An **object database** is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases are different from relational databases which are table-oriented. Object-relational databases are a hybrid of both approaches.

STRUCTURED DATA TYPES

SQL allows users to define new data types, in addition to the built-in types (e.g., integers). Distinct types stay within the standard relational model, since values of these types must be atomic.

Define Structured Types / Overview of Structured Types / What is Abstract Datatype? / How to define Structured types or Abstract datatypes? / Structured type (Abstract datatype) definition in Oracle with example.

Structured Types

The major advantage of using objects is the ability to define new data types (Abstract Data Types). In ORDBMS, the RDBMS extends the usage of objects that can be defined and stored as part of database. Like a CLASS declaration in C++ language, a new type can be defined in an ORDBMS as follows; (***the reserved words/keywords are given in UPPERCASE hereafter***)

```
CREATE TYPE type_name AS
(Attribute1_name data_type(size),
Attribute2_name data_type(size),
Attribute3_name data_type(size),
…….
AttributeN_name data+_type(size));
```

Here, data_type can be any of the following;

- It can be one of the valid data types like CHAR, VARCHAR, NUMBER, INTEGER, etc. Or
- It can be another User Defined Type.

We call this kind of new User Defined Types as **Structured Types / Abstract Datatypes**.

For example, Structured types can be declared and used in SQL:1999 as follows;

```
CREATE TYPE phone AS
(Country_code NUMBER(4),
STD_Code NUMBER(5),
Phone_Number NUMBER(10))
```

This type can be used in other TYPE definition or TABLE definition as follows;

```
CREATE TABLE contact
(Contact_name VARCHAR(25),
Street VARCHAR(25),
City VARCHAR(25),
Ph PHONE);
```

In this TABLE definition, PHONE is the structured type that we have defined through previous example.

Structured Types in Oracle

Let us see some examples of defining and manipulating Structured types in Oracle.

```
CREATE TYPE Address AS OBJECT
(Street VARCHAR(35),
City VARCHAR(30),
State VARCHAR(30),
Pincode NUMBER(10));
```

Execution of the above statement will create a new ABSTRACT datatype named ADDRESS and store the definition as part of the database.
This new type can be used to define an attribute in any TABLEs or TYPEs as follows;

```
CREATE TABLE Person
(Person_name VARCHAR(25),
Addr ADDRESS,
Phone NUMBER(10));
```

This table Person will consist of 3 columns where the first one and the third one are of regular datatypes VARCHAR, and NUMBER respectively, and the second one is of the abstract type ADDRESS. The table PERSON will look like as follows;

Person_name	Addr	Phone

	Street	City	State	Pincode	

Table 1 – Person table

OODBMS vs ORDBMS: Differences

The fundamental difference is really a philosophy that is carried all the way through: OODBMS try to add DBMS functionality to a programming language, where ORDBMS try to add richer data types to a relational DBMS.

Although the two kinds of object-databases are converging in terms of functionality, this difference in their underlying philosophy (and for most systeIns, their implementation approach) has important consequences in terMS of the issues emphasized in the design of these DBMS and the efficiency with which various features are supported, as the following comparison indicates:

- OODBMS aim to achieve seamless integration with a programming language such as C++, Java, or Smalltalk. Such integration is not an important goal for an ORDBMS. SQL:1999, like SQL-92, allows us to embed SQL commands in a host language, but the interface is very evident to the SQL programer.
- An OODBMS is aimed at applications where an object-centric viewpoint is appropriate; that is, typical user sessions consist of retrieving a few objects and working on theHl for long periods, with related objects (e.g., objects referenced by the original objects) fetched occasionally. Objects may be extremely large and may have to be fetched in pieces; therefore, attention Must be paid to buffering parts of objects. It is expected that most applications can cache the objects they require in memory, once the objects are retrieved from disk. therefore, considerable attention is paid to making references to ill-memory objects efficient. Transactions are likely to be of very long duration and holding locks until the end of a transaction may lead to poor performance; therefore, alternatives to Two-Phase Locking must be used.

An ORDBMS is optimized for applications in which large data collections are the focus, even though objects may have rich structure and be fairly

large. It is expected that applications will retrieve data from disk extensively and optimizing disk access is still the main concern for efficient execution. Transactions are assumed to be relatively short and traditional RDBMS techniques are typically used for concurrency control and recovery.

- The query facilities of OQL are not supported efficiently in most ORDBMS, whereas the query facilities are the centerpiece of an DBMS. To scene extent, this situation is the result of different concentrations of effort in the clevelopment of these systems. '1'0 a significant extent it is also a consequence of the systems' being optimized. for very different kinds of applications