CHAPTER 4

Document Oriented Database

Unit IV

Syllabus

Need of object-oriented database, Impedance matching problem between OO languages and Relational database, Case study db4O, Need of Document Oriented database, difference between Document Oriented Database and Traditional database. Types of encoding XML, JSON, BSON, Representation XML, Json Objects. Case study on doc oriented based such a Mariadb

4.1 Need of Object-Oriented Databases

Q. What is the need of object oriented database? MU - May 19 . 5 Marks

- Object oriented database systems are proposed as an alternative to relational systems and other available systems. It is aimed at application domains where complex objects play a major role.
- The term object-oriented-abbreviated by OO or O-O has its origins in OO programming languages, or OOPL.
- The OO approach is heavily influenced by OOPL (object-oriented programming languages) and this is used to give DBMS functionality to a programming languages.
- If we extend existing object oriented programming languages with features of relational model such as transaction, recovery, concurrency, atomicity etc. the resultant model is called as **object oriented database model**.
- The basic building blocks of object model is

(a) Object (b) Literals

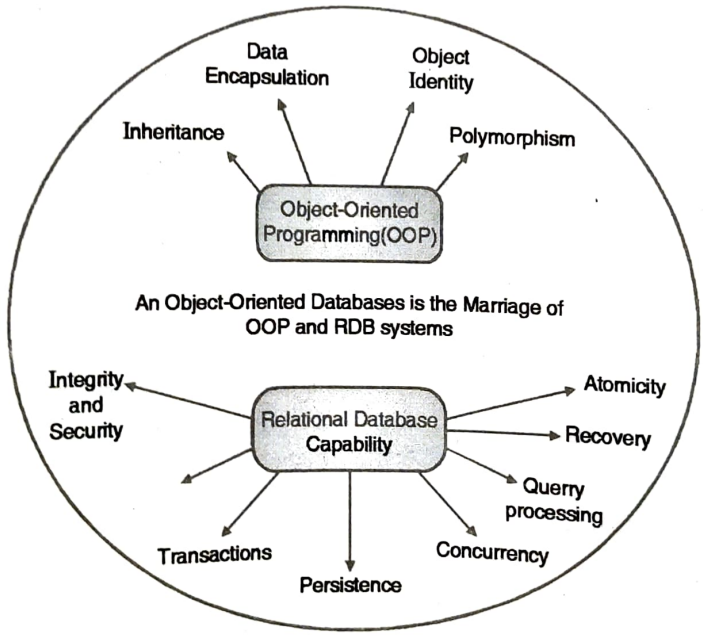


Fig. 4.1.1 : Object oriented database

4.1.1 Features of OODB (Object Oriented Databases)

1. Persistency	2. Identity
3. Complexity	4. Encapsulation
5. Relationship	6. Inheritance

1. Persistency

- OODBMS extends the capacity of OOPL that can create persistent object i.e. object remains in memory even after program execution terminates.
- This feature automatically introduces the concept of recovery and concurrency of databases.

2. Identity

- Generally in DBMS each entity is uniquely identified by a value of primary key, which is specified by user.
- This is value dependent identification.
- The problem of value dependent identification is removed in OODBMS by introducing concept of system generated unique value called as Object identity.
- Every object in OODBMS will be having unique object identity.

3. Complexity

- An object in OODBMS can have complex internal structure with multiple level of complexity.
- State of one object may contain other objects.

4. Encapsulation

- Encapsulation is one of the important characteristics of OO system.
- The idea of encapsulation in programming languages comes from abstract data types.
- This is related to concept of data hiding concepts in OO languages.

5. Relationship

- We maintain relationship between two objects using concept of inverse reference.
- One object maintains OID of other and vice versa.

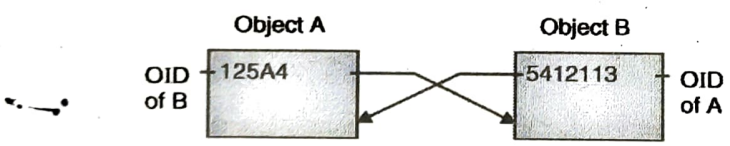


Fig. 4.1.2 : Object relationship

6. Inheritance

- Creating new type from existing type such that new type inherits all characteristics of existing type.
- The parent type is called as supertype while newly created type is called as subtype.

