Indexes come with a performance cost, but are more than worth the cost for frequent queries on large data set. Consider the relative frequency of each query in the application and whether the query justifies an index.

The best overall strategy for designing indexes is to profile a variety of index configurations with data sets similar to the ones you'll be running in production to see which configurations perform best.Inspect the current indexes created for your collections to ensure they are supporting your current and planned queries. If an index is no longer used, drop the index.

Generally, MongoDB only uses *one* index to fulfill most queries. However, each clause of an `$or` query may use a different index, and starting in 2.6, MongoDB can use an *intersection* (page 510) of multiple indexes.

The following documents introduce indexing strategies:

***Create Indexes to Support Your Queries* (page 547)** An index supports a query when the index contains all the fields scanned by the query. Creating indexes that supports queries results in greatly increased query performance.

***Use Indexes to Sort Query Results* (page 548)** To support efficient queries, use the strategies here when you specify the sequential order and sort order of index fields.

***Ensure Indexes Fit in RAM* (page 550)** When your index fits in RAM, the system can avoid reading the index from disk and you get the fastest processing.

***Create Queries that Ensure Selectivity* (page 550)** Selectivity is the ability of a query to narrow results using the index. Selectivity allows MongoDB to use the index for a larger portion of the work associated with fulfilling the query.

Create Indexes to Support Your Queries

An index supports a query when the index contains all the fields scanned by the query. The query scans the index and not the collection. Creating indexes that support queries results in greatly increased query performance.

This document describes strategies for creating indexes that support queries.

Create a Single-Key Index if All Queries Use the Same, Single Key

If you only ever query on a single key in a given collection, then you need to create just one single-key index for that collection. For example, you might create an index on `category` in the `product` collection:

```
db.products.createIndex( { "category": 1 } )
```

Create Compound Indexes to Support Several Different Queries

If you sometimes query on only one key and at other times query on that key combined with a second key, then creating a compound index is more efficient than creating a single-key index. MongoDB will use the compound index for both queries. For example, you might create an index on both `category` and `item`.

```
db.products.createIndex( { "category": 1, "item": 1 } )
```

This allows you both options. You can query on just `category`, and you also can query on `category` combined with `item`. A single *compound index* (page 489) on multiple fields can support all the queries that search a "prefix" subset of those fields.

Example

The following index on a collection: