

ONTOS DB 3.1

An object-oriented DBMS that is integrated in an extended C++ programming environment.

- Refer to chapter 24: Ontos: An Integrated Object System.
- Refer to the handout of "Introduction to ONTOS DB 3.1".

What is ONTOS DB (pp.2–17)

- A multi-user, distributed object database with a C++ class library interface.
- A reliable persistent storage facility for C++ objects.
- Standard database capabilities and special support for objects.
- A set of useful tools for application construction.

Database Schema: ONTOS DB types and C++ classes (pp. 2–19)

The database schema in ONTOS DB is represented by Types.

<i>ONTOS DB</i>	<i>C++</i>
<i>Type</i>	<i>Class</i>
<i>Property</i>	<i>data member</i>
<i>procedure</i>	<i>member function</i>

A Type specifies:

- An instance's state in terms of its properties.
- An instance's dynamic behaviour in terms of procedures.
- A means of constructing new instances.
- Information used in bringing its instances into memory from disk.
- An optional iterator for accessing all of tis instances.

Every instance in the database is an instance of some Type and at the same a corresponding C++ class.

- Type and class can generally be used interchangeably.
- In ONTOS DB, the term of Type refers to the database representation of a C++ class, and class refers to the C++ notion of a class or struct.
- Types differs from C++ classes.
 - Types are denotable. There is no class “class” in C++. In C++, it is impossible to declare a class variable.
 - Denotable types enable an application to create instances of a user–specified type, to select related properties for accessing or procedures for invoking, or even to create new types at run time.
 - A Type may have an extension, the collection of all instances of the type. C++ does not record or maintain this information.

Data members and properties (pp. 2–20)

- In ONTOS DB, data members are represented by OC_Property objects.
- OC_Property object is used transparently by ONTOS DB for mapping between the database representation of the object and its in-memory representation.
- Application directly access data member values.
- Access through the OC_Property object is used when the name of the data member is unknown by the application.

Member functions, free functions and Procedures (pp.2–22)

- Member function and free functions are represented in the database by procedure objects, instances of the type OC_Procedure.
- Access through Procedure objects is usually used when the name of the function is not known by the programmer.
- OC_ is a prefix used to distinguish ONTOS DB class names and free function names.

Entities and Primitives in ONTOS DB (pp. 2–23)

- The term “Entity” is intended to evoke a more abstract and amorphous image than does the term “Object”. Entities include OC_Integers, OC_Strings, OC_Reals, and objects of all kinds.
- Entities do not have a bounded lifetime, or an independent existence. Entities are “discovered” rather than created. Entities may or may not be physically represented in a database.
- In ONTOS DB, an Entity’s state is typically simple a value, and is immutable.
- All entities have a relationship to an object called their type.
- An Entity’s identity differs from an Object’s identity in that it is defined by its immutable state or value.
- An OC_Primitive is used to persistently represent C++ data types such as int, char*, etc.

Referencing in ONTOS DB (pp.2–24)

- Abstract references; Direct references

Name and name–lookup, OC_Lookup() (pp.2–26)

- Directories and object naming
- Absolute and relative names

Creation of new objects (new pp. 2–28)

Deletion of objects (delete vs deleteObject) (pp.2–29)

Distributed database and client–server architecture (pp.2–29)

- Binder and Database Registry
- Server: Primary Server, Secondary Server
- Area: Kernel areas and areas.
- Databases: Physical databases and Logical databases.

Object activation, locks, and deactivation (pp.2–38)