Fig. 4.1.3

4.2 Impendence Matching Problem Between OO languages and Relational Database

Q. Write short note on Role Based Access Control for multilevel security. MU - Dec. 18, 5 Marks

- Authorization is finding out if the person, once identified, is permitted to have the resource.
- Authorization explains that what you can do and is handled through the DBMS unless external security procedures are available.
- This is usually determined by finding out if that person is a part of a particular group, if that person has paid admission, or has a particular level of security clearance.
- Authorization is equivalent to checking the guest list at an exclusive party, or checking for your ticket when you go to the opera.
- Database management system allows DBA to give different access rights to the users as per their requirements.
- In SQL Authorization can be done by using read, insert, update or delete privileges.

4.2.1 Object Oriented Languages/Object-Oriented DBMS (OODBMS)

1. Introduction

(a) To represent complex objects in programming has led to the development of object-oriented (OO) systems.

(b) It uses ODMG standards.

2. Features

(a) Abstract data types (ADTs) in which the internal data structure is hidden and the external operations can be applied on the object (**encapsulation** of object).

(b) Inheritance The process in which one object inherits the properties of a previously defined object is called **inheritance**. Inheritance aids in the reuse of existing definitions for creating new objects.

(c) Polymorphism allows the same name, operator or symbol to have different implementations.

(d) Objects Identity (OID) OO databases provide unique object identifiers (OIDs) so that the objects can be uniquely identified.

(e) The data in object-oriented database management systems (OODBMS) is managed through two sets of relations, one describing the interrelations of data items and another describing the abstract relationships (inheritance).

3. Languages Supported

(a) SMALLTALK

(b) C++

(c) Java

4. Disadvantages

(a) The lack of a defining standard.

(b) Poor performance.

(c) Query optimization for OODBM is highly complex.

(d) OODBMS is also suffer from problems of scalability and are unable to support large-scale systems. Example of OODBMS are O2 (now called Ardent) developed by Ardent Software.

4.2.2 Relational DBMS (RDBMS) / Traditional Database

1. Introduction

(a) The relational model was introduced by Dr. E. F. Codd in 1970 and has evolved since then, through implementations by IBM and others.

(b) Standard for relational databases is published by ANSI (the American National Standard Institute) as SQL (ANSI 1986) or SQL1, called SQL-86.

(c) A revised standard is called SQL2, also referred to as SQL-92.

(d) The SQL standard enables users to easily migrate their database applications between different database systems.

(e) In addition, users can access data stored in two or more RDBMSs without changing the database sub-language (SQL).

2. Features

A relational database is composed of many relations in the form of two-dimensional tables of rows and columns containing related tuples (horizontal rows) known as logical view.

3. Examples

(a) Oracle, developed by Oracle Corporation

(b) Microsoft Access developed by Microsoft

4. Disadvantages

Inability to handle application areas like spatial databases (e.g. CAD), applications involving images, special types databases (e.g. complex numbers, arrays, etc.) and other applications that involve complex interrelationships of data.

4.2.3 Object-Relational DBMS (ORDBMS)

1. Introduction

(a) ORDBMS was designed to achieve the benefits of both the relational and the object models.

(b) ORDBMS is a data model that attempts to incorporate OO features into RDBMS.

2. Features

(a) All database information is stored in tables; these tabular entries may have richer data structure, termed abstract data types (ADTs).

(b) An ORDBMS supports SQL3 which is still in the development stages.

(c) The ORDBMS has the relational model in it because the data is stored in the form of tables having rows and columns.

(d) SQL is used as the query language and the result of a query is also table or tuples (rows).

(e) ORDBMS allow users define data types (UDT), functions and operators.

