Indexes

CS 327E
October 23, 2017

Announcements:

- Midterm: Next class in **ETC 2.108**
- Review session: Tomorrow at 11am in GDC 5.304
- Final Project phase: Starts next week

Midterm:

- Closed book exam
- No cheat sheets allowed :((
- Lasts 90 minutes
- Covers all topics to-date, including indexes
- 17 True/False questions
- 10 Multiple Choice questions
- 7 SQL questions

1) How does an index improve a query's access path?

A) It cuts down on the number of rows that need to be scanned.
B) It cuts down on the number of columns that need to be scanned.
C) It splits up the data across multiple DB instances.
D) It replicates the data across multiple DB instances.

2) Which of the following costs are associated with indexes?

A) Indexes slow down update operations.
B) Indexes slow down insert operations.
C) Indexes slow down delete operations.
D) All of the above.

3) SQL allows an index to be created on multiple columns as long as those columns belong to the same table.

A) True
B) False

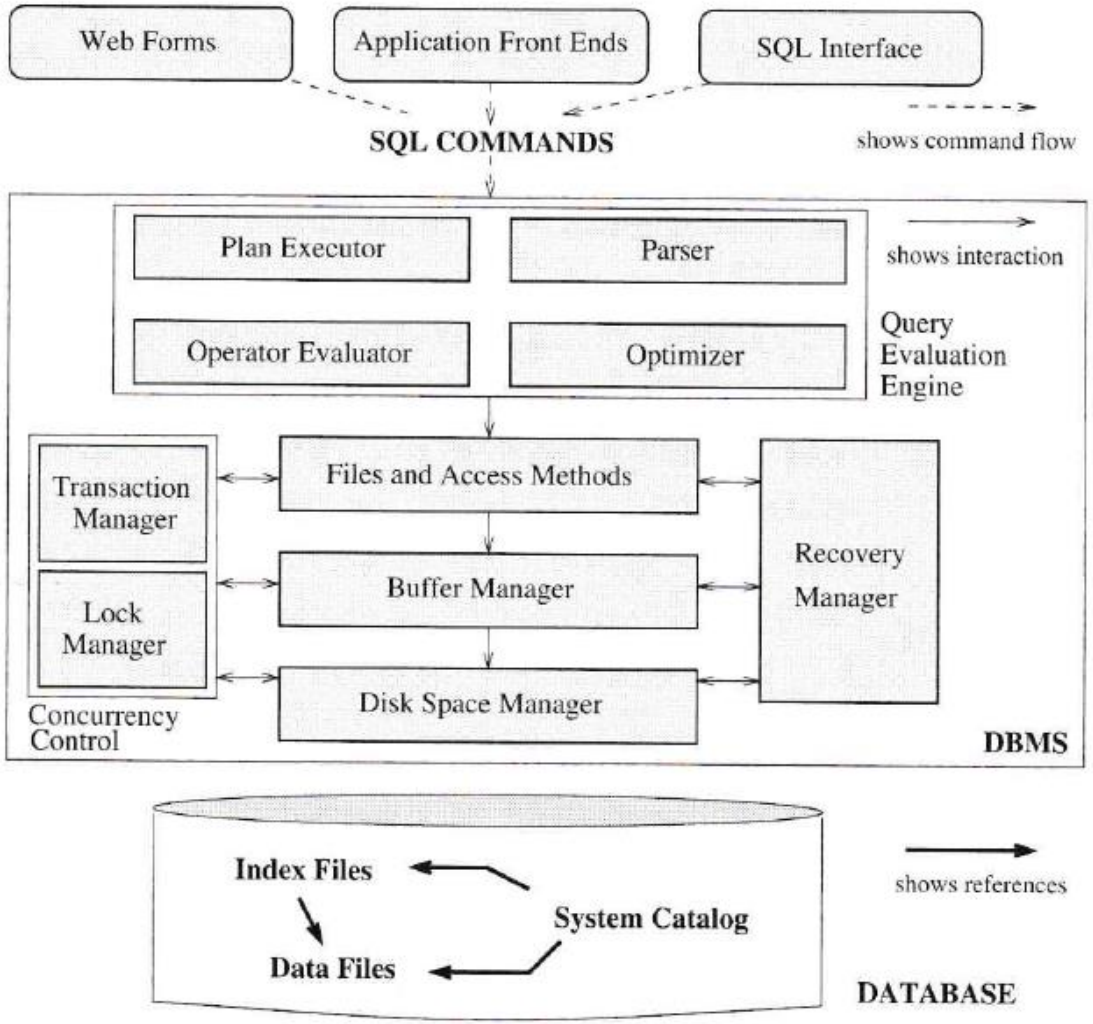
4) An index on a boolean column is generally helpful.

