Index Classification

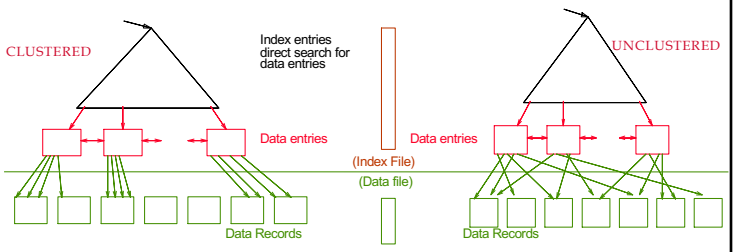
12

- 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 table can have at most one clustered index – why?

12

Clustered vs. Unclustered Index

13



13

Declaring Indexes

14

- No standard!
- Typical syntax:

```
CREATE INDEX BeerInd ON Beers(manf);
CREATE INDEX SellInd ON Sells(bar,
  beer);
```

14

Using Indexes

15

- Given a value *v*, the index takes us to only those tuples that have *v* in the attribute(s) of the index.
- Example: use BeerInd and SellInd to find the prices of beers manufactured by Pete's and sold by Joe.

15

Using Indexes --- (2)

16

```
SELECT price
FROM Beers, Sells
WHERE manf = 'Pete''s' AND
    Beers.name = Sells.beer AND
    bar = 'Joe''s Bar';
```

1. Use BeerInd to get all the beers made by Pete's.
2. Then use SellInd to get prices of those beers, with bar = 'Joe''s Bar'

16

Understanding the Workload

17

- 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.

17

Choice of Indexes

18

- 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?

18

Choice of Indexes (cont'd)

19

- 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.

19

Guidelines

20

- 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.

20

Examples

21

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

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

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

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

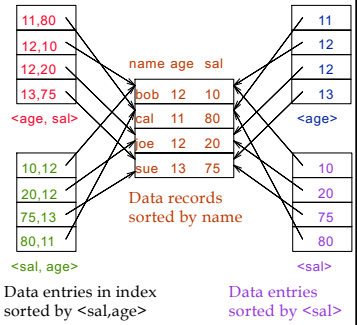
21

Composite Search Keys

22

- 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.

Examples of composite key indexes



22

Database Tuning

23

- 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.

23

Example: Tuning

24

- Suppose the only things we did with our beers database was:
  1. Insert new beers 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 helpful, but BeerInd on Beers(manf) would be harmful.

24

Tuning Advisors

25

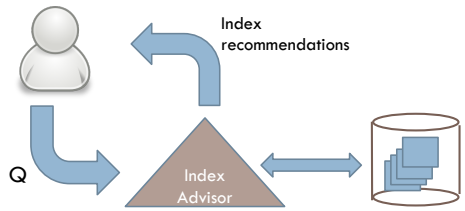
- A major research area
  - Because hand tuning is so hard.
- An advisor gets a *query load*, e.g.:
  1. Choose random queries from the history of queries run on the database, or
  2. Designer provides a sample workload.

25

Tuning Advisors --- (2)

26

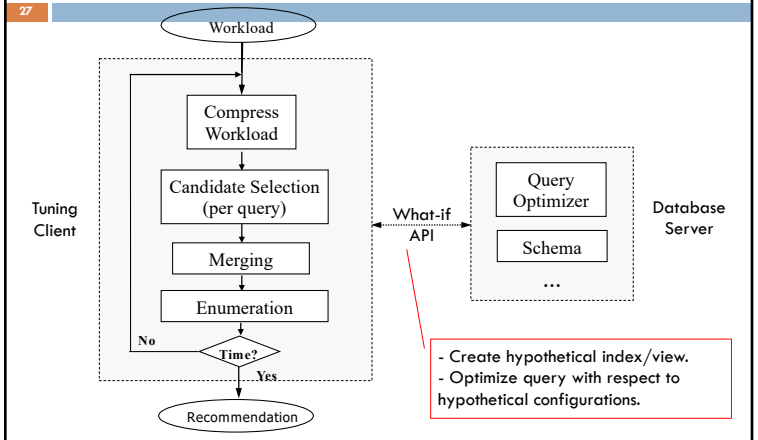
- The advisor generates candidate indexes and evaluates each on the workload.
  - Measure the improvement/degradation in the average running time of the queries.



26

Example: Database Tuning Architecture

27



27

Summary

28

- If selection queries are frequent, sorting the file or building an index is important.
  - Hash-based indexes only good for equality search.
  - Tree-based indexes best for range search; also good for equality search.

28

Summary (cont'd)

29

- Can have several indexes on a given file of data records, each with a different search key.
- Understanding the nature of the workload for the application
  - What are the important queries and updates? What attributes/relations are involved?
- Indexes are chosen to speed up important queries (and perhaps some updates!).
  - Consider index maintenance overhead on updates to key fields.
  - Choose indexes that can help many queries, if possible.

29