# Discuss query optimization techniques

## Indexes



Syntax:

CREATE [ UNIQUE | BITMAP ] INDEX index\_name ON table\_name ( {column\_name | column\_expressions} [ASC | DESC] [, …] ) [various storage attributes];

* BITMAP index –
* UNIQUE index –
* Terms:
  + B-tree index –
  + Clustered index –
  + Composite index –
  + Function-based index –

# How, what, why to Index

### Advantage of indexing:

### Disadvantages of indexing:

### Considerations

* Frequency of queries (reads) versus frequency of updates (writes)
* Likelihood that queries and updates are going to be slow
* Whether to use special types of indexes (e.g. bitmap indexes)

### Guidelines

* Use indexes:
* Don’t use indexes:

# Query Optimization

SQL is non-procedural – you specify what, not how. The DBMS determines how the work will be accomplished.

The DBMS has an optimizer that attempts to choose the most efficient method to execute the statement – for example, which table to access in which order, what indexes to use, etc.

* Rule-based –
* Cost-based –

There are ways to re-phrase some queries to improve the efficiency of the queries – these things are very system specific (e.g. Oracle quite different than SQL server), so check the docs. Examples include:

### General Guidelines:

# Query Optimization Tips/Rules

* Use a WHERE clause to filter rows
  + Example: SELECT \* FROM CUSTOMERS
    - Vs
  + SELECT \* FROM CUSTOMERS WHERE cnum IN (2001, 2002);
* Use Table joins instead of multiple queries
  + Example: SELECT cname FROM Customers WHERE cnum = 2004; and then another query SELECT onum FROM Orders WHERE cnum = 2004;
    - Vs
  + SELECT onum, cname FROM CUSTOMERS c JOIN Orders o ON c.cnum = o.cnum WHERE c.cnum = 2004;
  + You should join the tables such that the table with the least number of hits is joined last. Ex, tab1 1000 rows, tab2 100 rows, tab3 10 rows, we’d want to join them in that order.
* Use Fully Qualified Column References When Performing Joins
  + Example: SELECT **o.onum, c.cname** FROM CUSTOMERS c JOIN Orders o ON c.cnum = o.cnum WHERE c.cnum = 2004;
* Use indexes on tables
* Use WHERE rather than HAVING
  + Example: SELECT odate, avg(amt) FROM Orders GROUP BY odate HAVING odate IN (’03-OCT-00’, ’04-OCT-00’);
* Use EXISTS rather than IN for correlated subqueries
* Use EXISTS Rather than DISTINCT

## Comparing Execution Plans

* We can compare the cost of two execution plans together, the one with the lower cost should be more efficient.