Indexing

December 12, 2008

Introduction

- New tuple is stored without any order
  - next available space

Introduction

- New tuple is stored without any order
  - next available space
- Access will require inspection of every tuple

```
SELECT name, salary
FROM department
WHERE name LIKE 'A%';
```

  - requires visiting each row for comparison since no alphabetical storage

Introduction

- Indexing: speeds up access to desired data
  - e.g., a book index, catalog index in library

Introduction

- Indexing: speeds up access to desired data
  - e.g., a book index, catalog index in library
- DB Indexes: used by DB server to locate tuples in a table
- Indexes: special tables with *ordered* tuples
  - one (or more) columns from main table
  - includes pointer to the full row in main file

Basic Index Operations in MySQL

- Create on the column most often used in queries, update, delete
- In MySQL:

```
ALTER TABLE employee
ADD INDEX dept_indx (dept_id);
```

- Other DB:

```
CREATE INDEX dept_indx
ON employee (dept_id);
```

- View indexes on a table:

```
SHOW INDEX from employee;
```

- Drop an index:

```
ALTER TABLE employee drop INDEX dept_indx;
```

Multi-column Indexes

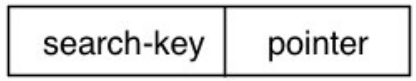
- Create index on two columns:

```
ALTER TABLE person
ADD INDEX name_indx (lname, fname);
```

- Can use for queries with:
  - both `lname` and `fname`
  - `lname` alone
  - **not** for `fname` alone

Basic Concepts for Index Implementation

- **SearchKey** - attribute to set of attributes used to look up records in a file.



- An **index file** consists of records (called index entries) of the form
- Index files are typically much smaller than the original file
- Two basic kinds of indices:
  - Ordered indices: search keys are stored in sorted order
  - Hash indices: search keys are distributed using a "hash function"

