PHYSICAL OPTIMIZATION

Iztok Savnik

Literature

- Sam Lightstone, Toby Teorey, Tom Nadeau, Physical Database Design, Morgan Kaufmann Publishers, 2007.
  - Chapter 4

Indexing concepts

- Indexes are one of the primary tools used in DBMS-S
  - Wide range of queries typically requested
- Indexes have wide range of purposes
  - Fast lookup for specific or range data
  - Uniqueness enforcement
  - Index-only answers
  - Sorting the data
  - ...

Indexing concepts

- Logical design
  - Clean and coherent design
  - Implementation follows
- Physical design
  - System is running and monitored
  - Performance tuning
    - Increase database throughput
    - Reduce database response time
  - ... for a set of transactions, queries, updates
- Find and eliminate bottlenecks

Physical design

- Hardware level
  - Reduce bottlenecks by increasing the performance
    - CPU, memory, disks, RAID, ...
- Database system level
  - Increase buffers, relax checkpoints, no transactions, ...
- Schema and transaction level
  - Redesign of transactions for speed
  - Careful re-writing SQL queries
  - Denormalization of tables
  - Materialized views, partitioning, index selection

Basic types of indexes

- B+ tree
- Hash table index
- Composite index
- Clusered index
- Covering index (index only)
- Bitmapped index
- Dense versus sparse index

Access methods for indexes

- Table scanning
- Index scanning
  - Clustered and nonclustered indexes
- Index-only scanning
  - Covering index
- Block and row index ANDing
  - Merge index entries for multipoint queries
- List prefetch
  - Sort on RIDs
  - Fetch sorted rows

Indexing rules of thumb

1. Index every primary key and most foreign keys
2. Attributes frequently referenced by SQL WHERE are good candidates for index
3. Use B+ trees for range and equality queries
4. Choose carefully one clustered index for each table
5. Avoid or remove redundant indexes
6. Add indexes when absolutely neccessary
7. Add or delete index columns for composite indexes. Do not alter primary key columns.

Indexing rules of thumb

8. Use attributes for indexes with caution when there are frequent updates
9. Keep up index maintenance on regular bases; drop indexes when they are clearly hurting perforance
10. Avoid extremes in index cardinality and value distribution
11. Covering indexes are useful but oftenly over
12. Use bitmap indexes for high-volume data, especially for data warehouses.

Index selection decisions

- Incremental design of indexes for a given database
  - Throughput and response time
  - No standard rule here
  - Need to consider updates
  - Rules from Ramakrishnan's textbook