A) Yes
B) No

5) Consider a 100m row table with 250+ columns. This table is being used to run large aggregate queries that access most of the rows, but only a few attributes at one time. Can partitioning this table potentially speed-up the queries? If so, what type of partitioning?

A) Yes, using vertical partitioning.
B) Yes, using horizontal partitioning.
C) No, partitioning is unlikely to have much impact.

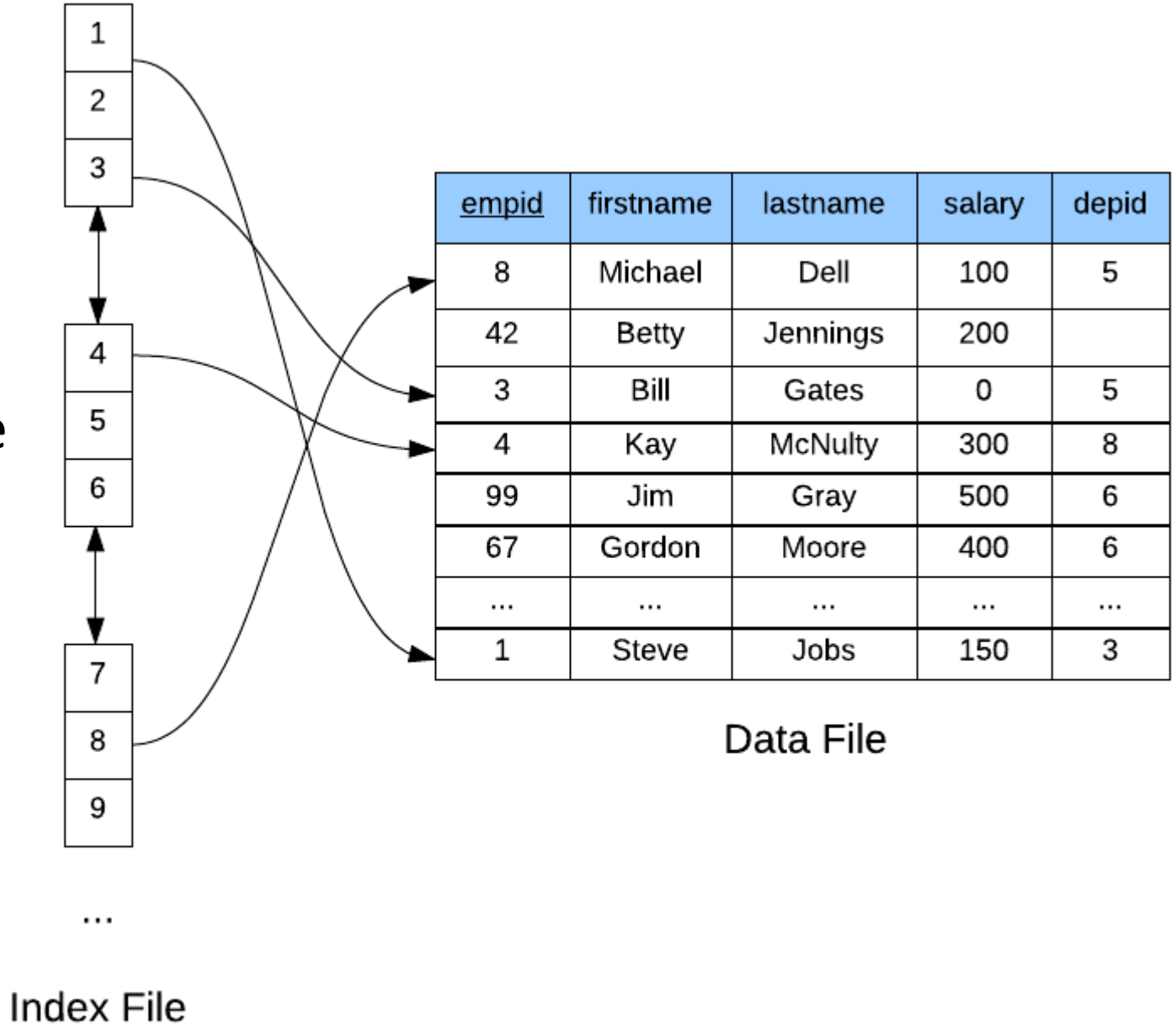
