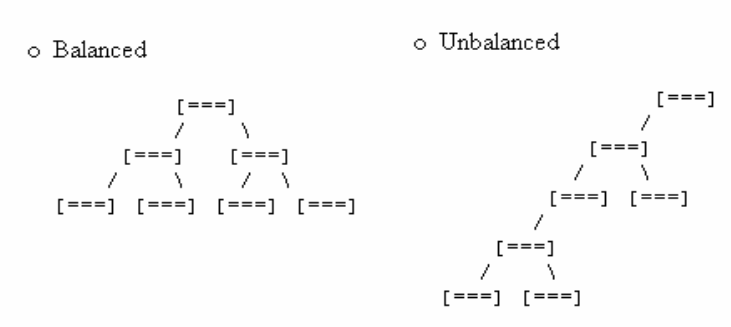
# Balanced vs. Unbalanced Trees

8

- In a balanced tree, every path from the route to a leaf node is the same length.




Credit: S. Lee

# Prefix Key Compression

9

- Key values in index entries only `direct traffic'; can often compress them.
  - E.g., If we have adjacent index entries with search key values *Dannon Yogurt*, *David Smith* and *Devarakonda Murthy*, we can abbreviate *David Smith* to *Dav*. (The other keys can be compressed too ...)

# Hash Based Indexes

10

- Good for equality selections.
- Index is a collection of *buckets.*
  - Bucket = *primary* page plus zero or more *overflow* pages.
  - Buckets contain data entries.
- *Hashing function* **h**: **h**($r$) = bucket in which (data entry for) record $r$ belongs. **h** looks at the *search key* fields of $r$.
  - *No need for "index entries" in this scheme.*

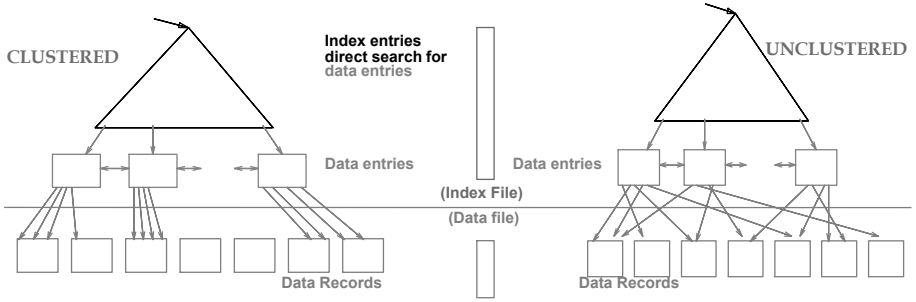
# Index Classification

11

- *Primary* vs. *secondary*: If search key contains primary key, then called primary index.
  - *Unique* index: Search key contains a candidate key.
- *Clustered* vs. *unclustered*: If order of index data entries is the same as order of data records, then called clustered index.
  - A file can be clustered on at most one search key.

# Clustered vs. Unclustered Index

12




# Understanding the Workload

13

- For each query in the workload:
  - Which relations does it access?
  - Which attributes are retrieved?
  - Which attributes are involved in selection/join conditions? How selective are these conditions likely to be?
- For each update in the workload:
  - The type of update (INSERT/DELETE/UPDATE), and the attributes that are affected.

# Choice of Indexes

14

- What indexes should we create?
  - Which relations should have indexes? What field(s) should be the search key? Should we build several indexes?
- For each index, what kind of an index should it be?
  - Clustered? Hash/tree?

# Choice of Indexes (cont'd)

15

- One approach: Consider the most important queries in turn. Consider the best plan using the current indexes, and see if a better plan is possible with an additional index.
  - Implies an understanding of how a DBMS evaluates queries and creates query evaluation plans.
- Before creating an index, must also consider the impact on updates in the workload!
  - Trade-off: Indexes can make queries go faster, updates slower. Require disk space, too.

# Guidelines

16

- Attributes in WHERE clause are candidates for index keys.
  - Exact match condition suggests hash index.
  - Range query suggests tree index.
- Multi-attribute search keys should be considered when a WHERE clause contains several conditions.
- Try to choose indexes that benefit as many queries as possible.
  - Since only one index can be clustered per relation, choose it based on important queries that would benefit the most from clustering.

# Examples

17

- B+ tree index on E.age can be used to get qualifying tuples.
  - Is the index clustered?

```
SELECT E.dno
FROM Emp E
WHERE E.age>40
```

- Equality queries and duplicates:
  - Indexing on *E.hobby* helps

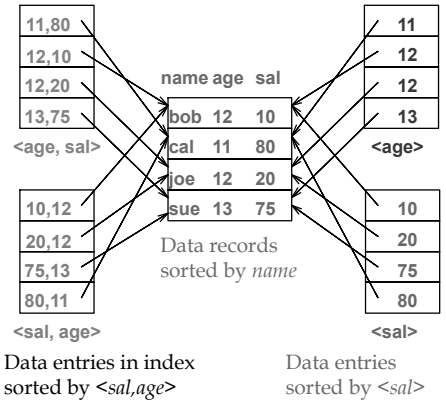
```
SELECT E.dno
FROM Emp E
WHERE E.hobby=Stamps
```

# Composite Search Keys

18

- *Composite Search Keys*: Search on a combination of fields.
  - Equality query: Every field value is equal to a constant value. E.g. wrt <sal,age> index:
    - age=20 and sal =75
  - Range query:
    - age=20 and sal > 10
- Data entries in index sorted by search key to support range queries.
  - Lexicographic order

Examples of composite key indexes using lexicographic order.




# Database Tuning

19

- A major problem in making a database run fast is deciding which indexes to create.
- Pro: An index speeds up queries that can use it.
- Con: An index slows down all modifications on its relation because the index must be modified too.

# Example: Tuning



- Suppose the only things we did with our beers database was:
  1. Insert new facts into a relation (10%).
  2. Find the price of a given beer at a given bar (90%).
- Then SellInd on Sells(bar, beer) would be wonderful, but BeerInd on Beers(manf) would be harmful.