4.2.4 Differences Between ODBMS, RDBMS and ORDBMS

Sr. No.	OODBMS	RDBMS	ORDBMS
1.	**Maturity :** Database concept is continuously evolving field, databases which are very old having lot many concepts supported to it called as mature databases. At the same time, development of databases is in progress such databases are called as Immature databases.		
	Relatively old so mature	Very old hence, Very mature	Still Developing So, Immature
2.	**Ease of use to End User** This parameter says system is simple for handling to even any person without much knowledge of databases.		
	Easy and enables end user to access.	OK for developers Easy SQL access for end user.	Easy without ORDBMS extensions.
3.	**Supported Languages**		
	Object oriented query language(OQL)	Structured Query language (SQL)	SQL with Object oriented features
4.	**Standard Used**		
	ODMG-2.0	SQL2	SQL3 (Development is still in progress).
5.	**Support for Object oriented concepts**		
	Yes As this is object oriented database. Hence, full support to all Objects oriented features.	**No** Does not support; It is difficult to convert objects in program to the database.	**Yes** Limited Support Still new types having support to some OO features.
6.	**Support for Complex Relationships**		
	Supports a wide range of data types and data with complex relationships.	No support For abstract data types (ADT).	Supports Abstract data types (ADT) and also complex relationships.
7.	**Performance**		
	Poor	Very Good	Excellent
8.	**Advantages**		
	Reusability of code Less coding	SQL dependencies Simple optimization	Support complex applications
9.	**Disadvantages**		
	Low performance because of complex query optimization, No support large-scale systems.	Cannot handle complex applications.	Low performance for web applications.

4.3 Need of Document Oriented Database

Q. Explain Discretionary Access Control, also explain access control list and access control entry w.r.t. the same. MU - Dec. 18. 5 Marks

Q. Write short note document oriented database. MU – May 19. 10 Marks

1. Overview

- Document based database is a not a RDBMS (Relational Database Management System).
- Document based database is specially designed for large amount of data stored in distributed environment.
- Document based database using NoSQL concept
- The important feature of NoSQL is, it is not bounded by table schema restrictions like RDBMS. It gives options to store some data even if there is no such column is present in table.
- NoSQL generally avoids join operations.

2. Need

- In real time, data requirements are changed a lot. Data is easily available with Facebook, Google+, Twitter and others.
- The data that includes user information, social graphs, geographic location data and other user-generated content.
- To make use of such abundant resources and data, it is necessary to work with a technology which can operate such data.
- SQL databases are not ideally designed to operate such data.
- NoSQL databases specially designed for operating huge amount of data.

3. Advantages

- Good resource scalability
- Lower operational cost
- Supports semi-structure data
- No static schema
- Supports distributed computing
- Faster data processing
- No complicated relationships
- Relatively simple data models

4. Disadvantages

- Not a defined standard
- Limited query capabilities

4.4 Difference Between Document Oriented Database and Traditional Database

Q. Give difference between document oriented database and traditional database. MU - May 19. 10 Marks

SQL databases are Relational Databases (RDBMS); whereas document based database are non-relational database.

1. Data Storage

- SQL databases stores data in a table whereas document based databases stores data as document based, key-value pairs, graph databases or wide-column stores.

- SQL data is stored in form of tables with some rows.
- NoSQL data is stored as collection of key-value pair or documents or graph-based data with no standard schema definitions.

2. Database Schema

SQL databases have predefined schema which cannot be change very frequently, whereas NoSQL databases have dynamic schema which can be change any time for unstructured data.

3. Complex Queries

- SQL databases provides standard platform for running complex query.
- NoSQL does not provide any standard environment for running complex queries.
- NoSQL are not as powerful as SQL query language.

Table 4.4.1 Comparison

Sr. No.	SQL	Document based Database
1.	Full form is Structured Query Language	Document base database is also a Non-relational database
2.	SQL is a declarative query language	This is Not a declarative query language
3.	SQL databases works on ACID properties, – Atomicity – Consistency – Isolation – Durability	NoSQL database follows the Brewers CAP theorem, – Consistency – Availability – Partition Tolerance
4.	Structured and organized data	Unstructured and un-replicable data
5.	Relational Database is table based	Key-Value pair storage, Column Store, Document Store, Graph databases
6.	Data and its relationships are stored in separate tables	No pre-defined schema
7.	Tight consistency	Eventual consistency rather than ACID property
8.	**Examples** – MySQL – Oracle – MS SQL – PostgreSQL – SQLite – DB2	**Examples** – MongoDB – Big Table – Neo4j – Couch DB – Cassandra – HBase