Dense Index Files

- **Dense Index**: search-key appears for every search-key value from main file

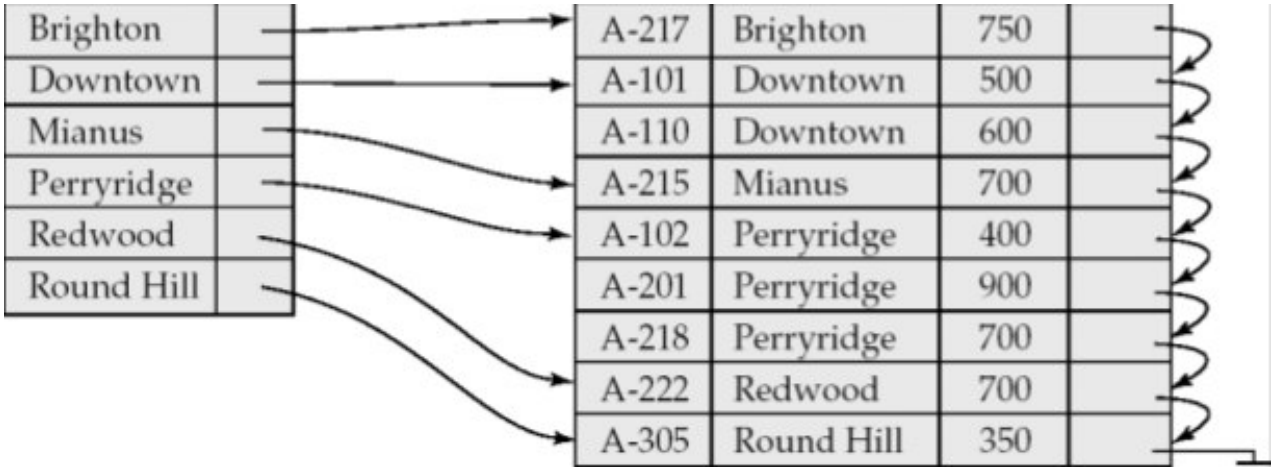


Figure: A dense index file

Sparse Index Files

- **Sparse Index**: search-key appears for only a few values

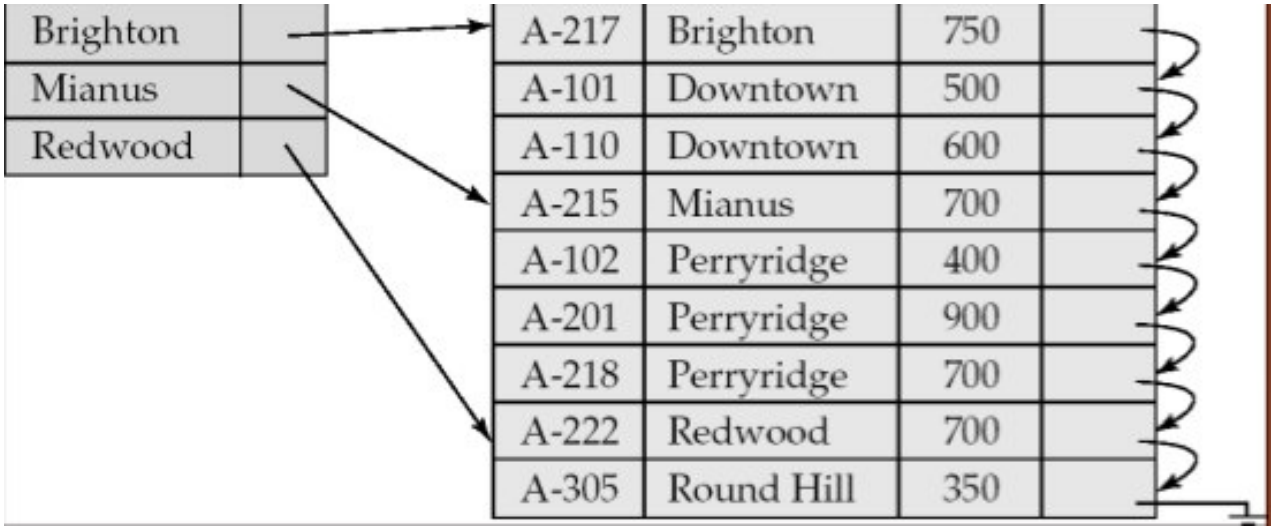


Figure: A sparse index file

B-Tree Indexes

- Balanced trees:
  - length from root to any leaf is same
  - every non-leaf node has n to $n/2$ nodes; n is fixed
- Most common, default index type

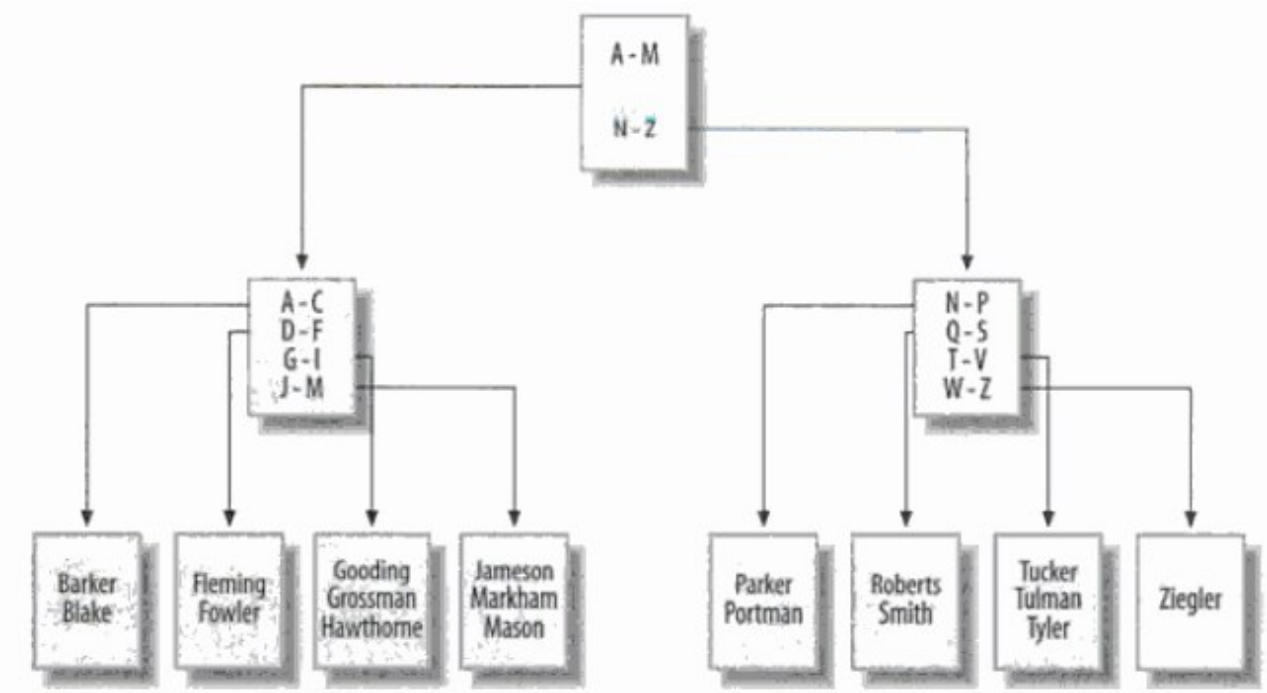


Figure: Example of a B-tree Index

B-tree Indexes

- Sequential search (without B-tree index): $O(n)$
- B-tree search: $O(log(n))$

B-tree Indexes

- Sequential search (without B-tree index): $O(n)$
- B-tree search: $O(log(n))$
- Adds:
  - performance overhead (insertion, deletion)
  - adds space overhead