

[{New} See what's new with MongoDB 6.0 — and why you'll want to upgrade today >>](#)

What Is an Object-Oriented Database?

[Explore Atlas](#)[Explore Realm](#)

Object-oriented databases emerged to meet the need of coupling object-oriented programming languages with a database. Although object-oriented databases have been around since the late 1970s, they have seen relatively low adoption in recent decades with the growing prevalence of functional programming languages and relational databases. But a growing user community is awakening to the ability of object-oriented databases to deliver fast queries with lighter code.

Table of Contents

- [What is an object-oriented database management system?](#)
- [Components of object-oriented data models](#)
- [Difference between relational and object-oriented databases](#)
- [Advantages and disadvantages of object-oriented databases](#)
- [Who should use object-oriented databases?](#)

For the purposes of this article, we'll use the terms “object-oriented database” and “object-oriented database management system” (OODBMS) interchangeably. The differences between the two are technical but minor.

What is an object-oriented database management system?

A brief explanation

An object-oriented database (OOD) is a database system that can work with complex data objects — that is, objects that mirror those used in object-oriented programming languages.

In object-oriented programming, *everything* is an object, and many objects are quite complex, having different properties and methods. An object-oriented database management system works in concert with an object-oriented programming language to facilitate the storage and retrieval of object-oriented data.

You might be thinking, “Wait, I use objects in my programming all the time. And I use a database. So, does that mean the database I use is an OOD?” Probably not, and the reason has to do with one of the main features of OOD: object **data persistence**.

When your program is running, you might have an *object* — for example, an instance of a task. That object has properties like a *name* and *status*. It might also have some methods like `update_task()` or `get_task_history()`. Somewhere early in your program, you initialized that task object, and now you have access to it because it is stored *in memory*.

What happens when your program terminates execution? Your object...is gone. That data object was *transient*, *not persistent*. The next time your program starts up, you'll need to retrieve those data values (like *name* and *status*) from your database all over again in order to initialize a *new* task object instance.

With an OOD, data objects are stored with all of their properties in the database. When your program terminates, the objects continue to persist, stored in the OOD. When your program starts up again, it can retrieve an object with

the properties from the database. The process of storing and retrieving a complex data object with an OOD is transparent to the user of the database.

This is quite different from relational databases (like MySQL or SQLite) but not significantly from document databases (like MongoDB). In relational databases, the developer needs to compose an object from the results of a set of queries, while in [document databases](#), the mapping of the document fields to the class properties should be almost transparent.

OODs have been around for decades. MongoDB [Realm](#) is one of the new and promising pieces of software in that field.

What is an example of an object-oriented database?

MongoDB does offer an OOD called Realm where the query language constructs native objects through the SDK you use. For example, in the JavaScript SDK, an object fetch look something like:

```
const myTask = realm.objectForPrimaryKey("Task", 12345);
```

Where are object-oriented databases used?

OODs are most often used with object-oriented programming languages like Java, Kotlin, C#, Node JS (React), and Swift. Industries that use OODs are typically those built on an object-oriented language and wanting to boost productivity while working with complex data structures.

One good example of an OOD is online IT training provider CBT Nuggets. [CBT Nuggets](#) uses Realm to offer more than 5,000 courses ranging from basic computer skills to complex network management (see this [case study](#)). CBT Nuggets works to ensure subscribers can view content from anywhere, at any time. To meet this promise, classes are delivered through streaming videos that range from 10- to 20-minute “nuggets.” Content is available on both desktop and through a mobile app.



Components of object-oriented data models

The elements of a OODM are:

- **Object:** A real world entity, such as a specific life task in a to-do list — “*take the garbage out*”.
- **Attributes and Methods:** An object has *state* and *behaviors*. An object has properties (which might also be called attributes) like name, status, and create_date. The set of properties taken together represents its *state*. Along with this, an object has behaviors (also known as methods, actions, or functions) that modify or operate on its properties, like update_task() or get_task_history().
- **Class:** The grouping of all objects with the same properties and behaviors form a *class*. In our example above, we talked about task objects. These objects together all belong to the Task class.

```
class task
{
    String name;
    String status;
    Date create_date;

    public void update_task(String status)
    {
        ...
    }
}
```

```
}
}
```

- **Object-Oriented Design Patterns:** Object-oriented data modeling also implies certain principles like *inheritance*, *polymorphism*, *overriding*, and *association*. An object-oriented database system will support these same concepts.


Relational versus object-oriented databases

What is the difference between a relational database and object-oriented?

