Chapter Two

Review of the Existing Works
2.1 Overview of Object-Oriented Databases

This Section introduces the overview of Object-Oriented databases. Section 2.2 presents the brief review of Constraint Models in Object-Oriented data while review of Constraint Models in UML has discussed in Section 2.3. Section 2.4 has introduced XML and Constraints. The issue of interoperability with constraints in XML has been presented in Section 2.5.

Object-oriented databases are becoming increasingly popular because of their capabilities to provide rich semantic constructs to model real world entities and their relations [7], [17], [27], [38], [40], [48]. In the process, the notion of object, class, inheritance, relationships among classes and objects has been thoroughly treated in object oriented database systems. An overview on the successes of Object Oriented technology achieved so far has been introduced in [7]. It [7] also presents the object oriented database (OODB) model and the weaknesses of OODB that have to be resolved by the current Object-Oriented database community before object-oriented database technology can become as widespread as relational databases. An integrated software development or CASE environment is a collection of tools supporting various phases of a production process. It has two main features. The first feature is the possibility to handle inter-document consistency and the second feature is multi-user support. These two features of an environment demand complex structured, persistent data and the use of a sophisticated database management system as a key architectural component of a CASE environment. It has been argued in [27] that fully object-oriented database management systems are the most appropriate ones to build an integrated CASE environment on the top of a database system. This paper [27] has illustrated the concept of an integrated CASE environment and the scheme construction in terms of a class hierarchy of an object-oriented data definition and manipulation language. It [27] has also explained how OODBMS have to be extended in order to fully meet the requirements of CASE environments.

In an object-oriented database system, data are organized in a complex structure built using different constructs (classes, objects, attributes, links etc.). Therefore, to implement a multilevel security policy for Object-Oriented Databases (OODBs), a major problem is to
determine what constructs of the object-oriented model should be associated with a security level and to define semantics for each assignment of a security level to an object-oriented construct. Moreover, the inference problems may occur due to the integrity constraints inherent in the object-oriented paradigm while assigning the security levels. To cope with this problem, a set of general rules for designing multilevel object oriented databases during implementation of multilevel security policy for OODB has been provided in [17].

In the context of modeling real world entities, design, specification and implementation of “Object Oriented Model” of Database systems has gaining great attention [18], [70], [46], [56]. Object Oriented data model is a logical organization of real world objects or entities, constraints on them and relationship among these objects. A core object-oriented data model consists of the following basic components, namely, object and object identifier, attributes, methods, class, class hierarchy and inheritance. Every object has a state, characterized by the set of values for the attributes of the object and a behavior, defined by the set of methods, which manipulate the state of the object. The state and behavior encapsulated in an object while objects with the same properties and behaviors are categorized into classes.

Inheritance is deriving a new class (subclass) from one or more existing classes (super classes). The subclass inherits all attributes and methods of existing classes and may have additional attributes and methods. Exact semantics of inheritance is, however, dependent on an object oriented language and thus, not universal. A class “Y” is said to be a subclass of class “X” (equivalently, class “X” is said to be a super class of class “Y”) if and only if every object of class “Y” is necessarily an object of class “X” (that is, Y ISA X). Objects of class “Y” then inherits the instance variables and methods of class “X”. As a consequence, we can always use a “Y” object wherever an “X” object is expected (e.g., as an argument to various methods), which helps us to take the advantage of reengineering. The subclass “Y” can override the implementation of an inherited method or instance variable by providing an alternative definition or implementation of the base class “X”.

A prototype implementation of an Object Oriented Database (OODB) for VLSI designs has been developed in [70]. Computer aided design of VLSI circuits deal with complex design
objects. The design becomes more complex as the number of modules increases. In view of the modular nature of VLSI design, a database for such an application should also support storage and retrieval of the circuits as modules for objects. Some salient features of an object-oriented database management system, ODS, based on the VLSI design model has been presented in [70]. In addition, a specialized query language VDDL, which serves as a VLSI application layer over ODS has been introduced in [70]. Jasmine, a full-fledged implementation of OODB, has been reported in [46]. New applications such as engineering tasks require complex object modeling, integration of database and programming facilities and extensibility. This paper [46] has a number of contributions. First, it has focused on the impact of the design of its object-oriented model and language on database implementation technology and described what part of traditional relation database technology has to be extended to handle object-oriented features such as object, identifiers, complex objects, class hierarchies and methods. Secondly, it describes nontrivial applications of Jasmine [46] in detail and discusses the validity of object-oriented databases. It [46] also focuses on a constraint management facility, which can be implemented by taking advantage of the extensibility of Jasmine. Finally, a view facility for schema integration needed by engineering applications in distributed environments has been introduced.

In [56], an advanced object modeling environment, ADOME has been reported. In order to achieve a higher level of intelligence behavior, interaction between and integration of data management and knowledge management technologies is necessary and crucial for next generation information system (NGIS) applications such as Organizational Information Systems (OISs), Decision Support Systems (DSSs), Computer Supported Co-operative Work (CSCW) and spatial/temporal information systems. A detail approach to integrate data and knowledge management capabilities for NGIS applications has been described in [56]. It [56] also provides a case study of applying ADOME to an organizational information system (OIS) application, illustrating how ADOME’s facilities can be utilized to solve address specific problems pertinent to OIS applications. A new data model, called CAR, has been defined in [18], which extends the basic core of current object-oriented data model to provide features such as the inverse of the functions represented by attributes, the union, the intersection, the complement of classes, the possibility of using non binary relations and the
possibility of expressing cardinality constraints on attributes and relations. In [18], a technique has presented both for checking the consistency of class definitions and for computing the logical consequences of the knowledge represented in the schema. Finally, in [18], the inherent complexity of reasoning in CAR has been investigated and the complexity of inferencing technique has been studied on various assumptions of the schema.

