Indexing & Storage Engines

April 10, 2013

Chapter 8

Join

E-R Diagram?

Professors

ProfID	FName	LName	StatusID
1	D	R	3
2	S	K	2
3	C	W	1

JobStatus

StatusID	Name	PayBonus	Tenure
1	Professor	10000	Yes
2	Associate	1000	Yes
3	Assistant	0	No

```
SELECT *
FROM    Professors
WHERE StatusID=3;

SELECT *
FROM JobStatus
WHERE PayBonus > 100;
```

```
SELECT ProfID, LName, Name, Tenure
FROM    Professors, JobStatus
WHERE Professors.StatusID=JobStatus.StatusID;
```

What happens? Primary Key? Index?

Join

FixInducing

BugID	FileID	TransID
1	1	100
2	1	100
3	2	150

E-R Diagram?

SourceCodeRevisions

FileID	TransID	FileText	Author
1	100	#include ...	Chadd
1	150	#include ...	Doug
2	150	/******** ...	Chadd

Files

FileID	FileName	Directory
1	main.c	src/driver
2	other.c	src/util
3	simple.c	src/datas...

```
SELECT *
FROM    FixInducing as FI, SourceCodeRevisions as S
WHERE FI.TransID=S.TransID
```

```
SELECT BugID, FI.FileID, FI.TransID, Author, FileName
FROM    FixInducing as FI, SourceCodeRevisions as S, Files as F
WHERE FI.FileID=S.FileID and
          FI.TransID=S.TransID and
          F.FileID=S.FileID
```

What happens? Primary Key? Index?

Hardware Basics

- Disk access time: 10 msecs
- Memory access time: 60 nanoseconds
  - faster than disk access by ???
- We can run many instructions in 10 msecs!
- What does it cost to find a row?

Storage Engine

- How is the data stored?
  - file format
  - indexes
  - transactions/concurrency
- MySQL ships with a number of storage engines
  - MyISAM
  - InnoDB
  - plug-ins can add support for others

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

InnoDB Transactions

- Atomic - all changes are either committed as a group, or all are rolled back as a group
- Consistent - transactions operate on a consistent view of the data, leaving the data in a consistent state (by transaction's end)
- Isolated - each transaction "thinks" it is running by itself - effects of other transactions are invisible until it commits
- Durable - once committed, all changes persist, even if there are system failures

http://www.innodb.com/wp/wp-content/uploads/2008/04/intro-to-innodb-at-the-2008-mysql-uc-final.pdf

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/products/innodb/info/
Intro to InnoDB at the 2008 MySQL User Conference

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

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

- Index File
  - Auxiliary file that matches database indexes to rids
  - data entry

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

- Generally, only option 1 is clustered
  - Expensive to keep a file sorted
  - often gaps are kept in the file to allow easy (sorted) insertion

- Why would this be useful?