Preliminaries



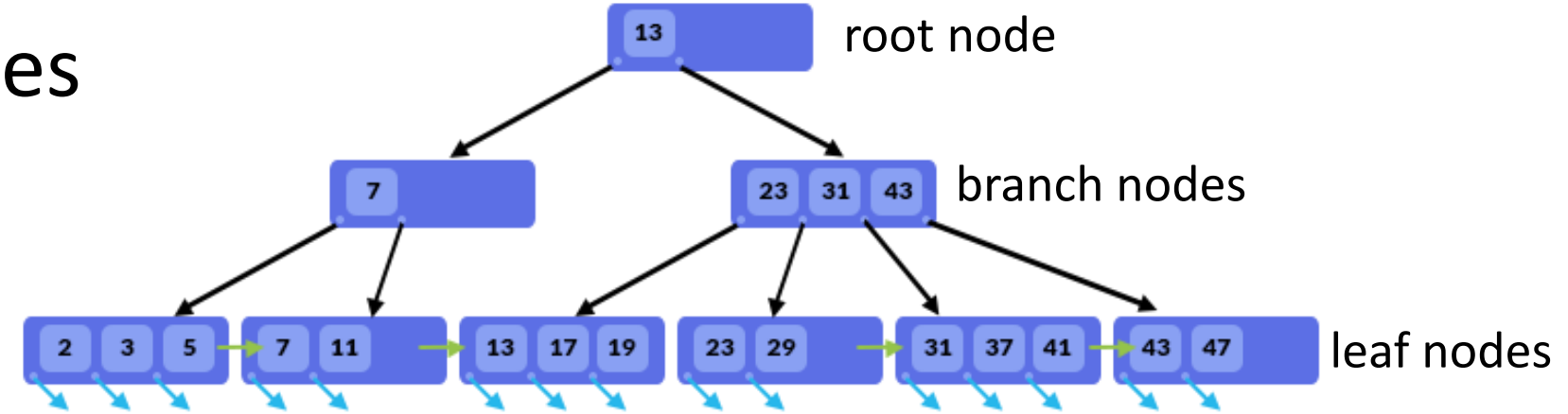
Source: Ramakrishnan and Gehrke, DBMS Systems, 3rd edition, 2003.

Database Indexes

- **Critical** to database systems
- At least one index per table
- DBA analyzes workload and decides which indexes to create (no easy answers)
- Creating indexes can be an expensive operation
- They work “behind the scenes”
- Query optimizer decides which indexes to use during query execution