Relational database management systems (RDBMS) work with tables, with each row in the table representing a record. The columns in a row represent the attributes of an individual record. Associations between records (“A Company has many Employees. An Employee belongs to a Company”) are facilitated with foreign keys in one table referencing IDs in another table. These associations make up the “relational” part of relational databases.

Data values stored in relational databases are atomic and primitive. By primitive, we mean that they are types like characters, text strings, numbers, and hashes. Even though MySQL and SQLite support the JSON (JavaScript Object Notation) data type, that’s not the same as supporting objects in the sense that OODs do.

Contrast this with the OOD, which typically stores and manages objects directly on the database server’s disk. There are no tables, no rows, no columns, no foreign keys. There are only objects.

Associations between objects in an OOD can also be established and persist, which can lead to powerful and fast querying of data across complex relationships. 

Is NoSQL an object-oriented database?

Popular NoSQL databases like MongoDB and AWS DynamoDB are document-oriented databases while others like Cassandra are key-value stores. Document databases, like OODs, don’t work in terms of tables, rows, and columns; but some languages might need an ODM to better work with objects.

Every “record” is seen as a document, which can shrink and grow in terms of the attributes it stores for a given entity. At times, relationships between documents might be the preferred approach.

For example, you can think of a `BlogPost` document that has associations with multiple `Comment` documents and `Like` documents. At other times, relationships can be embedded directly into a document. In this case, you can imagine a `BlogPost` document with a `Comments` attribute which is an array of text strings and usernames, and then another `Likes` attribute which is an array of usernames and timestamps.

Document databases provide flexible structures that scale well horizontally. They can be powerful in storing very complex documents which, on the surface, might seem like they’re the same thing as objects — and many modern programming languages confuse MongoDB further by calling these documents “objects.” However, these documents are not objects in the traditional sense of object-oriented programming as described above.

Document databases are similar but not the same thing as object-oriented databases.

What are the advantages of object-oriented databases?

With all of their complex associations to other objects, and because complex data objects persist in an OOD, the most significant advantage of the OOD over RDBMS is the ability to query across these complex relationships very quickly.

- There are no slow “joins” as in an RDBMS. Instead, you have fast queries with complex data.

- Since the database structure is so close to the programming objects, the code is simpler and lighter.

As another example, we might think back to our task object instance, which cannot be stored as-is in MySQL. It needs first to be decomposed into its attributes to be stored in the table as a row with columns. The reverse process will involve retrieval *and* composition. Not so with object-oriented or document databases. Have an object? Store the whole thing in the database.

What are the disadvantages of object-oriented databases?

An OOD may be a great choice if you're using an object-oriented programming language and need to manage complex data with complex object-to-object associations. Designing and optimizing a database system for these kinds of complexities, however, also has its trade-offs.

For one thing, the relative performance of very simple database operations — the ones you might do for a simple lookup of an attribute from a relational database table — may be sub-optimal.

Additionally, while users of RDBMS can enjoy a standard query language (SQL), users of object-oriented database systems may not have widely adopted standards at their disposal. For the most part, each flavor of OOD is coupled tightly to an object-oriented programming language, and querying syntax is very language-dependent.

Lastly, the OOD user community still feels small in comparison to the exploding ecosystem of web development within the RDBMS space. But the community is fast-growing and likely to make up for lost time.

Who needs object-oriented databases?

If your application is built with an object-oriented language, then there is likely an OOD or document DB that couples well with your language.

MongoDB Atlas and Realm

MongoDB Atlas is a cloud-based database service for deploying a fully managed instance of MongoDB, based on the foundation of open development with a large community. Realm is a mobile Offline-First database (OOD) which can be used alongside MongoDB Atlas Device Sync for a native, bi-directional sync with Atlas. Together with its sync to Atlas, Realm allows users to benefit from an OOD and a document store database, without the overhead of a complex setup or coupling to a specific object-oriented programming language.

Conclusion

We've covered quite a bit of ground in this article. Object-oriented databases can be a fascinating solution to explore and to use, especially if the value they offer aligns well with the language you're using and the business needs you're meeting.

FAQ

What is an example of an object-oriented database?



Where are object-oriented databases used?

Are NoSQL databases object-oriented databases?

What is the difference between a relational database and object-oriented database?

What are the advantages of an object-oriented database?

References

- <https://www.mongodb.com/realm>
- <https://www.mongodb.com/document-databases>
- <https://docs.mongodb.com/realm/>



English

About

Careers

Legal Notices

Security Information

Support

Contact Us

Atlas Status

Social

Investor Relations

Privacy Notices

Trust Center

Customer Portal

Paid Support



GitHub



Stack Overflow



LinkedIn



Youtube



Twitter



Twitch



Facebook

© 2022 MongoDB, Inc.