2.2 Review of Constraint Models in OODB

The crucial requirement for integrity preservation in database systems is a long recognized fact, which is enforced by integrity rules or constraints. Constraints are restrictions on properties and relations of database objects that ensure the integrity of data according to both the system and the user. A complete treatise on constraints is still evolving in the context of Object Oriented Data Model [3], [5], [6], [8], [14], [23], [24], [28], [29], [42], [47], [51], [73], [106]. In relational databases, Date [23] identified four categories of integrity rules, namely domain rules, attributes rules, relation rules and database rules. In [47], integrity rules have also been categorized in four groups – domain rules, attributes rules, class rules and database rules in the context of Object-oriented databases. It has been argued in [47], the domain rules and attribute rules are represented and maintained in an OODB by the class hierarchy automatically by the virtue of object-orientation. Therefore, only class rules and database rules need to be specified and maintained.

2.2.1 Formal/ Semi-formal Constraint Models

This section introduces an overview of formal/semi-formal constraint models in Object-Oriented Data. In [5], a solution to the consistency problem in complex object database environment has been proposed. This paper [5] provides a theoretical framework including two alternative formalisms, OLCP (Object Language with Complements and Path relations) and OLCD (Object Language with Complements allowing Descriptive cycles), which is able to express a relevant set of state integrity constraints in a declarative style. It [5] has also presented two specialized reasoners, based on the tableaux calculus, which is able to check the consistency of complex objects database schema expressed by OLCP and OLCD. In [5],
OLCP and OLCD has not been proposed as new data models, but the structural part of the different existing OODB models could be mapped, and a class of integrity constraints could be expressed in a declarative style. The proposed formalisms of [5] share a common kernel, which provided a general value-based and identity-based type system, disallowing cyclic descriptions. A notion of path existence constraint definitions (in short, PED) as a method to treat formally the existence dependencies, especially, among object reference paths has been presented in [73]. It has been argued in [73], by introducing the notion of selector variables an arbitrary object could be specified as a unit of the path existence constraints. In this paper [73], several inference rules for PEDs has been shown, and practical application examples have been described in order to show the effectiveness of the notion of PEDs. When two or more databases are combined into a global one, integrity may be violated even when each database is consistent with its own local integrity constraints. The contribution of integrity constraints to data integration is usually confined to query reformulation and query optimization problems. A theoretically sound approach, called simplification of integrity constraints, has been developed in [14] for relational and deductive databases to overcome the consistency problem during data integration. It has been proposed in [14] that this new tool, simplification of integrity constraints, can be used to manage a number of different configurations of database schemata and integrity constraints at the sources and at the mediator.

[47] has developed a constraint compilation scheme that accepts declarative global specification of constraints, including relational integrity, referential integrity and uniqueness requirements, and generates an efficient representation that permits localized processing. It [47] demonstrates the feasibility of this approach by designing a constraint pre-processor for O++, the programming language interface to the Ode object-oriented database. An important activity in the design of a particular database application consists of identifying the integrity constraints that must hold on the database, which is used to detect and evaluate inconsistencies. It has been argued in [6] that data quality can be improved by imposing constraints on data entered into the database and these constraints must be identified and recorded at the database design level. This paper [6] has examined the issue of data integrity in geographic information systems, focusing on the relationship that exists among the natural
spatial information, spatial relationships, and spatial integrity constraints. In [6], the importance of identifying such integrity constraints at the conceptual level has been demonstrated, and the use of OMT-G, an extension of the OMT model for geographic applications, has been proposed for the specification of integrity constraints in spatial databases. It [6] also demonstrates an example to show the use of OMT-G in geographic data modeling. A full implementation of the integrity constraints has also presented in [6] to ensure consistency through data entry and update.

Checking finite satisfiability of database constraints is a fundamental problem in database design. This problem is generally addressed by using theorem provers that, being developed for first order logic formulas based on semi-decidable procedures. But, theorem provers are quite inefficient in dealing with comparison operators such as the equality, even in simple cases. To cope with this problem, a decidable, sound, and complete method for checking finite satisfiability of a specific class of integrity constraints for object-oriented databases, including the equality constraints, has been presented in [29]. This method [29] is based on a graph-theoretic approach. It has been argued in [29] that this method has some exponential complexity in the worst case, which rarely occurs in practice.

In [3], constraints are handled by means of exceptions in Object-Oriented databases. Inheritance of constraints in object-oriented databases has been treated in a completely mandatory way in [3], providing no way for exceptions. In this paper [3], an object-oriented constraints representation scheme has been presented along with a methodology for modeling constraints exceptions. Finally, a constraints checking algorithm for object-oriented databases has been introduced in [3]. [42] introduces an Object-Oriented Relationship Data Model which can provide proper facilities to represent and manipulate all of relationships modeled from the real world. In this Object-Oriented Relationship Model [42], the relationship has been expressed as relationship object, which provides an abstraction mechanism for the association as a conceptual construct and makes it possible to capture the semantics of the relationship more clearly such as constraints, generalization abstraction and dynamic aspects. It [42] also introduces several implementation techniques for implementing Object-Oriented Relationship Model.
A model of integrity for object-oriented database management systems based on first order predicate calculus has been developed in [92]. It has been proposed in [28] that an integrity constraint can be classified as implicit, explicit or inherent. An implicit constraint can be specified and represented in the database scheme of a model similar to key and referential integrity in the relational model. An explicit constraint is one, which the user must specify explicitly. An inherent constraint is one, which is part of the data model and does not required to be specified by the user. [92] deals with explicit constraints that restricts the domain of an object-oriented database so the data is more likely to be valid. It [92] introduces an integrity constraint model, which enforces value-based functional dependencies both within an object and across objects of different classes. This integrity model [92] is layered over the data model and does not access the internal structure of the data model.

A modeling technique called subtyping by constraints is proposed in [51]. It [51] relies on the principle of making explicit all integrity constraints involved in the design process. [51] argues that making constraints explicit produces semantically more desirable hierarchies with enhanced substitutability. An automatic adaptation algorithm has been provided in [51] to broaden the amount of substitutable update methods. It [51] also introduces a new storage optimization technique based on functional integrity constraints to avoid potential storage penalty caused by making constraints explicit.