4.5 Types of Encoding XML

Q. Explain Mandatory Access, also explain access control list and access control entry w.r.t. the same.

MU – Dec. 18, 5 Marks

1. Introduction

- XML is a markup language is very much like HTML but XML was designed to carry (transfer) data and not to display data.
- XML stands for Extensible Markup Language which gives a mechanism to define structures of a document which is to be transferred over internet.
- The XML defines a standard way of adding element to documents. Hence, XML is used for structured documentation.
- Unlike HTML, XML tags are not predefined one can define their own tags in XML.

2. Goals of XML

- XML should be directly used over the Internet. Users must be able to view XML documents easily as like HTML documents. This may be possible only when XML browsers are as robust and widely available as HTML browsers.
- XML should support a wide variety of applications.
- It should be Easy to write programs and process various XML documents.
- The minimum number of optional features in XML as it causes more confusion in programmers mind.
- XML documents should be logically clear.
- The design of XML shall be formal and concise and can be prepared very fast
- XML documents shall be easy to create

4.5.1 Well Formed Document

- XML Documents which satisfy below given rules are called as well formed XML documents,

(a) XML document must start with an XML declaration to indicate the version number of XML being used all other relevant attributes.

```
<? xml version = "1.0" standalone = "Yes"?>
```

(b) A well-formed XML document should be syntactically correct.

(c) Conditions for Tree Model / syntactically correct XML document

- There should be only one root element.
- Every element is included in an identical pair of start and end tags within the start and end tags of the parent element.
- Above conditions will ensure that the nested elements specify *a well-formed tree structure*.

(d) **User defined Tags**

An element within XML document can have any tag name as specified by user. There is no predefined set of tag names that a document knows.

4.5.2 Valid XML Documents

XML Documents which satisfy below given conditions are called as valid XML documents,

(a) Conditions

- The document must be well formed
- Element used in the start and end tag pairs must follow the specified structure.
- This structure is specified in a separate XML DTD (Document Type Definition) file or XML schema file.

(b) DTD Syntax

- Start with a name is given to the root tag of the document.
- Then the elements and their nested structures are specified in top down fashion as below example.
- More details of this topic are given in section DTD set 5 of chapter

```
<! DOCTYPE company [
<! ELEMENT Employee (ID, Name, DeptNo proj) Company
<! ELEMENT ID (#PCDATA)
<! ELEMENT Name (#PCDATA)
<! ELEMENT DeptNo (#PCDATA)
<! ELEMENT project (Name, Number)
<! ELEMENT Name (#PCDATA)
<! ELEMENT Number (#PCDATA)
] >
```

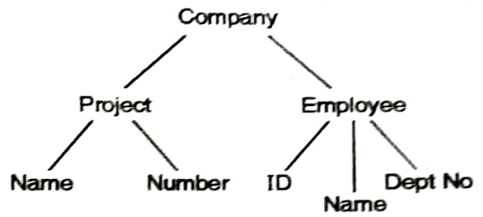


Fig. 4.5.1 : An XML DTD file called company

4.6 Representation XML Objects

1 Introduction

In the XML document the basic object is XML and it can be represented as hierarchical data model or tree data model.

2 Structuring concepts used to construct an XML document

(a) Element

- An element is a group of tags data values that can contain character data, child element or a mixture of both.
- Element can be of two type Simple and Complex.
- The elements are constructed from other elements by nesting them is called as complex element.
- The elements contain data values are called as Simple elements.

(b) Attributes

- Additional information that describes elements.

(c) There are some additional concepts used in XML, such as entities, identifiers, and references.

3 A major difference between XML and HTML

- **Tag names :** In HTML tag names are used to describe how text is to be displayed and in XML tag names are defined to describe the meaning of the data elements in the document.
- **Processing :** With help of user defined tags it is possible to process the data elements in the XML document automatically using computer applications.

