Performance

ERIN KEITH

Goals

1. Introduce Indexes
2. Explore Query Optimization

Performance

Many performance topics fall under the responsibility of database administrators

- Data defragmentation
- Increasing memory allocation
- Updating hardware
- Updating the DBMS version

Performance

Some fall in between the responsibility of database administrators and database developers

- Logging
- Normalization
- Improving indexes
- Tuning queries

Consider

A table of URLs which include
- Our sites
- Our backlinks
- Sites to contact

How big do you think that table would be?
What do you think common queries might look like?

Example

A table of URLs

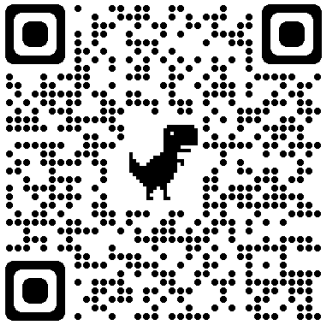
URL
https://www.bestcolleges.com/
https://www.bestcolleges.com/computer-science/bachelors/
https://www.bestcolleges.com/computer-science/bachelors/software-engineering/

Indexes

- Occur on table attributes
- Speed up queries
- Slow down database modifications
- Many DBMS provide indexing automatically

PostgreSQL

- https://www.postgresql.org/docs/current/indexes.html



- Read 11.1 & 11.2

Example

A table of URLs

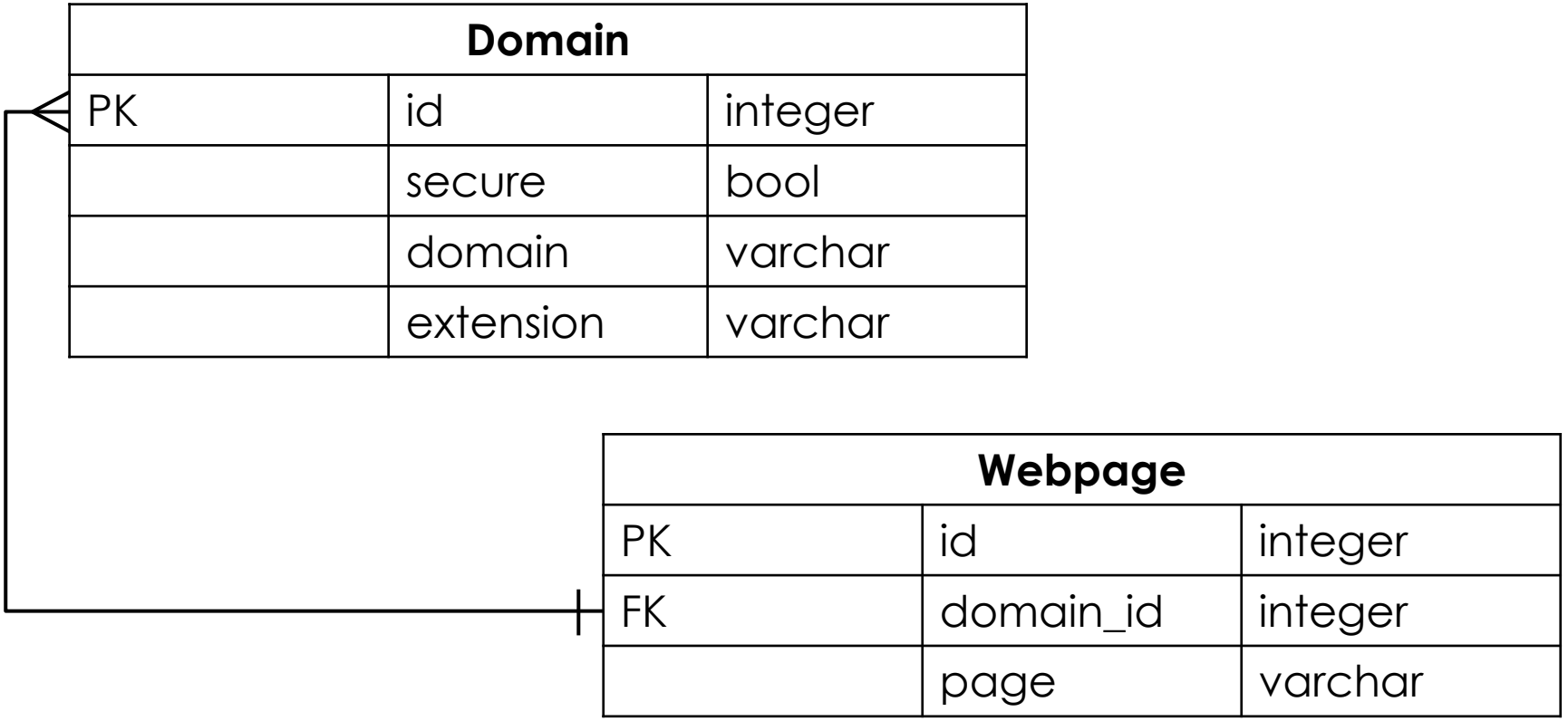
URL
https://www.bestcolleges.com/
https://www.bestcolleges.com/computer-science/bachelors/
https://www.bestcolleges.com/computer-science/bachelors/software-engineering/