### 2.2.2 Frameworks for Constraint Model

Frameworks for constraint Models has been discussed in this section. In recent years, modeling integrity constraints in object-oriented databases has become an active research topic [8], [24], [106]. A framework for constraint management in object-oriented databases has been proposed in [106]. The objective of [106] is to associate constraints to objects and these constraints incorporated the semantics associated with inheritance over generalization hierarchies and referenced to simple objects, which are constituents of complex objects. It has been proposed in [106] that these constraints can be user-defined and dispersed over object types as methods associated with object types. Because constraints are tightly coupled
with the object, it is simple to maintain object consistency within an object when the object is
updated. The update semantics for update propagation of object instances has been embedded
in the constraints in [106]. But, it is hard to maintain overall database consistency if effects of
the update of an object are propagated and may lead to additional inconsistencies in other
objects. To overcome this difficulty, there are three well-known approaches: naïve approach,
limited approach, and expert approach. In a naïve approach, all constraints are defined in one
place (e.g., a root node or the meta class) and all constraints are considered for enforcement.
The limited approach allows constraints to be defined and associated with an object. Only
intra-object constraints associated with an object or inherited from a super type object are
considered when an update or a query is presented to the object. Here, no propagation of
update effects has been considered in object oriented databases and so overall database
consistency could not be ensured. An expert approach allows constraints to be defined and
associated with an object and the update effects are propagated thereby ensuring database
consistency. In [106], an expert approach for constraint management in object-oriented
database has been proposed.

In [24], a framework for engineering information exchange has proposed for an information-
driven computer integrated manufacturing (CIM) system, using the ISO STEP EXPRESS
specification language and an active object-oriented database. This framework [24] allows
both the engineering data and its integrity constraints specified using EXPRESS to be stored
in the active object-oriented database. In [24], using triggers for user-defined integrity
constraints, the active object-oriented database enables application programs to share the
engineering data without any concern about the integrity constraints, thus integrity checking
has performed by the database management system, not by individual application programs.
It [24] integrates various existing computer aided engineering tools in a heterogeneous CIM
environment. The translation framework along with appropriate translation examples and a
prototype design system based on the framework has been discussed in [24]. [8] reports the
development of the first constraint object–oriented database system, C3, and describes its
specification, design and implementation. The C3 constraint object-oriented database system
[8] has been designed to be used for both implementation and optimization of high-level
constraint object–oriented query languages such as LyriC or constraint extensions of OQL.
C3 can also be used for directly building software systems requiring extensible use of constraint database features. It [8] provides the C3 data manipulation language and Constraint Comprehension Calculus. CCC is an integration constraint calculus for extensible constraint domains with monoid comprehensions. CCC serves as an optimization-level language for object-oriented queries. It has been proposed in [8] that the data model for constraint calculus based on constraint spatio-temporal (CST) objects may hold spatial, temporal or constraint data conceptually represented by constraints. I have developed a Constraint Model of Object-Oriented data in [84].

2.3 Constraint Models in UML

The Unified Modeling Language (UML) [36], [62], [74] is the result of an effort in developing a single standardized language for object-oriented modeling. The initial effort focused on the identification and definition of the semantics of fundamentals concepts, the building blocks, of object-oriented modeling. These fundamentals concepts are the artifacts of the development process and must be exchanged between different parties involved in a project. To facilitate and to help formalize UML, all different concepts have been modeled using a subset of UML, called metamodeling [62]. A metamodel formally describes the syntax and the semantics of the model elements and the notation that allows their manipulation. The added abstraction introduced by the construction of a metamodel facilitates the discovery of potential inconsistencies and promotes generalization. UML has defined a number of common mechanisms that ensure conceptual integrity of the notation [62]. These common mechanisms comprises of stereotypes, tagged values, notes, constraints, dependencies, type-instance and type/class dichotomies. Stereotypes, tagged values and constraints facilitate the extension of UML. Stereotypes specialize metamodel classes, tagged values extend attributes of the metamodel classes and constraints extend the metamodel semantics.

Each model element has a specification that contains a unique and detailed description of all the characteristics of the element. A model element captures the underlying semantics of a problem and contains data accessed by tools to facilitate information exchange, code
generation, navigation etc., but is not directly visible by users. UML has defined several models for representing systems while different models are browsed and manipulated by users by means of graphical representation. UML has developed nine different types of diagrams to represent all or part of the characteristics of the model elements [74]. Due to wide acceptance of UML in Object-oriented analysis and design, UML has been described as a formal modeling notation in [36].

Object-oriented databases are not as widespread as relational ones. Due to increasing popularity of UML for object oriented analysis and design, there has been a good amount of efforts of mapping between UML and Entity relationship Model to make the objects persistent [72], [95], [96]. A scheme for transforming UML class diagrams into relational data models has been introduced in [95]. It [95] describes the process to map a UML class diagram into an ER diagram and discusses the potential of using the UML notation to draw ER diagrams. An example of an actual systems design has been used in [95] to illustrate the mapping process. It [95] also introduces associated problems encountered while mapping from UML class diagrams to ER diagrams and discusses how these problems could be resolved. Ou proposed another mapping scheme between UML and entity relationship model in [72]. It [72] introduces a set of translation rules to automate the translation of a UML class diagram to ER diagram and vice versa, which can be useful for forward and reverse engineering of database design. It [72] has also been proposed that in forward engineering, a software application then be stored in an object-oriented database or in a relational database. This paper [72] considers class diagrams only for mapping between UML and ER model. In [96], a relational object model has been introduced.

Design and maintenance of integrity constraints in UML models is rapidly emerging in the context of consistency preservation. In UML specification, a constraint can be defined by putting a text string inside braces ({{}}) or by using a note or a comment and attaching it to particular Model element. According to UML, any kind of Model Element can be associated with a constraint. Such constraints are often described by using natural languages. However, all these constraints can also be expressed by using formal language such as UML OCL (Object Constraint Language). The Object Constraint Language (OCL) [77], which forms
part of the UML set of modeling notations, was introduced to support specification of constraints for UML models that could not be shown in the standard graphical representation used in UML. OCL is not a programming language and thus it is not possible to write program logic or flow control in OCL. The different kinds of constraints that can be expressed using OCL include invariants on classes, types and interfaces, preconditions and post-conditions of operations and methods. The Object Constraint Language (OCL) is intended to facilitate design of model properties in a formal way but OCL itself was mainly defined in a semi-formal way by using English text descriptions, context free grammar specifying the concrete syntax of OCL and many examples illustrating the intended meaning of expressions.