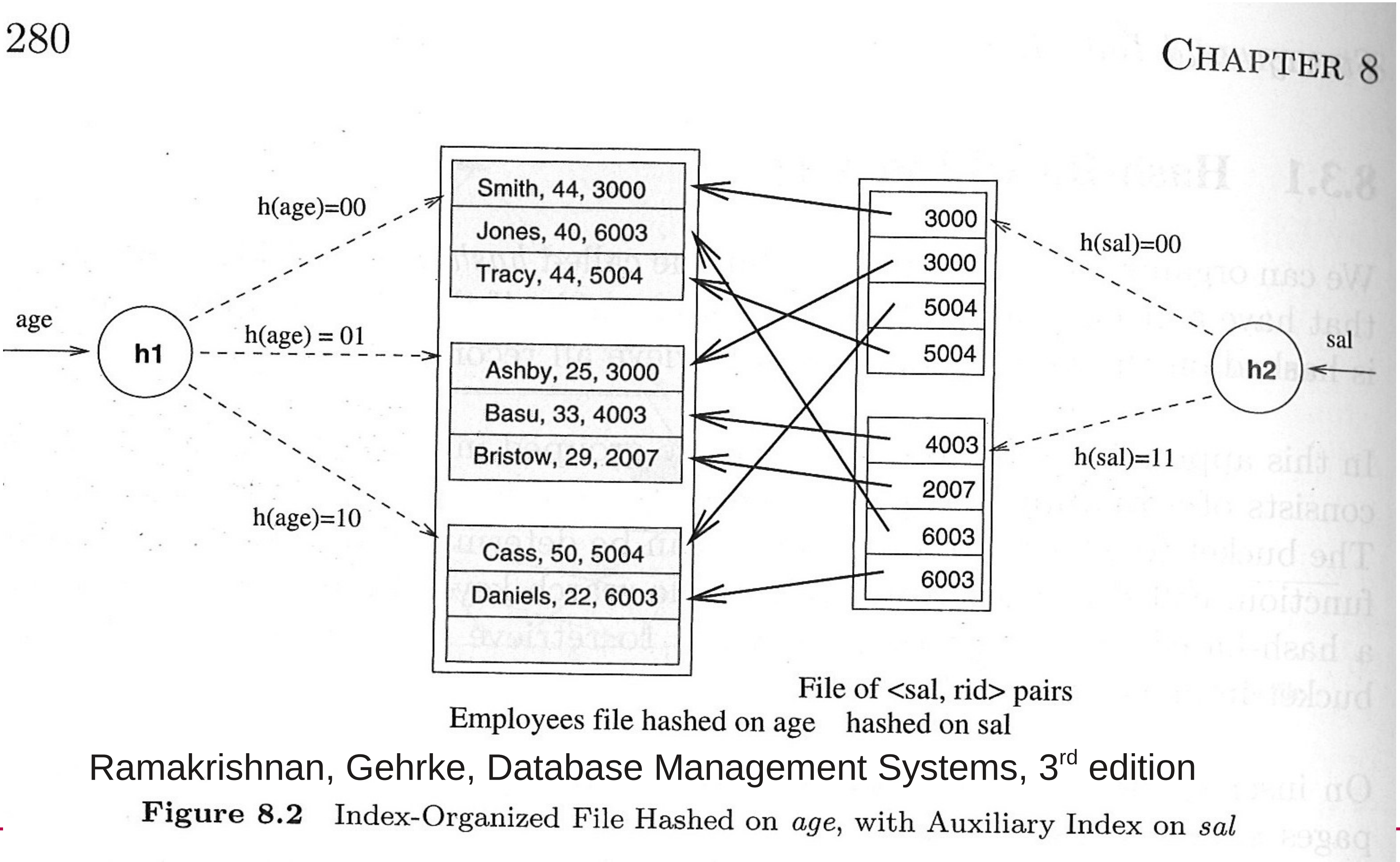
Index Data Structures

- 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



280 CHAPTER 8

Employees file hashed on age

File of <sal, rid> pairs hashed on sal

Ramakrishnan, Gehrke, Database Management Systems, 3rd edition

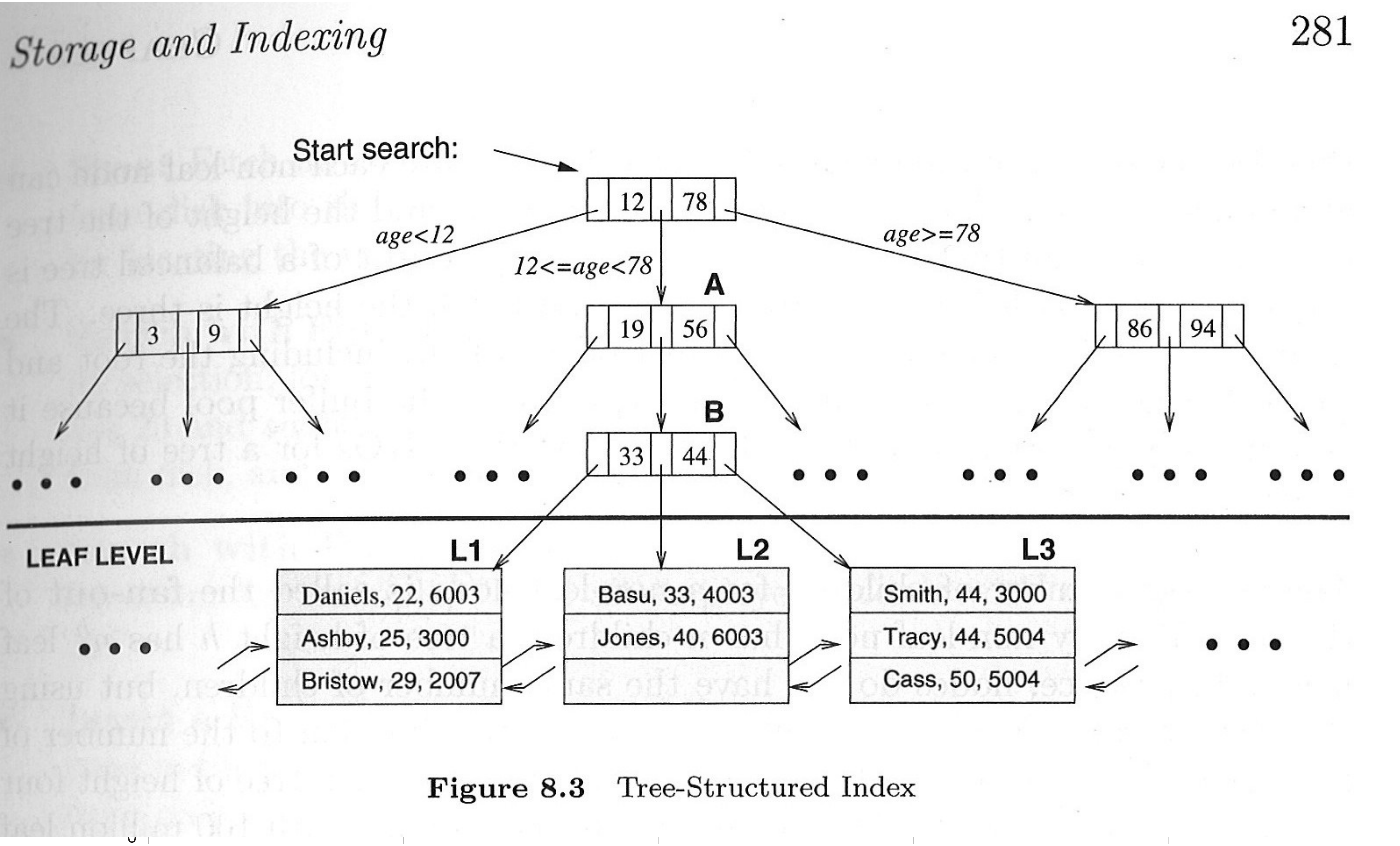
Figure 8.2 Index-Organized File Hashed on *age*, with Auxiliary Index on *sal*