4 Tree Representation

XML Model is made of two elements :

(i) Complex elements are shown with help of internal nodes.

(ii) Simple elements are shown by Leaf nodes.

Hence, XML Model is called as Tree Model or hierarchical model.

5. Example

(a) Tree Representation

- **Simple Elements :** <Price>, <amount>
- **Complex elements :** <Drink>, <Snack> etc.

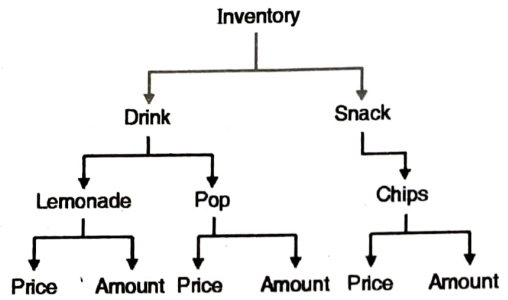


Fig. 4.6.1 : Tree model for above code

(b) Textual Representation

- Whenever value of the STANDALONE attribute in an XML document is set to "YES", such XML document is known as schemaless XML documents.
- E.g.

```
<inventory>
  <drink>
   <lemonade>
     <price>$2.50</price>
     <amount>20</amount>
   </lemonade>
   <pop>
     <price>$1.50</price>
     <amount>10</amount>
   </pop>
```

```
    </drink>
    <snack>
     <chips>
       <price>$4.50</price>
       <amount>60</amount>
     </chips>
    </snack>
</inventory>
```

4.6.1 Types of XML documents

(a) Data-centric XML documents

- The document contains many small data items that follow a specific structure and hence it may be extracted from a structured database are called as data centric XML document.
- In order to exchange and display data over internet the document is formatted as XML documents.

(b) Document-centric XML documents

- The document contains large amounts of text, such as news articles or books are referred as document centric XML documents.
- There are only few structured data elements in these documents. Sometimes there are no data elements in such documents.

(c) Hybrid XML documents

- If both types of data are used simultaneously in document then it is referred as hybrid XML document.
- In such documents some parts that contain structured data and other parts that are predominantly unstructured.

4.6.2 XML – Structured Data or Semi Structured Data

- Data centric XML documents sometimes considered as semi structured data or sometime considered as structured data.
- Document is considered as structured data if an XML document is written as per predefined XML schema or DTD.
- Document is considered as semi structure if an XML allows documents that do not conform to any particular schema.

4.6.3 XML Document Type Definition (DTD)

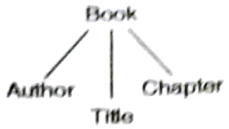
1. Introduction

- The way by which we describe a valid syntax of XML document by listing all the elements occur in the document which elements can occur in combination? How elements can be nested ? What attributes are available for each element type ? and so on.
- DTD describes the rules for analysing an XML document.
- A DTD ensures that the contents of an XML document conform to expected rules that document should follow.

Example

```
<! DOCTYPE BOOK [
<! ELEMENT Book (Author, Title, Chapter>)
<! Element Author (# CDATA)>
```

```
<! Element Title (# PCDATA)>
<! Element Chapter (# PCDATA)>
<! ATTLIST Chapter # REQUIRED>]>
```



(i) <! DOCTYPE Book [

The DTD starts with the above line. It indicates that this DTD corresponds to an XML document that contains Book as the main element or root element.

(ii) <! Element Book (Author, Title, Chapter)>

This line indicates that the first element in our XML document would be Book. Additionally this line also signifies that the Book element is the parent of sub-element namely, author, title and chapter. The + sign after chapter indicates that it can contain one or more chapter elements.

(iii) <! Element Author (# CDATA)>

This tag specifies that element Author is type of CDATA character data.

(iv) <! ELEMENT Title (# PCDATA)> & <! ELEMENT Chapter (# PCDATA)>

This tag specifies that element Title and Chapter are of type PCDATA - passed character data.

(v) <! ATTLIST chapter # REQUIRED>

This line specifies that id is an attribute of the element chapter. The #REQUIRED declarative, specifies that id is must.