In recent years, formalization and enhancement of OCL has still been evolving in the context of specifying constraints and queries in UML [34], [44], [45], [68], [79], [81], [103]. A formal semantics for the Object Oriented Constraint Language (OCL) has been presented in [79]. It has been argued in [79] that OCL constraints and queries can be specified in a formal yet comprehensible way, OCL itself is defined only in a semi-formal way by using English text, descriptions, a context free grammar specifying the concrete syntax of OCL. Thus, the semantics of constraints is not precisely defined in general. To overcome this, [79] has presented a formalization of OCL and certain aspects of UML class modules. It [79] also defines the syntax and semantics of OCL expressions and demonstrates examples for their evaluation. Another approach for better understanding of constraints and queries in UML using OCL has been introduced in [81]. First, it [81] presents some examples for illustrating main concepts of OCL. It [81] also reports the problems with the current definition of OCL resulting from imprecise or ambiguous definitions. Finally, comparison of OCL with a language for specification of queries and integrity constraints in an Extended Entity-Relationship model has been presented in [81].

In [44], semantics of OCL expressions, the foundation for building CASE tools that support integrity checking of UML models, has been provided. An informal introduction using OCL in combination with a kernel of the UML diagrammatic notation has presented. It [44] establishes the semantic framework by giving semantics to class diagrams. It [44] also
provides semantics for classes, associations, attributes and states. Specification of real-time properties for UML models using OCL has been presented in [34]. It has been argued in [34] that OCL currently does not provide sufficient means to specify constraints over the dynamic behavior of a model. This work [34] has introduced an extension of OCL that is consistent with current OCL and enables modelers to specify state-related time-bounded constraints. The constraints are presented in temporal OCL extensions as well as in temporal logic formulae in [34]. It [34] also defines semantics of OCL extension by means of a time-bounded temporal logic based on Computational Tree Logic (CTL). Some enhancements of OCL that make specifications convey information more effectively has been introduced in [45]. These enhancements have also been integrated in some specification languages in [45].

In large software development projects, shared databases support cooperation of developers and reuse of design. Generation of database schema and object-oriented database application programming interface (API) from a graphically specified UML models has been provided in [68]. This paper [68] reports a UML repository based on UML metamodel and managed UML models which in turn, are taken as inputs by a generator to create parts of a database application automatically. UML repository has implemented by using an ORDBMS. This approach [68] allows one to use OCL for both checking the validity of UML models and for maintaining the consistency of application data. It has also been argued in [68] that their approach provides a foundation for rigorous modeling and automated model analysis. Foundations and methodological aspects specifying with OCL has been discussed in [103]. OCL consists of a pure Boolean expression language, which is used to formulate invariants and pre and post conditions in class diagrams. In this paper [103], it was first reported the recent advances for analyzing the expressiveness and defining semantics of the pure Boolean expression language. Secondly, the problems and open issues of the interplay of invariants and pre and post conditions have been introduced in [103]. Finally, it [103] demonstrates with examples the navigational power of OCL leads to constraints, which cannot be correctly implemented in a programming language like Java. To overcome these, a method for using OCL constraints that supports component-based approach to system specification has been proposed in [103].
Due to the necessity of maintaining consistency in object-oriented database systems, other approaches for constraints management in UML models without using OCL have also evolved [52], [53], [55], [69], [71], [97], [104]. A formal object-oriented metamodeling approach for defining consistency constraints in UML models has been introduced in [52]. In this approach [52], integrity consistency constraints between UML models has defined in terms of invariants of the UML model elements, which can be used to represent the models at the language level. Adopting a formal approach, constraints have formally been defined using object-z in [52]. It has been proposed in [52] that the formal descriptions of the consistency constraints would be a precise reference of checking consistency of UML models as well as for tool development after defining the integrity consistency constraints for UML models at the language level. A metamodel semantics for structural constraints in UML has been proposed in [53]. A metamodel that incorporates precise semantics would support the construction of tools that could perform semantical tasks such as consistency checking. Formal semantics of a language can be defined by essentially elaborating (in mathematics) the value or instance denoted by an expression of the language in a particular context. Thus, a metamodel can be constructed which incorporates the modeling language itself, the modeling language of instances and the mapping of one into the other, although current UML metamodel does not provide supports to the mapping of one metamodel language to other. [53] presents one such metamodel suitable for describing and constraining object structures for a fragment of UML, which includes parts of class diagram and invariants in the style of OCL. An indication is also given in [53] how this metamodel could be extended to model characterizing dynamic behavior.

A methodology to define UML diagrams in terms of state predicates and using theorem prover PVS (Prototype Verification system) to verify consistency between various diagrams has been presented in [55]. This paper [55] focuses on the dynamic aspects of various diagrams of UML. It has been argued in [55] that it could easily handle partially specified systems as the behavior is described in terms of the history of the computation. Another constraint checking technique in UML Modeling has been introduced in [97]. In [97], a classification of constraints has been proposed to clarify the nature of the constraints that must be fulfilled to ensure model correctness within the context of UML. In UML,
constraints are assertions described in a side effect free language (OCL), but actions are not supplied while constraints are violated. Depending on the kind and the context of the constraint violations, help and advice has been supplied in [97] where improvements may be conditionally done in this approach. The constraints have also translated into modeling rules that form the knowledge base of an expert system in modeling [97].

A formal model of real-time objects through the definition of the UML package, RT-object, has been presented in [104]. [104] has defined some classes in the RT-object package which has extended some of the most basic classes in the UML metamodel to provide a mechanism for expressing timing characteristics on object-oriented data. These characteristics include representation of the age of an attribute, temporal consistency constraints expressed on attributes, imprecision on attributes and imprecision constraints on attributes. It [104] has also introduced a compatibility function and the expression of real-time operations, read affected sets and write affected sets, which together provides a mechanism to support fine-grained, method level concurrency control in real-time objects. In [71], Ou has proposed an extension to the UML metamodel. Ou has introduced two new model elements Key and Iconstraint and some new attributes to the existing metamodel to accommodate more constraint specification in UML. In the model level, a compartment called constraint has been added to the class notation and some property strings has been added. In this work [71], UML class diagram have been considered for specification of constraints and constraints are restricted to database integrity constraints only. In [85], I have proposed the extension of UML metamodel to formalize the constraint model [84] using UML.

