scans vs indexes

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
midterm feedback
you may get a copy of your midterm and walk through it with us during labs and OH
midway check-in
coming up quickly (last hours of Oct)

pass first 3 tests in testing infrastructure
(not your laptop)
write the first 3 design docs

the goal is to “push” everyone to do the bare minimum with the project to see the whole scope but in practice we expect you have finished M2 until end of October
clustered (all columns)

secondary indexes
subset of columns
b-tree - dynamic tree - always balanced
milestone 2

cache conscious b-tree-like index

node design, fill factor, etc

most differences with classic design you end up doing for the leaves

contiguous vs not, tuples vs single values, fill factor, need to maintain alignment info with the rest of the columns

insert & select now, updates later
declarative interface
ask what you want

db system

indexes/views/tuning knobs
select ... from R where A<v and ....

(secondary) **index vs scan**: the eternal battle
design/implement numerous possible algorithms + data representations

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

index
scan
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
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.

Intermediate out of order.
covering indices

A

→

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

sequential access pattern but needs to access all data
turning point

index

scan

response time

selectivity

% of values that qualify
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

Q1 Q3 Q5 Q7 Q9 Q11 Q13 Q16 Q19 Q22

Normalized execution time

Original Tuned

Index Scan Full Scan

Execution time (sec)

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

![Graph showing execution time vs. result selectivity for different strategies: Full Scan, Index Scan, Optimizer decision, and Avg. statistics collection.]
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 even when we have wrong statistics? how?
\textbf{select} \texttt{min(A)} \textbf{from} \ R \textbf{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
smooth scan
gradually morph from index scan to full scan

for each qualifying tuple\text{tID}
\textbf{mode1} fetch the respective page and get the row
\textbf{mode2} check all tuples in a fetched page
\textbf{mode3} fetch and check adjacent pages as well
\textbf{mode3+} increase # of pages fetched
some design points

tuple cache to avoid producing the same tuple twice
page cache to avoid reading the same page twice
result cache to produce result in indexed order
when to morph
in order to achieve a smooth behavior

optimizer start when selectivity > estimation
SLA respect an upper threshold
selectivity morph when selectivity increases by z
pessimistic morph with every new probe
The bar chart shows the execution time (in seconds) for different TPC-H queries using PostgreSQL and PostgreSQL with Smooth Scan.

- **Q1 (98%)**: PostgreSQL has a much lower execution time compared to PostgreSQL with Smooth Scan.
- **Q4 (<1%)**: Both show similar execution times, with PostgreSQL slightly lower.
- **Q6 (2%)**: PostgreSQL with Smooth Scan has a higher execution time compared to PostgreSQL.
- **Q14 (<1%)**: PostgreSQL with Smooth Scan has a higher execution time compared to PostgreSQL.

**Execution time** on the y-axis ranges from 0 to 1400 seconds. The queries are labeled with their respective percentages.
random access & page-based access

need to only read \( x \)...
but have to read all of page 1

data value \( x \)

data move

page1  page2  page3  ...

CPU
registers
on chip cache
on board cache
memory
disk
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

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

next: fast scans
scans vs indexes

DATA SYSTEMS

prof. Stratos Idreos