Example

A table of URLs

Secure	Domain	Domain Extension	Page
true	bestcolleges	.com	home
true	bestcolleges	.com	computer-science/bachelors
true	bestcolleges	.com	computer-science/bachelors/software-engineering/

Example



Domain		
PK	id	integer
	secure	bool
	domain	varchar
	extension	varchar

Webpage		
PK	id	integer
FK	domain_id	integer
	page	varchar

Query Processor

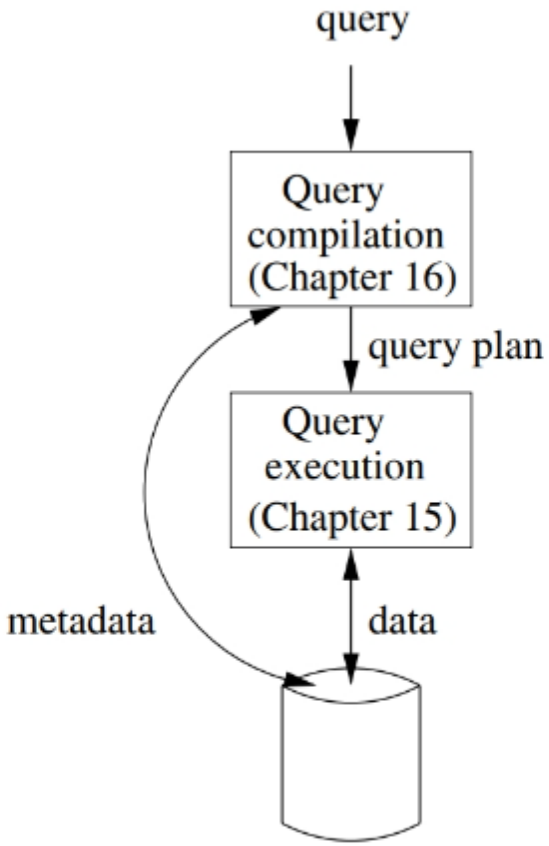


Figure 15.1: The major parts of the query processor

Query Compilation

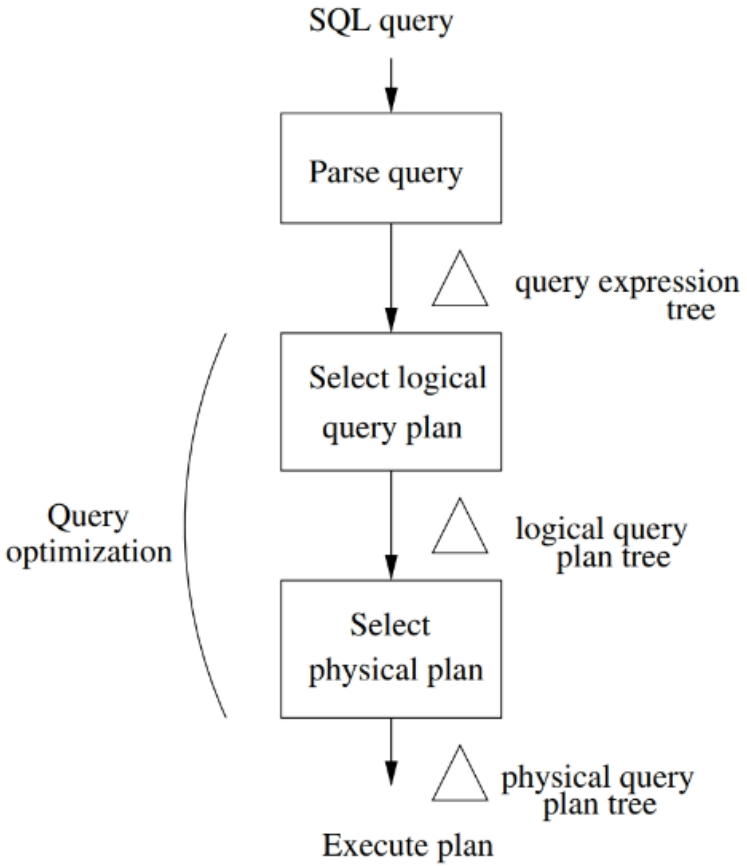


Figure 15.2: Outline of query compilation

Query Plan

- Which of the algebraically equivalent forms of a query leads to the most efficient algorithm for answering the query?
- For each operation of the selected form, what algorithm should we use to implement that operation?
- How should the operations pass data from one to the other, e.g., in a pipelined fashion, in main-memory buffers, or via the disk?

Query Plan

- Each of these choices depends on the metadata about the database:
  - the size of each relation
  - statistics such as the approximate number and frequency of different values for an attribute
  - the existence of certain indexes
  - and the layout of data on disk.