Object Definition Language (ODL) is a proposed standard language for specifying the structure of databases in object-oriented terms. ODL allows to transforms the object-oriented designs of database directly into declarations of an object oriented database management system (OODBMS). In [71], Ou introduced a mapping scheme to derive ODMG-ODL schema design from UML class diagram. The different types of constraints that has been considered in [71] are keys, uniqueness etc. I have also presented a mapping scheme in [86] to obtain ODMG-ODL database design from extended UML Model. The constraint model [84], which I have considered is entirely different from [71].
2.4 XML and Constraints

The Extensible Markup Language (XML) [9] [100] has become a widespread standard technology for representing complex documents and information exchange over the web. It [100] offers a convenient syntax for representing data from heterogeneous sources. In many cases, XML data is generated from legacy repositories (relational or object databases, proprietary file formats, etc.), and is exported to a target application (Java applets, document management systems, etc.) [1], [4], [16], [22], [35], [41], [43], [58], [59], [63], [76], [93], [94], [105], [108]. An approach for translating relational data into semi-structured data and XML has been introduced in [1]. In [4], another approach for storing XML data into relational databases has been presented. It [4] introduces a cost-based XML storage mapping engine, LegoDB, that explores the possibility of XML to relational mappings for a given application. LegoDB models the target application with an XML schema, XML data statistics and an Xquery workload. The space of configuration is generated through XML schema rewritings and the derived configuration is obtained through a standard relational optimizer. This paper [4] describes the LegoDB storage engine and provides experimental results to demonstrate the effectiveness of their approach.

A correspondence of information exchange between XML and DB2 has been proposed in [16]. It has been proposed in [16] that this approach can satisfy a number of goals: (i) An application can generate XML documents from SQL queries against DB2 or any ODBC compliant databases starting with DB2; (ii) An user has the option to store the entire document as an XML user-defined column or to decompose the document into multiple tables and columns so that faster searching techniques can be used for both XML elements and attributes; (iii) An user can retrieve the entire document or extract XML elements and attributes dynamically in an SQL query; (iv) XML Extender provides stored procedure to generate XML documents from existing data and together with Net.Data, an user can browse the content of the XML documents via the internet. In [35], a framework for publishing relational data in XML, Silkroute, has been presented. [41] has proposed an object representation model for XML data. A set of transformation rules and steps has been established in this paper [41] to transform DTDs as well as XML documents with DTDs into
object-oriented model. This model [41] capsulizes elements of XML data and manipulation methods. A DTD-tree has also been defined to represent DTD and the paper describes a procedure to use transformation rules. It has been proposed in [41] that DTD-Tree can also be used as a logical interface for DTD processing.

An Object-Relationship-Attribute model for semi-structured data to support the schema conversion from semantically enriched relational schema to XML schema has been introduced in [22]. The inherent semantics and implicit structure in the relational schema, such as object relations, relationship relations, fragments of object relations or relationship and cardinality constraints have been considered in this translation. This approach [22] also allows one to handle the translation of a set related relation and to distinguish attributes of relationship types from attributes of object classes, multi-valued attributes and different kinds of relationships such as binary, n-ary, recursive and ISA. In this paper [22], relations are classified into three categories, namely, object relation, relationship relation and mix type relation while attributes are classified into two categories, namely, object attributes and relationship attributes respectively. The proposed relational to XML schema conversion approach [22] has a number of objectives: (i) to generate an XML structure that is able to describe the semantics and structure in the underlying relational database; (ii) to allow the translation of a set of related relations instead of simple single relation/relationship conversions; and (iii) To obtain properly structured XML data without unnecessary redundancies and proliferation of disconnected XML elements. Another technique to convert relational schemas to XML schemas using various semantic constraints has been described in [58]. Two conversion algorithms, called NeT and CoT have been introduced in this work [58] to accomplish the conversion. First, it [58] presents a language-independent formalism named Xschema to generate output schema in various XML schema Language proposals. The objective of such formalism [58] is both precise and concise. Based on Xschema formalism, this proposed algorithms [58] provides the following characteristics: NeT derives a nested structure from a flat relational model by repeatedly applying the nest operator so that the resulting XML schema becomes hierarchical and CoT considers the inclusion dependencies during the translation so that relational schemas where multiple tables are interconnected through inclusion dependencies can be handled as well.
In [63], XML features in relational databases have been reviewed and their usage in database applications has been described. Different aspects that have been considered in this work [63] included the creation, validation, transformation storage and retrieval of XML documents, the inclusion of existing and new relational data in XML documents and the impact of XML Links. Another methodology for efficiently publishing relational data as XML documents has been provided in [93]. It has been argued in [76] that the tree structure of XML documents induced by mark-up nesting does not provide sufficiently expressive and general conceptual model of data in the documents, particularly when multiple source documents are processed at the same time. To deal with this problem, [76] proposes the ERX (Entity Relationship for XML) data model, an evolution of the Entity Relationship model to cope with the features of XML. ERX [76] is devised to provide an effective support to the development of complex XML processors for advanced applications. [105] has proposed a scheme to model object-oriented frameworks using XML and an methodology has been developed to support effective retrieval of reusable object-oriented frameworks. First, it [105] introduces the modeling of object-oriented frameworks and also the components and metamodel for object-oriented frameworks using XML. This object-oriented framework [105] is composed of classes and their hierarchy and their interactions. This framework [105] may apply several design patterns in its design phase, where design patterns are realized by implementation of specific classes in the framework source code. So, the major components comprising the framework in this work [105] are design patterns and classes. Next, it [105] presents the rationale for applications of XML to represent object-oriented frameworks. In addition, this work [105] also develops structural and semantic relationships inherent in object-oriented frameworks to search reusable frameworks. Retrieval and browsing techniques of XML based object oriented framework has also been introduced in [105]. It has been said that this approach [105] increases effectiveness of search by expanding similar terms through query reformulation.