2. Types of DTD as per location of DTD

A DTD is either contained in <!DOCTYPE> tag of same file as XML document or contained in an external file referenced from a <!DOCTYPE> tag or both.

(a) Internal DTD : DTD Embedded in XML Document

(b) External DTD : DTD as a separate external file

3. Types of DTD Declarations

(a) Element Type Declaration (b)Attribute list declaration

(c) Entity Declaration (d)Notation Declaration

(a) Element Type Declarations

- An element declaration defines one of the kinds of elements you can use, that is, one of the tag types.
- Example
- Suppose a <speech> element can contain any mixture of regular text, and text tagged with the elements <loud> and <soft>

```
<!ELEMENT speech ((#PCDATA|loud|soft)*)>
<!ELEMENT loud (#PCDATA)>
<!ELEMENT soft (#PCDATA)>
```

- Notations

Let, A is element in XML document.

- **A *** (optional multi valued (repeating) element)
- Element name **A*** means that the element A can be repeated zero or more times in the XML document.
- **A +** (required multi valued (repeating) element)
- Element name **A+** means that the element A can be repeated one or more times in the XML document.
- **A ?** (optional single-valued (non repeating) element)
- Element name **A?** Means that the element A can be repeated zero or one times in the XML document.
- **A** (Required single-valued (no repeating) element)
- An element appearing without any of the preceding element must appear exactly once in the XML document.
- **A1 | A2** (Bar Symbol)
- Only one out of *A1* and *A2* can appear in the XML document.
- **(A)** ***Parentheses*** can be nested when specifying elements

(b) Attribute List Declaration

- Attribute list declaration identifies which elements have attributes they may have attributes, values and optional attributes.
- Parts
  - (i) Name
  - (ii) Type
  - (iii) An optional default value
- Example

```
<!ATTLIST vehicle_kind (car|truck|bike) #REQUIRED>
<!ATTLIST vehicle_kind (car|truck|bike) "car">
```

In above example first indicates vehicle_kind is not null or required attribute while second indicate vehicle_kind have 'car' as default value.

(c) Entity Declaration and Notation Declaration

- Element declaration associates a name with same fragment of content, such as a piece of regular text, a piece of DTD or a reference to an external file containing text or binary data.

  E.g. <!ENTITY DH "Xavier's Institute">
- The process of unparsed entities is responsibility of application. Some information about the entities internal format must be declared after the identifier that indicates the entity location.

  E.g. <!ENTITY Logo "XavierLogo.jpg" NDATA JPEGFormat>

4. Advantages of DTD

(a) A DTD allow an XML parser to validate an XML document against its definition. This allows the online validation of an XML document before it is processed.

(b) A DTD can extend the capabilities of an XML document by allowing further processing possibilities.

(c) A DTD serves as an automotive documentation on XML document.

(d) A DTD can be used to validate data against the business rules. These business rules would be defined in the DTD and the incoming data can be validating against those.

5. Disadvantages of DTD

(a) It is written in a different (non XML) syntax.

(b) It has no support for Namespaces.

(c) It only offers extremely limited data typing.

4.7 JSON

1. Overview

- JSON stands for JavaScript Object Notation.
- JSON is basically used for data transmission in the web applications.
- JSON is the extension of the JavaScript language.
- JSON can be used for storing and exchanging data.
- JSON can be used as an alternative to XML.
- JSON is a lightweight, text based data-interchange format.Text can be read and used as a data format by any programming language.
- JSON is language independent and can be used with modern programming languages. All popular programming languages have parsing code and support for generating JSON data.
- JSON is self-describing and easy to understand.
- JSON has file name extension .json.
- JSON supports name-pair values, arrays, list, vector, sequences etc.

2. JSON code

```
{"movies" : [
{"Name" :"Tom and Jerry", "year" : "2000"},
{"Name" :"John Carter", "year" : "2014"},
]}
```

3. XML Code for the same program

```
<movies>
  <movie>
    <Name>Tom and Jerry</Name>
    <year>2000</year>
  </movie>
  <movie>
    <Name>John Carter</Name>
    <year>2014</year>
  </movie>
</movies>
```

From the above two codes it is clear that JSON requires less efforts to write the same code as compared to XML.