Storage & Indexing in Modern Databases

ECS 165A – Winter 2021

Mohammad Sadoghi
Exploratory Systems Lab
Department of Computer Science

UC Davis
University of California
How to quickly search for the desired information?
Searching for 76
Searching for 76
Searching for 76
Searching for 76
Searching for 76
Searching for 44?
(what-if the value does not exist)
(could we have early termination?)
Could we impose order to improve the search?
Searching for 76
Searching for 76
Searching for 44?
(could we have early termination?)
Could we impose a structure to further improve the search?
Searching for 76
Searching for 76
Searching for 76
Searching for 44?
(could we have early termination?)
Searching for 76-91
Could we spread the data cleverly to improve the search?
Hashing (●) = ?

(returns a value between 1 to n, where n is the number of buckets)
Inserting 81

Hashing (81) = 6

81
Hashing (43) = 10

Inserting 43
Inserting 76

Hashing (76) = 8
Hashing (91) = 10

Inserting 91

collisions
(when multiple values
hash to the same bucket)
collisions
(when multiple values hash to the same bucket)
Searching for 76
(now we can have constant lookup cost)

Hashing (76) = 8
Searching for 76-91?

Could we instead search for 76, 77, 78, ..., 90, 91?
Hashing (76) = 8
Hashing (77) = 1
Hashing (78) = 3
Hashing (81) = 6
Hashing (84) = 7
Hashing (90) = 8
Hashing (91) = 10

Searching for 76-91
Could we instead search for 76, 77, 78, ..., 90, 91?
Searching for 76-91

How about 76.01, 76.02, 76.03, ...?
(simply not practical)
Could we rethink the design to search for a range of values efficiently?
Let’s promote a subset of values as seeds

34 71 91
Let’s promote a subset of values as seeds

Suppose every value points to its next larger value

R-Hash- In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Searching for 76-91

R-Hash: In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Searching for 76-91

Find the largest seed smaller than 76: 71

R-Hash: In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Searching for 76-91

Hashing (71) = 3

Find the largest seed smaller than 76: 71

then simply follow the pointers to find all values between 76-91

R-Hash: In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Inserting 79

Hashing (79) = 10

R-Hash: In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Inserting 79

Hashing (71) = 3

Hashing (79) = 10

Find the largest seed smaller than 79: 71

sorted seeds
34 71 91

R-Hash: In-memory latch-free index structure by Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Inserting 79

Hashing (71) = 3

Find the largest seed smaller than 79: 71

adjust the pointers accordingly

Hashing (79) = 10

R-Hash· In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Database Storage Layouts

(how often do we need an index for range queries?)
a database record, e.g.,
[Name: Alice, Age:21, Major: CS]

Row-based Layout
database pages (containing a set of records)

a database record, e.g., [Name: Alice, Age:21, Major: CS]

Row-based Layout

Column-based Layout
### Database Pages
(containing a set of records)

A database record, e.g.,

[Name: Alice, Age:21, Major: CS]

- **Row-based Layout**
  - [Name: Alice, Age:21, Major: CS]
  - [Name: Bob, Age:21, Major: CS]
  - [Name: Joe, Age:23, Major: EE]
  - [Name: Alex, Age:24, Major: EE]
  - [Name: Sally, Age:25, Major: EE]

- **Column-based Layout**
  - [Name]
    - [Alice]
    - [Joe]
    - [Alex]
    - [Sally]
  - [Age]
    - [21]
    - [21]
    - [24]
    - [25]
  - [Major]
    - [CS]
    - [CS]
    - [CS]
    - [LS]
<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>21</td>
<td>CS</td>
</tr>
<tr>
<td>Joe</td>
<td>23</td>
<td>EE</td>
</tr>
<tr>
<td>Bob</td>
<td>21</td>
<td>CS</td>
</tr>
<tr>
<td>Alex</td>
<td>24</td>
<td>CS</td>
</tr>
<tr>
<td>Sally</td>
<td>25</td>
<td>EE</td>
</tr>
</tbody>
</table>

Searching for all students between the age of 21 to 24 (returns many students)
Searching for all students between the age of 21 to 24 (returns many students)

Index on Age

Row-based Layout

[Name: Alice, Age:21, Major: CS]
[Name: Bob, Age:21, Major: CS]
[Name: Joe, Age:23, Major: EE]
[Name: Alex, Age:24, Major: EE]
[Name: Sally, Age:25, Major: EE]

Column-based Layout

[Name] [Age] [Major]
[Alice] [21] [CS]
[Bob] [21] [CS]
[Joe] [23] [EE]
[Alex] [24] [CS]
[Sally] [25] [EE]
<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Major</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>21</td>
<td>CS</td>
</tr>
<tr>
<td>Bob</td>
<td>21</td>
<td>CS</td>
</tr>
<tr>
<td>Joe</td>
<td>23</td>
<td>EE</td>
</tr>
<tr>
<td>Alex</td>
<td>24</td>
<td>EE</td>
</tr>
<tr>
<td>Sally</td>
<td>25</td>
<td>EE</td>
</tr>
</tbody>
</table>

**Row-based Layout**

**Column-based Layout**

**Index on Age**

Searching for all students between the age of 21 to 24 (returns many students)
Searching for all students between the age of 21 to 24 (returns many students)

Row-based Layout

Index on Age

[21, 23, 24]

Column-based Layout

Alternatively read only the Age column to find the relevant values
Searching for all students between the age of 21 to 24 (returns many students)

Row-based Layout

Column-based Layout

Is the index really useful here?
Searching for all students over the age of 24 (returns only few students)

Row-based Layout

- [Name: Alice, Age:21, Major: CS]
- [Name: Bob, Age:21, Major: CS]
- [Name: Joe, Age:23, Major: EE]
- [Name: Sally, Age:25, Major: EE]
- [Name: Alex, Age:24, Major: EE]

Column-based Layout

- [Name]  [Age]  [Major]
  - [Alice]  [21]  [CS]
  - [Bob]  [21]  [CS]
  - [Sally]  [25]  [EE]
  - [Alex]  [24]  [CS]
  - [Joe]  [23]  [EE]
Searching for all students over the age of 24 (returns only few students)

Could we instead employ hashing with the seeding idea?

Index on Age

Row-based Layout

Column-based Layout

[Name: Alice, Age:21, Major: CS]

[Name: Bob, Age:21, Major: CS]

[Name: Sally, Age:25, Major: EE]

[Name: Alex, Age:24, Major: EE]

[Name: Joe, Age:23, Major: EE]
Thank You

Questions?