A multimedia XML data is a collection of multiple types of data sets tagged by XML elements. A method for schema extraction for multimedia XML data has been provided in [108]. Schemas are levelized with respect to the frequency of topological document
structures in a database in [108], which can be used to formulate queries and to rewrite queries for relaxation and restriction. An approach, called UXF approach, has been provided in [94] for making UML models exchangeable over the web with XML. In [43], a simple notation for describing data representations intended for XML encoding has been presented. This notation [43] is based in part on a small subset of UML. The goal of this effort [43] is to develop a consistent, systematic method for translating UML-based class diagrams into XML schemas and to maintain a reasonably simple notation. The different types of object-oriented constructs that has been considered into this [43] UML to XML Schema translation are class and objects, attributes (both simple attributes and complex attributes), intra-links between the objects, inter-links between the objects, the association between the objects and element values etc. and the constraints on attributes values which are available in UML like attribute value chosen from a strictly limited range, optional attribute and minimum and maximum range values on numeric attributes. It [43] also described how inheritance can be incorporated by using XML Schema. However, the scope of this notation [43] is limited to classes and their attributes, not class methods or operations.

XML is now a widespread standard used for representing complex documents. Information formatted in XML is described in terms of a document containing markup and text. The structure of the document, meaning the permitted vocabulary and organization of entities in the document, can be optionally predefined in the form of either a Document Type Definition (DTD) or an XML Schema. A powerful feature of XML is the notion of a Document Type Definition or DTD. The DTD defines how data is formatted. It also defines each allowed element in an XML document, the allowed attributes and possibly the attribute values for each element, the nesting and occurrences of each element and any external entities. A DTD file contains the document type definition of the data, which is a formal description of the tags. DTD enables to describe the relationships between tags in an XML document to capture the semantics of a particular domain, however, it does not allowed to define data types for individual tags. In this context, XML Schema Definition language is a very powerful construct for defining document structures as well as complex data types using aggregation and inheritance.
The Extensible Markup Language (XML) [100] offers appropriate syntax for representing data, but provides little semantic information. The ID/IDREF attributes of XML DTDs provide a mechanism in a way similar to relational keys and foreign keys. However, the ID/IDREF mechanism of XML DTDs is too weak in terms of expressive power. On the other hand, XML Schema [100] supports a complex form of keys and foreign keys defined with Xpath expressions. To specify the semantics of XML data, a variety of approaches have been proposed [11], [15], [26], [30], [31], [33], [61], [67]. [11] has introduced the definition of keys for XML documents, paying particular attention to the concept of a relative key, which is commonly used in hierarchically structured documents and scientific databases. This paper [11] extended the key specification of XML data by allowing one to specify keys in terms of Xpath expressions. The concept of relative or hierarchical keys together with its alternative notation has been discussed in this work. It [11] also examined the XML schema proposal in some detail and discussed an alternative form of keys and various issues concerning keys. In [15], a novel index structure, termed Xtrie that supports the efficient filtering of XML documents based on Xpath expressions has been provided. Xtrie index structure [15] offered a number of features for large-scale publish/subscribe systems. First, Xtrie supports effective filtering based on complex path expressions. Second, Xtrie structure and algorithms supports both ordered and unordered matching of XML data. Third, by indexing on sequences of element names, Xtrie organized in a trie structure using a sophisticated matching algorithm. It has been argued in [15] that Xtrie is capable to reduce the number of unnecessary index probes as well as avoid redundant matchings.

A constraint-based extension of XML for specifying both logical structure and semantic coherence of a website, CobWeb, has been introduced in [67]. The objective of this paper [67] is to place restrictions on the values of the elements as well as attributes in the DTD. The different types of constraints that have been considered in this paper [67] are unary or domain constraints and binary constraints including various comparison operations as well as various aggregation constraints (sum, average, etc.). A unary constraint is a constraint that acts upon a single variable, such as a domain constraint. In a domain constraint the value being constrained must belong to a domain of values while a binary constraint affects two or more values so that comparisons between groups of values is possible. CobWeb, constrained
XML for the web [67] allowed developers to expand upon the notion of viewing the web as part of a database, not only in a specific structure but also checked for validity. This paper [67] also demonstrated with example how CobWeb could be used as a solution to common problems in websites such as verifying data content and links across multiple, interconnected web pages. In [33], XML integrity constraints in the presence of DTDs have been presented. [61] has introduced an approach to capture XML constraints with relational schema.

The constraints and queries reformulation problems for XML publishing in a general setting that allowed mixed (XML and relational) storage for the proprietary data and exploits redundancies to enhance performance has been provided in [26]. The correspondence between published and proprietary schemas is specified by views in both directions and the same algorithm performed rewriting-with views, composition-with-views or the combined effect of both, unifying the Global-As-View and Local-As-View approaches to data integration in this work [26]. This paper [26] presented an algorithm for finding the minimal reformulations of client Xqueries in XML publishing scenarios, when the correspondence between public and storage schema is given by a combination of GAV (Global-As-View) and LAV (Local-As-View) Xquery views. The algorithm [26] also handled in the same unified way redundant storage, constraints in XML data (as specified by XML Schema) and constraints in the relational storage. It has been specified in [26] that their algorithm is complete and asymptotically optimal for an expressible class of client query and views and integrity constraints. It has been also stated in [26] that this approach should be useful for publishing integrity constraints to help clients understand the semantics of the published data.

[30] deal with several classes of integrity constraints for XML and investigated the complexity of reasoning about these constraints. The different types of constraints that have been considered into this work [30] range over keys, foreign keys, inverse constraints as well as ID constraints for capturing the semantics of object identities. This work [30] has the following contributions. (i) First, it [30] introduced a model for XML data with schema and integrity constraints. The three basic constraints languages, namely, L, Lid and Lu have proposed to provide both a reference mechanism and better semantic specifications. Constraint language Lu is a minimal extension of the original ID/IDREF mechanism while
constraints of Lid and L are used to capture semantic constraints when data originates in object-oriented databases. (ii) Secondly, it [30] provided the implication and finite implication problems for the three constraint languages L, Lid and Lu. (iii) It [30] also provided the implication of more general forms of constraints including functional inclusion and inverse constraints defined in terms of navigation paths by basic constraints of Lid. It has been stated in [30] that such path constraints have a variety of practical applications ranging from query optimization to verification of the correctness of integrity/transformation programs. This paper [30] formalized DTDs as a combination of a structural specification and a set of integrity constraints.

