```
                  ]
}
```

You could then create the geospatial index on the `addresses.loc` field as in the following example:

```
db.records.createIndex( { "addresses.loc": "2d" } )
```

To include the location field with the distance field in multi-location document queries, specify `includeLocs: true` in the `geoNear` command.

Text Indexes

New in version 2.4.

MongoDB provides `text` indexes to support text search of string content in documents of a collection.

`text` indexes can include any field whose value is a string or an array of string elements. To perform queries that access the `text` index, use the `$text` query operator.

Changed in version 2.6: MongoDB enables the text search feature by default. In MongoDB 2.4, you need to enable the text search feature manually to create `text` indexes and perform *text search* (page 502).

Create Text Index

To create a `text` index, use the `db.collection.createIndex()` method. To index a field that contains a string or an array of string elements, include the field and specify the string literal `"text"` in the index document, as in the following example:

```
db.reviews.createIndex( { comments: "text" } )
```

A collection can have at most **one** `text` index.

However, you can specify multiple fields for the `text` index. For examples of creating `text` indexes on multiple fields, see *Create a text Index* (page 539) and *Wildcard Text Indexes* (page 500).

Wildcard Text Indexes

To allow for text search on all fields with string content, use the wildcard specifier (`$**`) to index all fields in the collection that contain string content. Such an index can be useful with highly unstructured data if it is unclear which fields to include in the text index or for ad-hoc querying.

With a wildcard text index, MongoDB indexes every field that contains string data for each document in the collection. The following example creates a text index using the wildcard specifier:

```
db.collection.createIndex( { "$**": "text" } )
```

Wildcard text indexes are `text` indexes on multiple fields. As such, you can assign weights to specific fields during index creation to control the ranking of the results. For more information using weights to control the results of a text search, see *Control Search Results with Weights* (page 543).

Wildcard text indexes, as with all text indexes, can be part of a compound indexes. For example, the following creates a compound index on the field `a` as well as the wildcard specifier:

```
db.collection.createIndex( { a: 1, "$**": "text" } )
```