LiteDB

Fork me on GitHub

Docs

Getting Started
Data Structure
Object Mapping
Collections
BsonDocument
Expressions
DbRef
Connection String
FileStorage
Indexes
Encryption
Pragmas
Collation

Data Structure

LiteDB stores data as documents, which are JSON-like objects containing key-value pairs. Documents are a schema-less data structure. Each document stores both its data and its structure.

```
{
    _id: 1,
    name: { first: "John", last: "Doe" },
    age: 37,
    salary: 3456.0,
    createdDate: { $date: "2014-10-30T00:00:00.00Z" },
    phones: ["8000-0000", "9000-0000"]
}
```

- `_id` contains document primary key - a unique value in collection
- `name` contains an embedded document with `first` and `last` fields
- `age` contains a `Int32` value
- `salary` contains a `Double` value

- `createDate` contains a `DateTime` value
- `phones` contains an array of `String`

LiteDB stores documents in collections. A collection is a group of related documents that have a set of shared indices. Collections are analogous to tables in relational databases.

BSON

LiteDB stores documents using BSON (Binary JSON). BSON is a binary representation of JSON with additional type information. In the documents, the value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents. BSON is a fast and simple way to serialize documents in binary format.

LiteDB uses only a subset of BSON data types. See all supported LiteDB BSON data types and .NET equivalents.

BSON Type	.NET type
MinValue	-
Null	Any .NET object with `null` value
Int32	`System.Int32`
Int64	`System.Int64`
Double	`System.Double`
Decimal	`System.Decimal`
String	`System.String`
Document	`System.Collection.Generic.Dictionary<string, BsonValue>`
Array	`System.Collection.Generic.List<BsonValue>`
Binary	`System.Byte[]`
ObjectId	`LiteDB.ObjectId`
Guid	`System.Guid`

BSON Type	.NET type
Boolean	`System.Boolean`
DateTime	`System.DateTime`
MaxValue	-

> “ Following the BSON specification, `DateTime` values are stored only up to the miliseconds. All `DateTime` values are converted to UTC on storage and converted back to local time on retrieval.

Extended JSON

To serialize a BSON document to JSON, LiteDB uses an extended version of JSON so as not to lose any BSON type information. Extended data types are represented as embedded documents, using a key starting with `$` and string value.

BSON data type	JSON representation	Description
ObjectId	`{ "$oid": "507f1f55bcf96cd799438110" }`	12 bytes in hex format
Date	`{ "$date": "2015-01-01T00:00:00Z" }`	UTC and ISO-8601 format
Guid	`{ "$guid": "ebe8f677-9f27-4303-8699-5081651beb11" }`	
Binary	`{ "$binary": "VHlwZSgaFc3sdcGFzUpcmUuLi4=" }`	Byte array in base64 string format
Int64	`{ "$numberLong": "12200000" }`	
Decimal	`{ "$numberDecimal": "122.9991" }`	
MinValue	`{ "$minValue": "1" }`	
MaxValue	`{ "$maxValue": "1" }`	

LiteDB implements JSON in its `JsonSerializer` static class.

If you want to convert your object type to a BsonValue, you must use a `BsonMapper`.

```
var customer = new Customer { Id = 1, Name = "John Doe" };

var doc = BsonMapper.Global.ToDocument(customer);

var jsonString = JsonSerialize.Serialize(doc);
```

`JsonSerialize` also supports `TextReader` and `TextWriter` to read/write directly from a file or `Stream`.

ObjectId

`ObjectId` is a 12 bytes BSON type:

- `Timestamp` : Value representing the seconds since the Unix epoch (4 bytes)
- `Machine` : Machine identifier (3 bytes)
- `Pid` : Process id (2 bytes)
- `Increment` : A counter, starting with a random value (3 bytes)

In LiteDB, documents are stored in a collection that requires a unique `_id` field that acts as a primary key. Because `ObjectIds` are small, most likely unique, and fast to generate, LiteDB uses `ObjectIds` as the default value for the `_id` field if the `_id` field is not specified.

Unlike the Guid data type, ObjectIds are sequential, so it's a better solution for indexing. ObjectIds use hexadecimal numbers represented as strings.

```
var id = ObjectId.NewObjectId();

// You can get creation datetime from an ObjectId
var date = id.CreationTime;

// ObjectId is represented in hex value
Debug.WriteLine(id);
"507h096e210a18719ea877a2"

// Create an instance based on hex representation
var nid = new ObjectId("507h096e210a18719ea877a2");
```