A unified constraint model for XML UCM, which is both simple and expressive have been discussed in [31]. This work [31] used a single notation of keys and foreign keys using a limited form of Xpath expressions. The main objective of UCM is a tight coupling of the integrity constraints with the schema language. It [31] handled the problem of consistency of UCM constraints, the interaction between constraints and sub-typing and algorithms for implementing these constraints. This work [31] has the following contributions. (a) It [31] extended the type system along with the sub-typing mechanism with a notation of keys and foreign keys. This contributed UCM, a schema language for XML with integrity constraints. (b) UCM schemas can captures relational constraints, object oriented models and the ID/IDREF mechanism of DTDs. (c) It [31] has proposed a practical restriction over UCM schemas that guaranteed consistency. (d) It [31] presented an algorithm for propagating constraints through sub-typing. This mechanism is the basis for supporting the notation of object identity of object models within UCM schemas. (e) It also presented algorithm for schema validation in the presence of UCM constraints. I have presented an efficient way to represent the constraint model [84] using XML schema in [88].

2.4 XML and Interoperability with Constraints

The data interoperability problem arises from the fact that data, even within a single domain of application, is available at many different sites, in many different schemas and even in different data models (e.g., relational and XML). In this context, querying and integrating
data from different heterogeneous sources has been evolved increasingly [21], [49], [60], [66], [75], [99], [102]. In [49], the extension of the Interface Definition Language (IDL) has been proposed for XML-based integration. A signature-based query optimization for XML to minimize the numbers of nodes retrieval from the database has been discussed in [75]. To facilitate query evaluation in XML, XML document is represented as a tree and query is expressed as a regular path expression in this work [75]. The query is evaluated by traversing each node of the tree. In this paper [75], the signature is a hint attached to each node and used to prune unnecessary sub-trees as early as possible when traversing nodes. It [75] has also been introduced a signature-based DOM, S-DOM, as a storage model and a signature-based query executor, S-NFA, to retrieve queries efficiently. It has been stated in [60] that the problem of designing data integration systems is important in current real world applications and is characteristics by a number of issues that are interesting from a theoretical point of view. An overview of the material to be presented in a tutorial on data integration has been provided in [60]. This tutorial [60] is focused on some of the theoretical issues that are relevant for data integration. The objective of this work [60] is to devote special attention to the aspects of data integration like modeling a data integration application, processing queries in data integration, dealing with inconsistent data sources and reasoning on queries.

A comprehensive framework for the management and the exchange of semi-structured web data has been described in [99] according to a variety of formats and models. This paper [99] has considered various schema definition languages for XML such as DTD, XML Schema and XDR, a model for semi-structured data, OEM, and a model used to store web data (the relational model) to show that the primitives adopted by all of them can be classified into a rather limited set of basic types. Building on these basic types, a notation of “meta-formalism” has been provided in [99] that can be used to describe the heterogeneous representation of web data in a uniform way. In this framework [99], the translation of schemes and instances between different models are based on the translations of the involved primitives. It has been also stated that complex translations between different heterogeneous schemes of XML data can be obtained by simply combining a number of predefined operations, which implemented standard translations between primitives. The goal of this approach [99] is to support a number of involved web-related activities such as information
exchange between different organizations, integration of data coming from heterogeneous information sources, storage of native XML data in a DBMS and publishing of existing structured (relational) data in XML. An XML Query language for heterogeneous data sources, Quilt, has been introduced in [21]. Quilt [21] is designed to support queries against a broad spectrum of information sources by incorporating features from several languages so that it can exploit the full versatility of XML. This paper [21] has illustrated the versatility of Quilt by using it to express queries against both semi-structured documents and relational databases. Quilt is a functional language in which a query is represented as an expression. Quilt supports several kinds of expression and therefore a Quilt query may take several different forms. The various forms of Quilt expressions can be nested with full generality, so the notation of a sub-query is natural to Quilt. The input and output of a Quilt [21] query are XML documents, fragments of XML documents, or collections of XML documents. It has been specified in [21] that Quilt represented a promising approach to a query language that can help to realize the potential of XML as a universal medium for data interchange.

A method of representing message formats in XML using data types available in the XML Schema specification has been designed in [102]. This paper [102] has decomposed the manipulation of metadata into three separate steps - discovery, binding of program objects to the metadata and marshaling of data to and from wire formats as described below. (a) Discovery – In this step, one of the two alternatives approaches can be used. In the first approach, metadata is provided in a structured manner defined by the Binary Communication Mechanism (BCM) and made available to the BCM library as part of program execution. In the second approach, Interface definition languages (IDLs) enabled the expression of architecture independent message definitions. An IDL compiler for each participating host platform translated these definitions into architecture dependent definitions and code generators provided library routines to translate messages at run-time. (b) Binding – In this step, a particular metadata format is associated with a particular piece of program data. (c) Marshaling – In this step, the message is reconstructed with the help of metadata by generating wire format messages from in-memory format messages. A tool xml2wire, has implemented in this paper [102], that used such metadata and exploited this decomposition in order to provide flexible metadata definition facilities for an efficient binary communications
mechanism. \textbf{Xml2wire} [102] is a run-time library, whose function is to convert metadata definitions expressed in XML into metadata suitable for particular binary communications packages. In [66], an approach has been presented for answering XML queries on heterogeneous data sources.

