 customer reviews)

List Price: $75.95
Price: $52.56 & eligible for free shipping with **Amazon Prime**
You Save: $23.39 (31%)