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
  - 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



Storage and Indexing 281

Figure 8.3 Tree-Structured Index

0

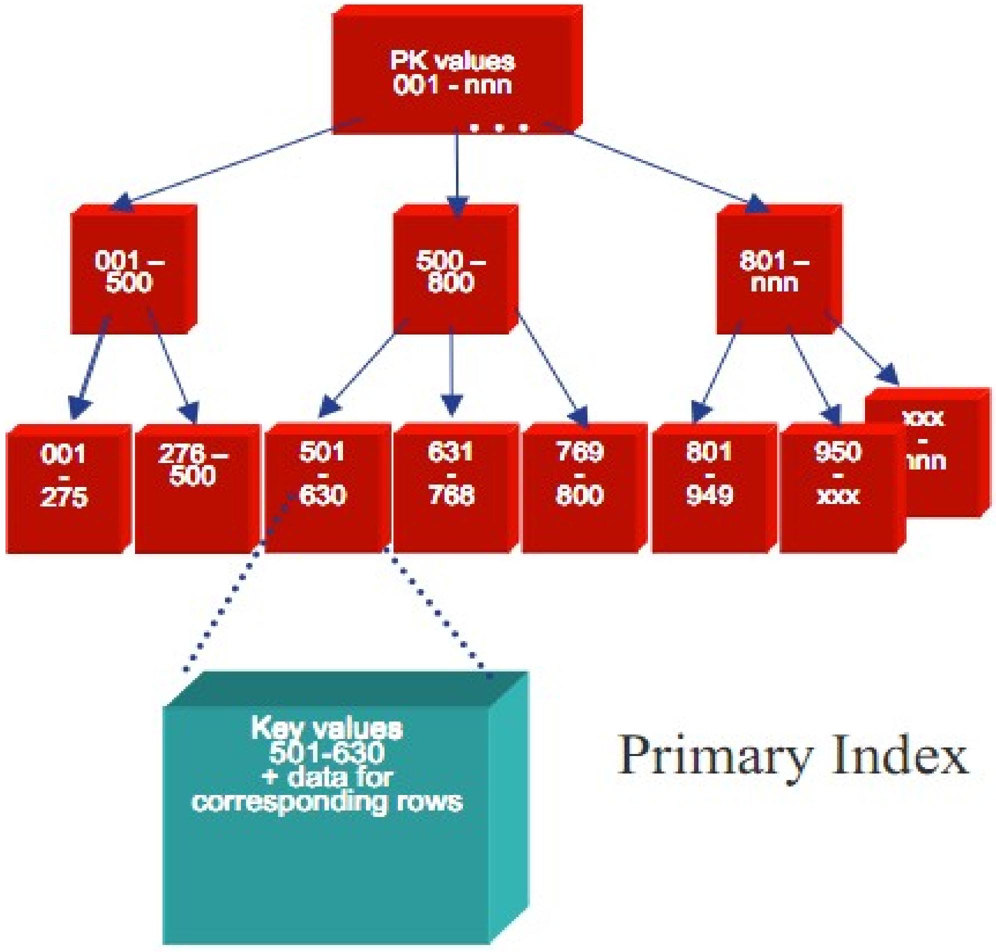
Row 1 Row 2 Row 3 Row 4

Ramakrishnan, Gehrke, Database Management Systems, 3rd edition

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

InnoDB Indexes - Primary



Primary Index

- Data rows are stored in the B-tree leaf nodes of a clustered index
  - B-tree is organized by primary key or non-null unique key of table, if defined; else, an internal column with 6-byte ROW_ID is added.

