**Advantages of DDB**

* Management of distributed data with different levels of transparency like network transparency,fragmentation transparency, replication transparency, etc.
* Increase reliability and availability.
* Easier expansion.
* Reflects organizational structure.
* Local autonomy or site autonomy.
* Protection of valuable data.
* Improved performance.
* Economics.
* Modularity.
* Reliable transactions.
* Hardware, Operating System, Network, Fragmentation, DBMS, Replication and Location Independence.
* Continuous operation.
* Distributed Query processing.
* Distributed Transaction management.

**Disadvantages of DDB**

* Increased complexity
* Demand for security
* Difficult in maintaining integrity
* Lack of standards
* More complex design
* Requirement of additional software
* OS with distributed environment support
* Concurrency control

**Object Oriented Database** (**OODB**) provides all the facilities associated with object oriented paradigm. It enables us to create classes, organize objects, structure an inheritance hierarchy and call methods of other classes. Besides these, it also provides the facilities associated with standard [database](http://ecomputernotes.com/fundamental/what-is-a-database/advantages-and-disadvantages-of-dbms) systems. However, object oriented database systems have not yet replaced the RDBMS in commercial business applications.

**Following are the two different approaches for designing an object-oriented database:**

• Designed to store, retrieve and manage objects created by programs written in some object oriented languages (OOL) such as C++ or [java](http://ecomputernotes.com/java/what-is-java/what-is-java-explain-basic-features-of-java-language).

Although a relational database can be used to store and manage objects, it does not understand objects as such. Therefore, a middle layer called object manager or object-oriented layer software is required to translate objects into tuples of a relation .

• Designed to provide object-oriented facilities to users of non object-oriented programming languages (OOPLs) such as C or Pascal.

The user will create classes, objects, inheritance and so on and the database system will store and manage these objects and classes. This second approach, thus, turns non-OOPLs into OOPLs. A translation layer is required to map the objects created by user into objects of the database system.

**Advantages of OODBMS**

* **Enriched modeling capabilities**

The object-oriented data model allows the 'real world' to be modeled more closely. The object, which encapsulates both state and behavior, is a more natural and realistic representation of real-world objects. An object can store all the relationships it has with other objects, including many-to-many relationships, and objects can be formed into complex objects that the traditional data models cannot cope with easily.

* **Extensibility**

OODBMSs allow new [data type](http://ecomputernotes.com/java/data-type-variable-and-array/explain-data-types-in-java)s to be built from existing types. The ability to factor out common properties of several classes and form them into a superclass that can be shared with subclasses can greatly reduce redundancy within system is regarded as one of the main advantages of object orientation. Further, the reusability of classes promotes faster development and easier maintenance of the database and its applications.

* **Capable of handling a large variety of data types**

Unlike traditional databases (such as hierarchical, network or relational), the object oriented database are capable of storing different types of data, for example, pictures, voice video, including text, numbers and so on.

* **Removal of impedance mismatch**

A single language interface between the Data Manipulation Language (DML) and the programming language overcomes the impedance mismatch. This eliminates many of the efficiencies that occur in mapping a declarative language such as SQL to an imperative 'language such as 'C'. Most OODBMSs provide a DML that is computationally complete compared with SQL, the 'standard language of RDBMSs.

* **More expressive query language**

Navigational access from the object is the most common form of data access in an OODBMS. This is in contrast to the associative access of SQL (that is, declarative statements with selection based on one or more predicates). Navigational access is more suitable for handling parts explosion, recursive queries, and so on.

* **Support for schema evolution**

The tight coupling between data and applications in an OODBMS makes schema evolution more feasible.

* **Support for long-duration, transactions**

Current relational DBMSs enforce serializability on concurrent transactions to maintain database consistency. OODBMSs use a different [protocol](http://ecomputernotes.com/computernetworkingnotes/computer-network/protocol) to handle the types of long-duration transaction that are common in many advanced database application.

* **Applicability to advanced database applications**

There are many areas where traditional DBMSs have not been particularly successful, such as, Computer-Aided Design (CAD), Computer-Aided Software Engineering (CASE), Office Information System(OIS), and Multimedia Systems. The enriched modeling capabilities of OODBMSs have made them suitable for these applications.

* **Improved performance**

There have been a number of benchmarks that have suggested OODBMSs provide significant performance improvements over relational DBMSs. The results showed an average 30-fold performance improvement for the OODBMS over the RDBMS.

**Disadvantages of OODBMS**

There are following disadvantages of OODBMSs:

* **Lack of universal data model:** There is no universally agreed data model for an OODBMS, and most models lack a theoretical foundation. This .disadvantage is seen as a significant drawback, and is comparable to pre-relational systems.
* **Lack of experience:** In comparison to RDBMSs the use of OODBMS is still relatively limited. This means that we do not yet have the level of experience that we have with traditional systems. OODBMSs are still very much geared towards the programmer, rather than the naïve end-user. Also there is a resistance to the acceptance of the technology. While the OODBMS is limited to a small niche market, this problem will continue to exist
* **Lack of standards:** There is a general lack of standards of OODBMSs. We have already mentioned that there is not universally agreed data model. Similarly, there is no standard object-oriented query language.
* **Competition:** Perhaps one of the most significant issues that face OODBMS vendors is the competition posed by the RDBMS and the emerging [ORDBMS](http://ecomputernotes.com/database-system/adv-database/object-relational-database-systems) products. These products have an established user base with significant experience available. SQL is an approved standard and the relational data model has a solid theoretical formation and relational products have many supporting tools to help .both end-users and developers. Query optimization compromises encapsulations: Query optimization requires. An understanding of the underlying implementation to access the database efficiently. However, this compromises the concept of incrassation. Locking at object level may impact performance Many OODBMSs use locking as the basis for concurrency control protocol. However, if locking is applied at the object level, locking of an inheritance hierarchy may be problematic, as well as impacting performance.
* **Complexity:** The increased functionality provided by the OODBMS (such as the illusion of a single-level storage model, pointer sizzling, long-duration transactions, version management, and schema evolution--makes the system more complex than that of traditional DBMSs. In complexity leads to products that are more expensive and more difficult to use.
* **Lack of support for views:** Currently, most OODBMSs do not provide a view mechanism, which, as we have seen previously, provides many advantages such as data independence, security, reduced complexity, and customization.
* **Lack of support for security:** Currently, OODBMSs do not provide adequate security mechanisms. The user cannot grant access rights on individual objects or classes.