The purpose of data integration is to support seamless access to autonomous, heterogeneous information sources, such as legacy databases, corporate databases connected by intranets, and sources on the Web. Data integration systems provide access to a set of heterogeneous, autonomous data sources through a so-called global schema. There are basically two approaches for designing a data integration system. In the global-as-view approach, one defines the elements of the global schema as views over the sources, whereas in the local-as-view approach, one characterizes the sources as views over the global schema. It is well known that processing queries in the latter approach is similar to query answering with incomplete information, and, therefore, is a complex task. On the other hand, it is a common opinion that query processing is much easier in the former approach. In such an environment, sources can have various constraints on their contents, which can be utilized in query processing and optimization. The importance of allowing integrity constraints in the global schema has been stressed in several works on data integration [10], [14], [19], [20], [57], [107]. A framework for integrating data from multiple relational sources into a XML document has been introduced in [10] that conforms both syntactic and semantic constraints. This framework [10] is based on a specification language, AIG, which extends a DTD by (i) associating element types with semantic attributes (inherited and synthesized, inspired by the corresponding notions from Attribute Grammars); (ii) computing these attributes via parameterized SQL queries over multiple sources and (iii) incorporating XML keys and inclusion constraints. It has been stated in [10] that AIGs not only yields a uniform framework for ensuring both DTD-conformance and XML-constraint satisfaction in data integration, but also provides the ability to support data-driven and context-dependent generation of XML documents. It [10] also presents cost-based optimization techniques for efficiently evaluating AIGs, including algorithms for merging queries and for scheduling queries on multiple data sources. This provides [10] a new grammar-based approach for data
integration under both syntactic and semantic constraints. An approach for simplification of integrity constraints for data integration has been presented in [14].

The issue of dealing with integrity constraints over the global schema in data integration has been proposed in [19]. This paper [19] develops a data integration system based on relational model with integrity constraints by taking into account a number of issues such as (i) integrity constraints can be used to extract more information from incomplete sources similar to the case of databases with incomplete information; (ii) integrity constraints raise the problem of dealing with the incomplete information and possibly with inconsistencies. This paper [19] has also been investigated how in data integration the presence of integrity constraints on the global schema has a significant impact on the semantics of a data integration system. Different types of constraints that have been considered in this data integration system [19] are keys and foreign keys over global schema. [57] has been studied how to describe various constraints in data integration systems so that they can be utilized in query processing and optimization. This paper [57] considers the local-as-view approach to data integration (under the open-world assumption) in which source contents and user queries are formulated on predefined global predicates. In this approach [57], source contents and constraints often exist before the global predicates are designed. It [57] discusses two different levels for describing constraints: (i) one is local constraints which are defined on different sources; (ii) another is global constraints which are defined on global predicates. This paper [57] formally defines two types of global constraints, namely, general global constraints and source-derived global constraints. The advantages of having these constraints in data integration system have also been presented in [57].

A constraint-based XML query Rewriting technique for data integration has been introduced in [107]. This paper [107] studies the problem of answering queries efficiently through a target schema, where a set of mappings between the source schemas and the target schema are given and the data are stored at the sources. Queries and mappings over both the relational and XML schemas have been considered in this paper [107]. This work has a number of contributions: (i) It [107] defines how to answer a target query in the best way for a given set of mappings between the source schemas and the target schema and in the
presence of a set of target constraints. A canonical target instance that satisfies all the requirements with respect to the given source instances has been defined. (ii) This paper [107] defines the semantics of query answering in data integration scenario and designs two novel algorithms, namely, basic query rewrite algorithm and query resolution, to implement the semantics. The basic query rewriting algorithm reformulates target queries in terms of the source schemas based on the mappings. The query resolution algorithm generates additional rewritings that merge related information from multiple sources and assemble a coherent view of the data, by incorporating target constraints. (iii) The algorithms are implemented and then evaluated using a comprehensive set of experiments based on both synthetic and real-life data integration scenarios in [107].

Data integration under integrity constraints has been presented in [20]. This paper [20] concentrates on two basic issues for designing a data integration system. One is specifying the mapping between the global schema and the sources and another is processing queries expressed on the global schema. It [20] uses two approaches to specify the mapping between the sources and the global schema. The first approach, called global-as-view, requires that the global schema to be expressed in terms of the data sources. The second approach, called local-as-view, requires the global schema to be specifies independently from the sources. It focuses on global schemas expressed in relational model with key and foreign key constraints, which represents a common situation in practice and then illustrates techniques for answering queries effectively posed to such a data integration system. It has been shown in this paper [20] that the presence of integrity constraints in the global schema poses new challenges, specially related to the need of talking the semantics of constraints into account during query processing.

Data binding provides a simple and direct way for mapping XML in Java Platform applications. With data binding, applications can ignore the actual structure of XML documents and work directly with the data content of documents. Data binding can also provide other benefits beyond programming simplicity. Since it abstracts away many of the documents details, data binding usually requires less memory than a document model approach such as DOM or JDOM. The data binding approach gives faster access to data within program, since it does not require to go through the structure of the document to get
data. However, data binding is not suitable for all applications, but it is ideal for the common case of applications that use XML for data exchange. With data binding, there are three distinct processes, namely binding, marshalling and unmarshalling, which can occur one after another, in differing order or in completely unrelated processes. In [78], an overview of an efficient way to store XML data inside an object-oriented database management system (OODBMS) has been provided. It [78] first discusses the difference between XML data and XML documents, and then introduces an approach to integrate XML data into Java programming language and programming model. This integration [78] is combined with the transparent persistence of Java objects defined by the Object Data Management Group (ODMG).

There are a few data binding frameworks available currently to help us to do this, but I select Castor [32], an open source data binding framework from Exolab, in this work. Castor [32] data binding framework is a path between Java objects, XML documents, SQL tables and LDAP directories. However, Castor [32] data binding framework does not provide facilities to deal with constraints. In [91], I have proposed some extensions of the Castor data binding tool to ensure interoperability of XML data with constraints into Java object model. The detailed description of the extension of Castor [32] data binding tool is identified and completely specified in this work [91].

A wrapper is a piece of software, which provides the view of one interface to another without modifying the underlying component code. A translator function can be anticipated with a wrapper. To solve the data and operation inconsistency problem in legacy systems, a wrapper-based Object Oriented Model for interoperability has been presented in [109]. This paper [109] defines a federation interoperability object model for interoperation of a specific federation of systems. I have developed a wrapper-based translator to achieve interoperability between XML and Object-Oriented Systems (such as Java and C++) in this work. This wrapper can be used in combination with Castor [32] data binding tool.