Indexing

Nov 6, 2007

Chapter 8

Hardware Basics

- Disk access time: 10 msecs
- Memory access time: 60 nanoseconds
- We can run many instructions in 10 msecs!

Indexing

```
mysql> CREATE TABLE Actors
            (ActorID INT NOT NULL AUTO_INCREMENT,
            LastName VARBINARY(50),
            FirstName  VARBINARY(50) NOT NULL,
            Gender ENUM('Male', 'Female') NOT NULL,
            PRIMARY KEY(ActorID),
            INDEX(Gender)
            ) ENGINE=InnoDB;
```

- Common access methods
  - Scan
  - Equality
  - Range

http://www.innodb.com/innodb/info/

Database Files

- Data File – data from one table
  - Collection of file pages
    - Each page contains a number of data records
    - InnoDB: 16KB page size
    - One disk access to retrieve each page
  - Data records
    - 1 record = 1 row in a table
    - Each data record has a record id (rid) <pageid, slotid>
    - Can be used to retrieve the record
  - Unordered file: heap file
- Index File
  - Auxiliary file that matches database indexes to rids
  - Data entries

Assume each index is tied to exactly 1 column in the table

Index Files

- Three types:

  1 The data entry is the database row

    - No auxiliary file
    - Called an indexed file

  2 The data entry is a <db index, rid> pair

  3 The data entry is a <db index, rid-list> pair

- For any table, you can have one indexed file and many of 2 or 3

- Primary & Secondary indexes

Clustered Indexes

- Data records stored in near sorted order
  - Records in a page are nearly ordered
  - Clustered indexes
  - Otherwise, called unclustered indexes
- Generally, only option 1 is clustered
  - Called a clustered file
  - Expensive to keep a file sorted
  - often gaps are kept in the file to allow easy (sorted) insertion
- Why would this be useful?
- Only cluster one index per table. Why?

Index Data Structures

- Hash table
  - Chapter 11
  - hash(ActorID) = PageID

- Trees
  - Chapter 10
  - B+ Trees

Hashing

- What is the O() for the access time of a hash table?
- Example: Page 280, Figure 8.2
- Can you cluster a Hash index?

Trees

- Let's review Binary Search Trees
  - fan-out?
  - O() for finding a value in a BST?
  - Why?
  - What problems do BSTs have?

B+ Tree

- B+ Tree
  - rebalancing tree!
    - all paths from the root to any leaf are the same length
  - may have a higher fan-out than 2
  - B+ tree of order b has between (b/2)+1 and b keys per node
    - except the root, between 2 and b keys
  - all data stored at the leaf nodes
    - (B trees can store data in any node)
- Example: page 281, Figure 8.3

B+ vs BST

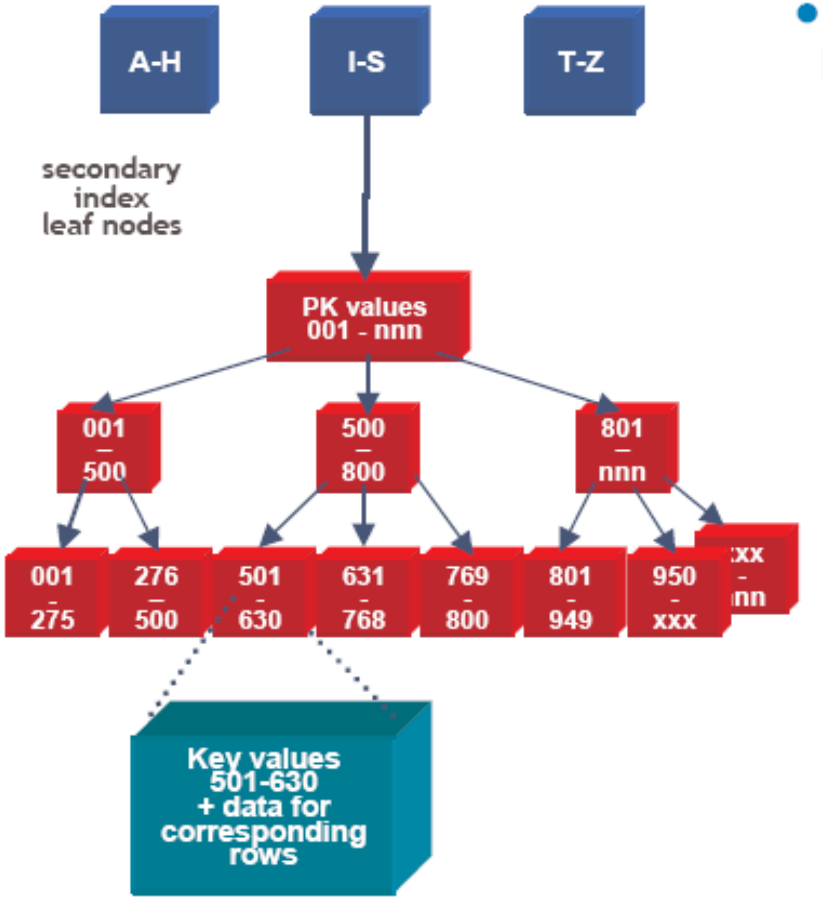
- If we have 100,000,000 records
  - how long would it take to find a record with a BST?
  - with a B+ Tree with fan-out 100?
    - 100 is a typical fan-out for a B+ Tree in an index
  - Each step in the tree may be a disk read

Composite Keys

```
mysql> CREATE TABLE WasIn
           (ActorID INT NOT NULL,
           MovieID INT NOT NULL,
           PRIMARY KEY(ActorID, MovieID),
           INDEX(MovieID)
           ) ENGINE=InnoDB;
```

- An index is tied to 2+ columns in the table
- Let's assume a hash for simplicity....
  - what happens with `WHERE ActorID=4 AND MovieID=3`
  - what happens with `WHERE MovieID=3`
  - what happens with `WHERE ActorID=4 AND MovieID < 9`

InnoDB Data & Index Storage



- Data rows are stored in the B-tree leaf nodes of a clustered index (aka "index-organized table")
  - B-tree is organized by primary key or non-null unique key of table, if defined; else, an internal column 6-byte ROW_ID is added and used
- Secondary index B-tree leaf nodes contain, for each key value, the primary keys of the corresponding rows, used to access clustering index to obtain the data
  - Note: long primary keys use a lot of space in secondary indexes! Use short or surrogate key …

INNOBASE

http://www.innodb.com/wp/wp-content/uploads/2007/04/innodb-overview-mysql-uc-2006-pdf.pdf