[**Data Warehousing**](http://www.1keydata.com/datawarehousing/datawarehouse.html) > [**Data Warehouse Design**](http://www.1keydata.com/datawarehousing/processes.html) > **Query Optimization**

For any production database, SQL query performance becomes an issue sooner or later. Having long-running queries not only consumes system resources that makes the server and application run slowly, but also may lead to table locking and data corruption issues. So, query optimization becomes an important task.

First, we offer some guiding principles for query optimization:

**1. Understand how your database is executing your query**

Nowadays all databases have their own query optimizer, and offers a way for users to understand how a query is executed. For example, which index from which table is being used to execute the query? The first step to query optimization is understanding what the database is doing. Different databases have different commands for this. For example, in MySQL, one can use "EXPLAIN [SQL Query]" keyword to see the query plan. In Oracle, one can use "EXPLAIN PLAN FOR [SQL Query]" to see the query plan.

**2. Retrieve as little data as possible**

The more data returned from the query, the more resources the database needs to expand to process and store these data. So for example, if you only need to retrieve one column from a table, do not use 'SELECT \*'.

**3. Store intermediate results**

Sometimes logic for a query can be quite complex. Often, it is possible to achieve the desired result through the use of subqueries, inline views, and UNION-type statements. For those cases, the intermediate results are not stored in the database, but are immediately used within the query. This can lead to performance issues, especially when the intermediate results have a large number of rows.

The way to increase query performance in those cases is to store the intermediate results in a temporary table, and break up the initial SQL statement into several SQL statements. In many cases, you can even build an index on the temporary table to speed up the query performance even more. Granted, this adds a little complexity in query management (i.e., the need to manage temporary tables), but the speedup in query performance is often worth the trouble.

***Below are several specific query optimization strategies.***

* **Use Index**  
  Using an index is the first strategy one should use to speed up a query. In fact, this strategy is so important that index optimization is also discussed.
* **Aggregate Table**  
  Pre-populating tables at higher levels so less amount of data need to be parsed.
* **Vertical Partitioning**  
  Partition the table by columns. This strategy decreases the amount of data a SQL query needs to process.
* **Horizontal Partitioning**  
  Partition the table by data value, most often time. This strategy decreases the amount of data a SQL query needs to process.
* **Denormalization**  
  The process of denormalization combines multiple tables into a single table. This speeds up query performance because fewer table joins are needed.
* **Server Tuning**  
  Each server has its own parameters, and often tuning server parameters so that it can fully take advantage of the hardware resources can significantly speed up query performance.