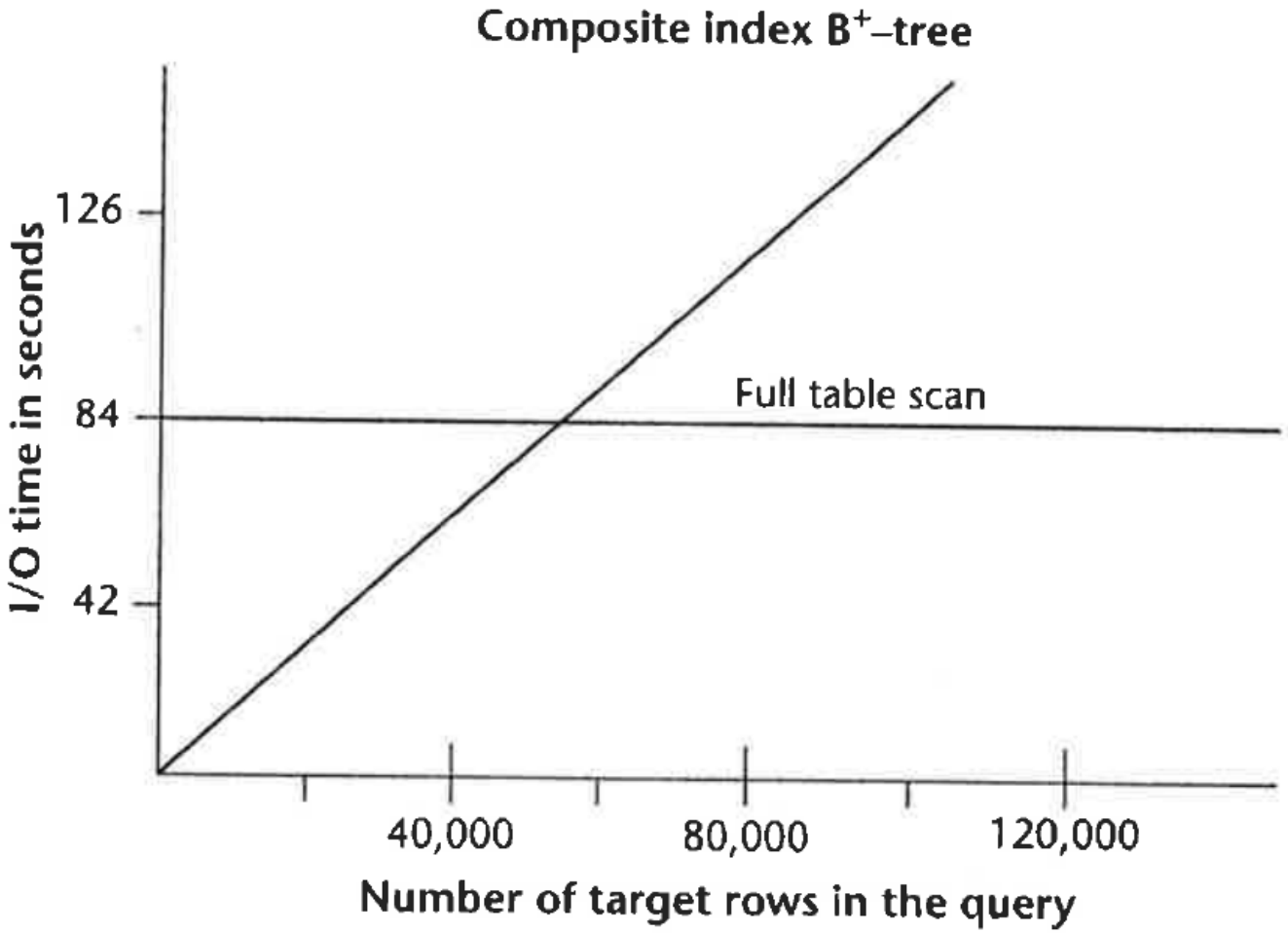
Index selection decisions

- Design decision 1:
  - Does this table require an index or not?
  - If so, which search key should I build an index on?
- Rules
  - Index Pks and Fks
  - Check conditions in WHERE statements
  - Indexes are chosen for each query
  - Indexes that speed up more than one query

Index selection decisions

- Design decision 2:
  - When do I need multi-attribute (composite) search keys?
  - Which ones should I choose?
- Rules
  - A multipoint query involves a WHERE clause that has multiple attributes
  - Single composite index on n attributes can be significantly faster than n separate indexes
    - Merge step to find intersection RIDs

Index selection decisions



Index selection decisions

- Design decision 3:
  - Should I use a dense or sparse index?
- Rules
  - When rows are small comparing to page
    - Dense index will have many etries (more levels)
  - Dense indexes can be efectively merged into composite indexes for multipoint queries
  - When rows are large comparing to page
    - Sparse indexplansonlx is rearly more efficient
  - Dense index can be used in index-only plans

Index selection decisions

- Design decision 4:
  - When can I use covering index?
- Rules
  - Composite index can be used for certain queries as a covering index
  - Index on attributes feature, make and model can be used to completely satisfy the query

```
SELECT make, model, vin, year
FROM carOnLot
WHERE feature = 'catalytic converter';
```

Index selection decisions

- Design decision 5:
  - Should I create clustering index?
- Rules
  - Only one clustering index can be created for a given table
  - If an index can be used as covering index then no clustering is needed
  - Range queries, multipoint queries and single-point queries on non-primary key may all benefit from clustering

Index selection decisions

- Range queries, multipoint queries and single-point queries on non-primary key may all benefit from clustering
  - Accessing multiple rows => any custering would improve performance
  - When there are multiple choices then the tradeoff analysis may be needed

Index selection decisions

- Design decision 6:
  - Is an index still preferred when updates are taken into account?
  - What are the tradeoffs between read and update queries for each index choosen?
- Rules
  - Once the indexes are chosen to improve performance of for known queries, consider inserts, deletes and updates on the target tables
  -

Index selection decisions

- Consider I/O time for all transactions involving queries and updates of each target table
- Take into account the frequency of each query and update over a fixed time period

Benefit of the index
= I/O time (all queries without index) –
I/O time (all queries with index).

Cost of the index
= I/O time (all updates with index) –
I/O time (all updates without index).

Index selection decisions

- Notes:
  - Create index if the benefit is greater then the cost
  - This rule is beased solely on I/O time
    - Exceptions must be considered
  - Priority of queries may be much higher than updates
    - Updates can be batch processed with few hours delay
    - It may also be vice versa: queries can be delayed...
- Note that each update has two components
  - Query to access the rows to be updated
  - Actual update itself
- Clustered indexes tend to have higher update cost then unclustered

Index selection decisions

- Design decision 7:
  - How do I know I made right index choice?
- Rules
  - Investigate if right choice has been done
  - Analytical performance tradeoff analysis
    - Estimate the number of I/O time needed to answer the known queries on known table or set of tables and index
  - Once index is set up new queries often emerge
    - New queries use new index

Index selection decisions

- Data needs to be collected before and after index is implemented
  - Determine whether index is making database perform better
- Data collection facilities are now common to the major vendors
  - IBM, DB2 Instrumentation Facility
  - Microsoft, SQL Server, Performance Monitor
  - Oracle, Automatic Workload Repository
- If there is no improvement or degradation in performance then search aternatives