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 76
Searching for 76
Searching for 76
Searching for 44?
(what-if the value does not exist)
(could we have an early termination?)
Could we impose an order to improve the search?
Searching for 76
Searching for 76
Searching for 44?
(could we have an 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 an 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
Inserting 43

Hashing (43) = 10
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 a constant lookup cost)

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

Could we instead search for 76, 77, 78, ..., 90, 91?
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 imagine a new design to support searching 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

sorted seeds

34 71 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 by 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
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

Find the largest seed smaller than 79: 71

Hashing (79) = 10
Hashing (71) = 3

Find the largest seed smaller than 79: 71

Hashing (79) = 10

adjust the pointers accordingly

Inserting 79

sorted seeds
34 71 91

R-Hash: In-memory latch-free index structure B Bhattacharjee, M Canim, M. Sadoghi, US Patent 9,858,303
Database Storage Layouts
(how likely that we need an index for range queries?)
database pages
(containing a set of records)

[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]

Column-based Layout

Row-based Layout
### Database Pages

Database pages (containing a set of records)

### Database Record

A database record, e.g.,

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

### Column-based Layout

<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>Sally</td>
<td>25</td>
<td>EE</td>
</tr>
<tr>
<td>Alex</td>
<td>24</td>
<td>CS</td>
</tr>
</tbody>
</table>

### Row-based Layout

<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>Sally</td>
<td>25</td>
<td>EE</td>
</tr>
<tr>
<td>Alex</td>
<td>24</td>
<td>CS</td>
</tr>
</tbody>
</table>

---

**Column-based Layout**

- [Name: Alice, Age:21, Major: CS]
- [Name: Joe, Age:23, Major: EE]
- [Name: Alex, Age:24, Major: EE]
- [Name: Sally, Age:25, Major: EE]
Searching for all students between the age of 21 to 24 (may return many students)

<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>Sally</td>
<td>25</td>
<td>EE</td>
</tr>
<tr>
<td>Alex</td>
<td>24</td>
<td>CS</td>
</tr>
</tbody>
</table>

**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**

<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>Sally</td>
<td>25</td>
<td>EE</td>
</tr>
<tr>
<td>Alex</td>
<td>24</td>
<td>CS</td>
</tr>
</tbody>
</table>
Searching for all students between the age of 21 to 24 (may return 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]
Searching for all students between the age of 21 to 24 (may return many students)

Row-based Layout

- Alice, Age: 21, Major: CS
- Bob, Age: 21, Major: CS
- Sally, Age: 25, Major: EE
- Alex, Age: 24, Major: EE

Column-based Layout

- Alice
  - Name: Alice
  - Age: 21
  - Major: CS
- Joe
  - Name: Joe
  - Age: 23
  - Major: EE
- Sally
  - Name: Sally
  - Age: 25
  - Major: EE
- Alex
  - Name: Alex
  - Age: 24
  - Major: EE

Index on Age

[21, 23, 24]
Searching for all students between the age of 21 to 24 (may return many students)

Row-based Layout

Index on Age

Column-based Layout

Alternatively read only the Age column to find the relevant values

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

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

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

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

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

[Name: Bob]

[21]

[CS]

[Name: Sally]

[25]

[EE]

[Name: Alex]

[24]

[EE]

[Name: Joe]

[23]

[EE]
Searching for all students between the age of 21 to 24 (may return many students)

Index on Age

Row-based Layout

Column-based Layout

Is an index really useful here?
Searching for all students over the age of 24 (may return only a few 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]
[Alice]
[Joe]
[Alex]
[Sally]

[Age]
[21]
[23]
[24]
[25]

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

Searching for all students over the age of 24 (may return only a few students).

Could we instead employ hashing with the seeding idea?

Index on Age

Row-based Layout

Column-based Layout
Thank You
Questions?