Algorithms

- One-pass
- Two-Pass
- Three(+)-Pass

Algorithms

- One-pass
  - tuple-at-a-time (unary)
    - selection, projection
  - full-relation (unary)
    - grouping, de-duping
    - limited by the number of main-memory buffers available
  - full-relation (binary)
    - union, intersection, difference, joins, and products
    - limited by the number of main-memory buffers available

Algorithms

- One-pass
  - tuple-at-a-time (unary)
    - selection, projection

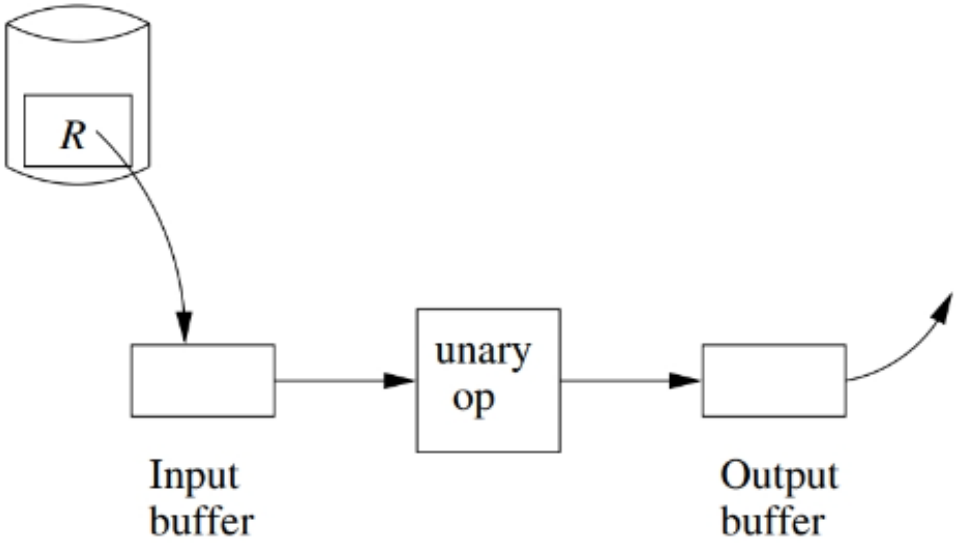
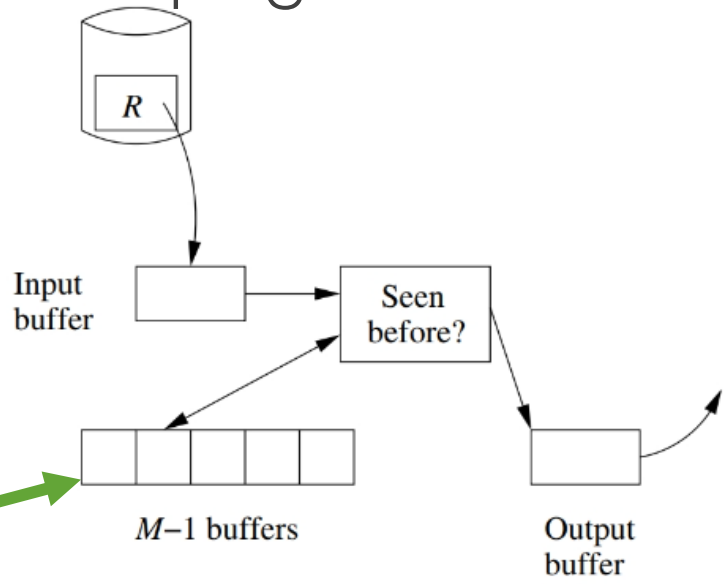


Figure 15.5: A selection or projection being performed on a relation R

Algorithms

- One-pass
  - full-relation (unary)
    - grouping, de-duping



what data structure?

Figure 15.6: Managing memory for a one-pass duplicate-elimination

Algorithms

- One-pass
  - full-relation (binary)
    - union, intersection, difference, joins, and products

Intersection?

Natural join?

Algorithms

Intersection S∩R

- Read S into M - 1 buffers
- Build a search structure with full tuples as the search key
- Read each block of R into the one remaining main-memory buffer
- For each tuple t of R, see if t is also in S
  - If so, copy t to the output
  - If not, ignore t

Algorithms

Natural join $S(Y; Z) \bowtie R(X; Y)$

- Y represents all the attributes that R and S have in common
- X is all attributes of R that are not in the schema of S
- Z is all attributes of S that are not in the schema of R

Algorithms

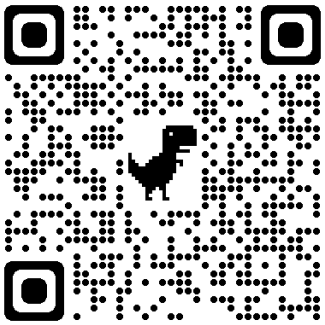
Natural join $S(Y; Z) \bowtie R(X; Y)$

- Read S into M - 1 buffers
- Build a search structure with the attributes of Y as the search key
- Read each block of R into the one remaining main-memory buffer
- For each tuple t of R, find the tuples of S that agree with t on all attributes of Y
  - For each matching tuple of S, form a tuple by joining it with t
  - Move the resulting tuple to the output

Query Tuning

Are our queries inefficient?

- Execution Plans
  - https://www.postgresql.org/docs/8.3/using-explain.html



Next Class

Module:

Week 15: Advanced Topics

Topic:

Grad Student Presentations