scans vs indexes

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
**b-tree** - dynamic tree - always balanced

![Diagram of a b-tree structure](image-url)
select ... from R where A<v and ....

(secondary) **index vs scan**: the eternal battle
select … from R where A<v and …. 

(secondary) **index vs scan**: the eternal battle

Just having indexes in the system is or can be useless… or even bad for performance

Knowing when to use an index is key
Primary index vs secondary vs scan?

Just having indexes in the system is or can be useless... or even bad for performance.

Knowing when to use an index is key.
design/implement numerous possible algorithms + data representations

choose the best data source, algorithms and path for each query
scan
Table

Scan

Secondary Index Scan

Heap Page (P)

Tuple (T)

1  7
2  8
3  9
4 10
5 11
6 12

Root

Internal Nodes

Leaf Nodes
A secondary index on A values out of order with base data

A query that select on A and then needs B

intermediate out of order
secondary index on A
values out of order with base data

a query that select on A and then needs B
intermediate out of order
A secondary index on A values out of order with base data.

A query that selects on A and then needs B.

The intermediate data is out of order.
covering index:
contains all columns needed for a set of queries

no need to go to base data but…
random access to traverse the tree & need to sort result  

Vs.

sequential access pattern but needs to access all data
the standard solution
1) maintain statistics,
2) optimizer chooses access path depending on estimated selectivity
the standard solution
1) maintain statistics,
2) optimizer chooses access path depending on estimated selectivity

what is wrong with that
Motivation

TPCH (SF10) 2/2

Normalized execution time

Original

Tuned

Q1 Q3 Q5 Q7 Q9 Q11 Q13 Q16 Q19 Q22

Execution time (sec)

Result selectivity (%)
ROBUSTNESS
can we just recompute the statistics?

- **basic stats**
  - Full Scan
  - Index Scan
  - Optimizer decision
  - Avg. statistics collection

- **per column**
  - Execution time (sec)

- **for pair**
  - Execution time (sec)
if I keep 30 data systems researchers “trapped” in a castle for a week, we might be able to define “robust query processing” and find a few solutions
robust query processing (best definition to date by Goetz)
graceful degradation when the environment changes
Can we avoid bad access path selection (secondary index vs scan) when we have stale (or no) statistics?
select \text{min}(A) \text{ from } R \text{ where } B<10 \text{ and } C<80
SWITCH SCAN
while index probing
switch to scan
if cardinality > estimation

good: avoids worst case
bad: performance cliff

SMOOTH SCAN
goal avoid performance cliff
close to optimal

Design smooth scan
Read: **Access Path Selection in Main-Memory Optimized Data Systems:** Should I Scan or Should I Probe?
Mike. Kester, Manos Athanassoulis, and Stratos Idreos
ACM SIGMOD International Conference on Management of Data, 2017

Browse: **Smooth Scan: Statistics-Oblivious Access Paths**
Renata Borovica, Stratos Idreos, Anastasia Ailamaki, Marcin Zukowski and Campbell Fraser
IEEE International Conference on Data Engineering (ICDE), 2015

Extra: **Efficient mid-query re-optimization of sub-optimal query execution plans**
Navin Kabra and David DeWitt
ACM SIGMOD International Conference on Management of Data, 1998
scans vs indexes

DATA SYSTEMS

prof. Stratos Idreos