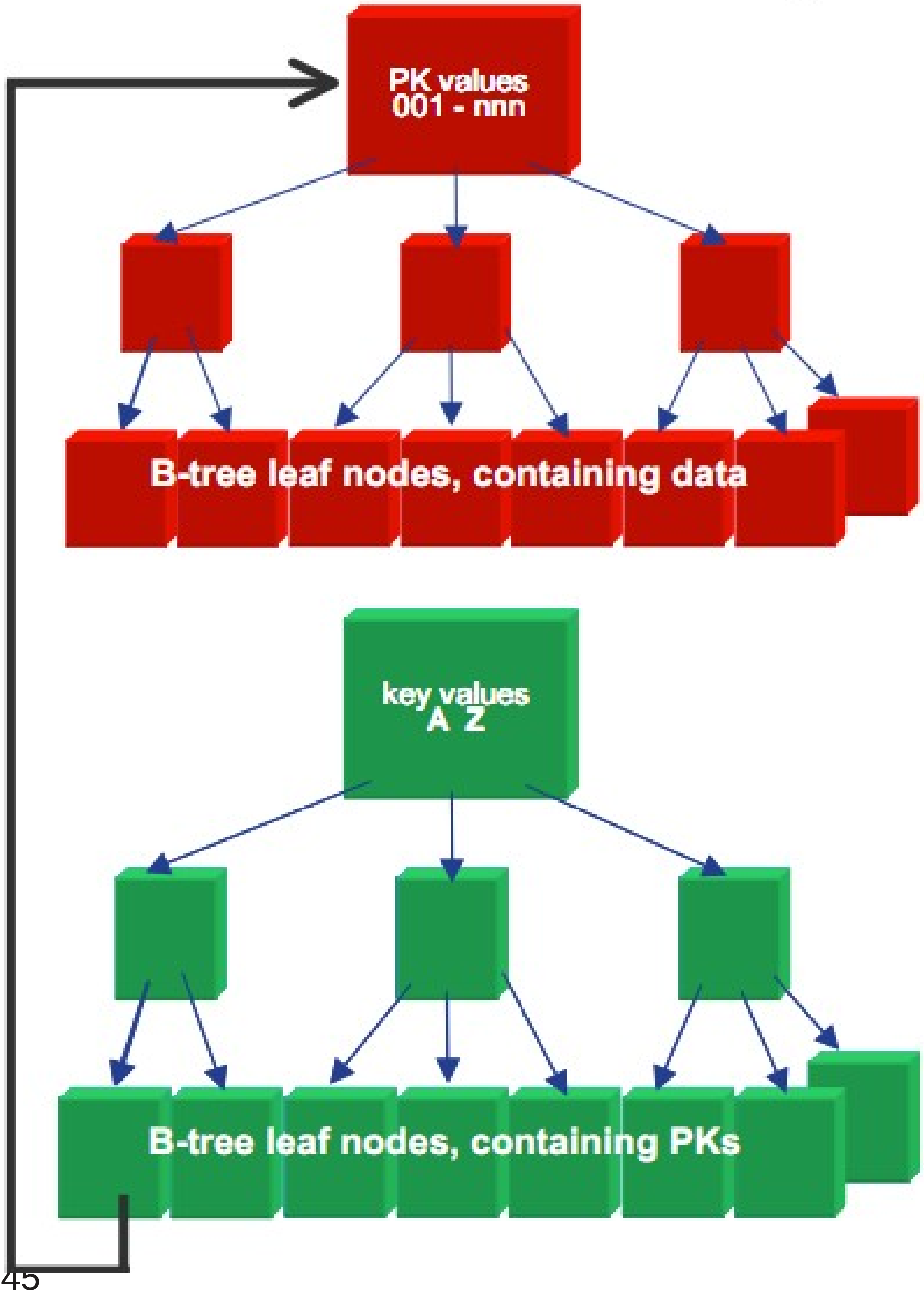
http://www.innodb.com/wp/wp-content/uploads/2009/05/innodb-file-formats-and-source-code-structure.pdf

INNOBASE

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

InnoDB Indexes - Secondary

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


PK values
001 - nnn
B-tree leaf nodes, containing data
key values
A Z
B-tree leaf nodes, containing PKs

Secondary Index

Resources

- http://en.oreilly.com/mysql2011/public/schedule/proceedings
  - **A Beginner's Guide to MariaDB**
    - community version of MySQL
  - **InnoDB: Status, Architecture, and Latest Enhancements**
- http://dev.mysql.com/doc/refman/5.5/en/innodb-index-types.html
- http://dev.mysql.com/doc/refman/5.5/en/innodb-introduction-features.html