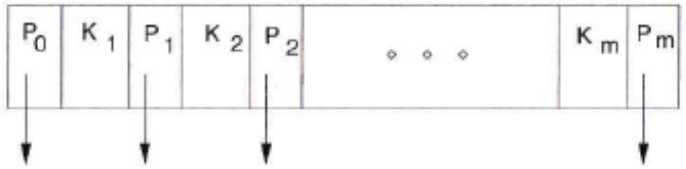
Properties of B+ Trees



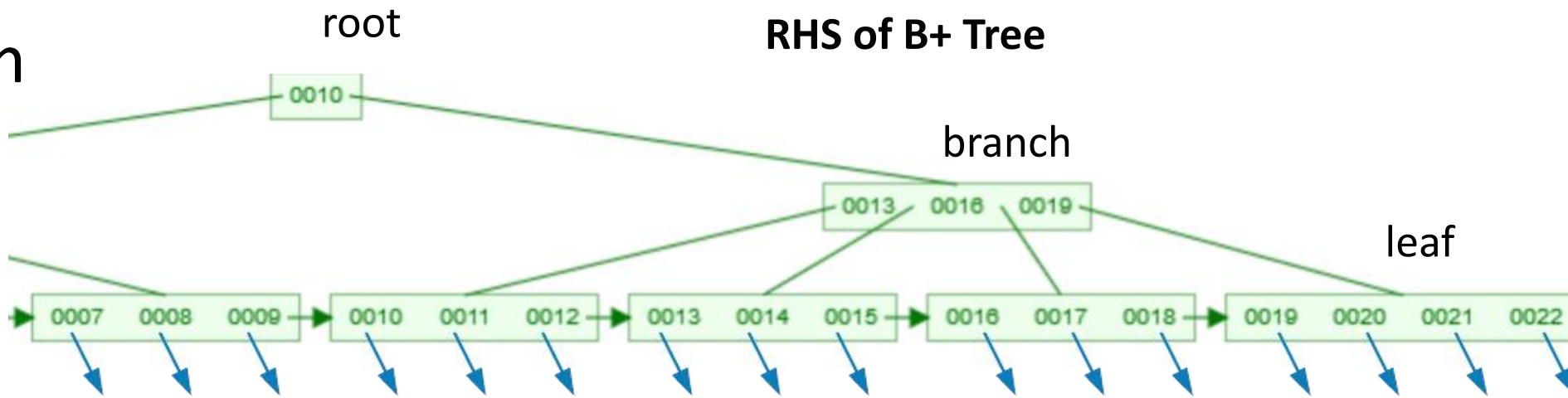
Structure of a B+ Tree

- height is balanced
- has several children
- data stored in the leaf nodes
- leaf nodes are ordered
- leaf nodes are connected (doubly linked list)
- each node stores several index entries
- index entry = (key value, pointer)
- search speed ≈ height of tree

Format of a Node



Search Algorithm



- Let S = Search Key
- Let K = Key Value
- An Index Entry = (P, K)
- Begin at root:
  - If S < K, follow K's left pointer
  - If S = K, follow K's right pointer
  - If S > K and K is not in last entry, scan forward to next entry
  - Repeat for each entry until last entry is reached:
    - If S < K, follow K's left pointer
    - If S ≥ K, follow K's right pointer
- Repeat until leaf node is reached
- Scan forward leaf node until K = S
- Follow K's pointer to row id in data file

Demo

Practice Problem 1

Suppose we want to find all the 'Saturday Night Live' episodes using the query:

```
select te.*, tb.primary_title, start_year
from Title_Episodes te join Title_Basics tb on te.title_id = tb.title_id
where te.parent_title_id = (select title_id from Title_Basics
                            where primary_title = 'Saturday Night Live'
                            and title_type = 'tvSeries' and start_year = 1975)
order by season_num, episode_num;
```

This query runs in ~7 sec and we want to get it under 1 sec. Can you suggest some indexes that would improve the access path of this query?

Note that the primary key columns (Title_Basics.title_id and Title_Episodes.title_id) have already been indexed by the DBMS.

Practice Problem 1

Suppose we want to find all the 'Saturday Night Live' episodes using the query:

```
select te.*, tb.primary_title, start_year
from Title_Episodes te join Title_Basics tb on te.title_id = tb.title_id
where te.parent_title_id = (select title_id from Title_Basics
                            where primary_title = 'Saturday Night Live'
                            and title_type = 'tvSeries' and start_year = 1975)
order by season_num, episode_num;
```

How many indexes would you create to speed up this query?

A) 0 indexes
B) 1 index
C) 2 indexes
D) ≥ 3 indexes