Indexes Chapter 14

An index is a data structure that given one or more attributes finds those tuples that match those attributes.

The attributes of the index are known as search keys of the index.

Why do we need indexes?

In the absence of indexes (or indices, both are ok) any query that has a selection must test every tuple

⇒ could be very expensive if relation is large and few tuples match selection.

A primary key creates an index on the primary key attributes.

- Improves performance during insertion, update of key.

⇒ otherwise it is necessary to read entire table at every insertion.

A relation with attributes that are foreign keys has an automatic index created on the FK.

- Improves performance when looking up specific values of the FK.

  ⇒ otherwise it is necessary to read entire table at every insertion.

To create an index:

```
CREATE INDEX <idxname> ON
        <relname>(attlist);
```

where attlist is a list of attributes (order matters... more on that later).

Data Storage

- Relations have their tuples stored in heaps (data files).
- The minimal amount of data that can be read from disk is a block.
- We will assume heaps have no wasted space (in practice not true).

- The size of the heap of a relation $B(R)$ depends on::
  - Number of tuples per block: $tb(R)$
  - Number of tuples in relation: $|R|$

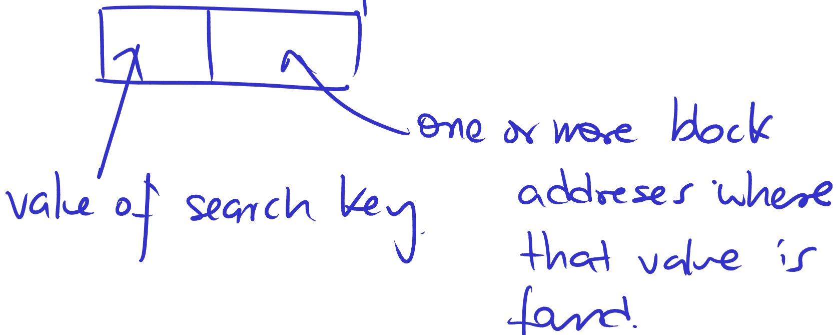
$$B(R) = \left\lceil \frac{|R|}{tb(R)} \right\rceil$$

- In the absence of indexes any query that uses R will read $B(R)$ blocks.
  - We ignore caching for the sake of simplicity.

$\Rightarrow$ $B(R) \propto |R|$ and
$B(R) \propto$ size of each tuple.

Indexes

- Contain a set of index entries.



(we will assume one only).

Dense index:

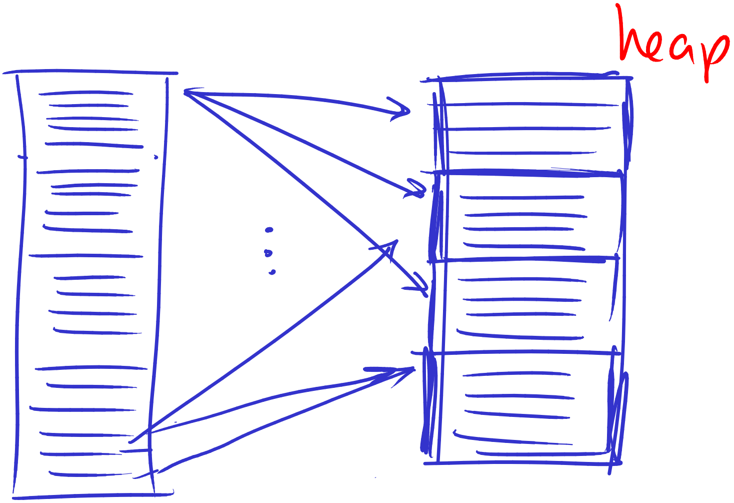
- The index contains one index entry per tuple in the relation.

Sparse index:

- The index contains one index entry per block in the relation.

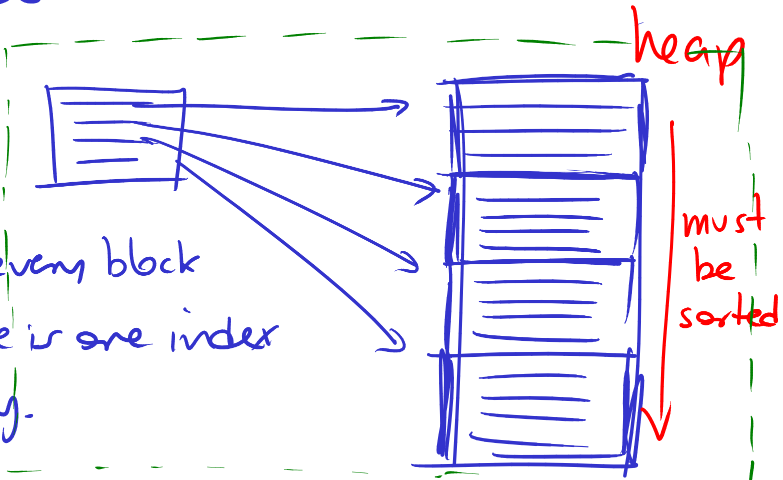
⇒ only useful if heap is ordered according to search key of index

Dense



for every tuple there is one index entry.

Sparse



for every block there is one index entry.

Sparse index and heap might be stored together

Clustered index.

Number of used entries in index. I of R

Dense:

$|R|$

Sparse: $B(R)$

Total number is larger because of waste (more later).

Sparse indexes determine the order of tuples in the heap.

⇒ called primary indexes

In general there can only be one primary index per relation.

A secondary index does not determine the order of the heap.

A relation can have zero or more secondary indexes.

⇒ Secondary indexes are never sparse.

When do we use indexes?

Assume

$R(\underline{a}, b)$ $S(\underline{a}, c)$

$\leftarrow$ a references R(a)

What indexes might be useful for?

$(\sigma_{b=5} R) \bowtie S$

Index on R(b) and

Index on either S(a)

Why is R(a) not useful?

$R \bowtie S$

Index on R(a) or S(a)

but not both!!

$\sigma_{a=3 \text{ and } b=3} R$

Index on R(a) or R(b) but not both!!

$\sigma_{a=3 \text{ or } b=3} R$

Both index on R(a) and R(b)

$\sigma_{b=5}(R \times S)$

If the cross product is done first then index is no longer useful

$\Rightarrow$ $R \times S$ is a temporary relation.

$\sigma_{a>5} R$

$\Rightarrow$ Depends on the type of index.

$\sigma_{a \text{ is NULL}} R$

$R(a)$

In general, every query requires to access tuples from the heap.

- either directly: called sequential scan
- or using one index.

DBMS will choose cheapest (in terms of read blocks).