CHAPTER 2

LITERATURE SURVEY

This section provides a survey of related works in the areas of ER modeling, normalization and refactoring. In section 2.1, research works carried out in the area of entity-relationship modeling are discussed. In section 2.2, research works carried out in normalization are discussed. Section 2.3 discusses the research works carried out in the area of refactoring techniques.

2.1 ENTITY-RELATIONSHIP MODELING

Abrial (1974) proposes data semantics model. This model organises various objects in a database into different categories. Binary relations define atomic links between pairs of objects belonging to certain categories. The two directions of a binary relation are named uniquely. Each name, called an access function, corresponds to a binary relation followed in one direction. It is to be noted that access functions in this model are not really functions in the mathematical sense of the word, since their application can yield more than one value. Relationship types between more than two categories are represented by generating new categories. Abrial's model provides logical access to data by means of elementary operations on access functions via programs. It also provides operations for manipulating objects in a category (e.g., introducing a new object) and for manipulating connections between
objects (e.g., relating an object in one category to an object in another category). In addition, one can define operations that take a more general form and affect many objects in many categories according to a program.

Schmid and Swenson (1975) proposed a model in which the real world is considered as consisting of objects and associations. An object can either be independent or dependent; the difference is that a dependent object must be "existence-dependent" on some independent object, whereas independent objects exist on their own right. Both dependent and independent objects can have properties. An association is a relationship between independent objects. Both objects and associations are represented as relations. In fact, relations in this model are classified into five different types, according to the type of information they represent. Five different relation types are: relation that represents an independent object type, relation that represents a repeating characteristic object type, relation that represents a complex characteristic object type with a key, each relation that represents a repeating complex characteristic object type with a key and each relation represents an association. Associated with this categorization is a set of integrity rules; for example, an association can be created only if all associated objects already existed. Conversely, an object cannot be deleted if it currently participates in any association. This model was proposed since questions of a semantic nature cannot be answered completely by it due to the mathematical nature of the relational model.

Senko (1975), proposed Data Independent Accessing Model (DIAM) which has five different levels: the end-user level, the information level, the string level, the encoding level, and the physical device level. The end-user level, corresponding to the ANSI/SPARC external level, consists of entity, property, fact, entity set and enterprise. At the information level, names
of things are dealt with; hence the terms used are attribute name, attribute value, identifier name, identifier value and fact representation. A fact representation is a pair of attribute names, also called an association pair, linking identifiers. The association pair is symmetric, one attribute name uses one identifier as the subject and the other identifier as the value, while the other attribute name does just the opposite. N-ary and many-to-many associations are treated by creating artificial entities. The language associated with DIAM II is called FORAL.

Chen (1976) proposed a data model, called the entity-relationship model. The information about an entity or a relationship is obtained by observation or measurement and is expressed by a set of attribute-value pairs. The ER model adopts the more natural view that the real world consists of entities and relationships. It incorporates some of the important semantic information about the real world. The model can achieve a high degree of data independence and is based on set theory and relation theory. Chen used the ER model as a framework from which the three existing data models namely network model, the relational model and entity set model may be derived. It is explained that the decomposition or transformation approach described for normalization of relations may be viewed as a bottom-up approach in database design. The ER model adopts a top-down approach, utilizing the semantic information to organize data in entity-relationship relations. Chen states that the concept of attribute of relationship is important in understanding the semantics of data and in determining the functional dependencies among data. Further, ER model will be useful in understanding and specifying constraints for maintaining data integrity.
Elmasri and Wiederhold (1979) introduce various types of connections between relations to enhance its constraints. Since then, numerous modifications to the ER model have been suggested.


Jajodia et al (1983) have made an investigation to find out when two entity diagrams should be considered equivalent, when both represent the same information about the given real world. It is possible to obtain several structurally different ER diagrams based on the given description of the requirements of the database system, each of which conforms to the description of the application in an informal sense. The criteria for ascertaining the concept of equivalence for ER diagrams are discussed based on notions of “domain data compatibility”, “data dependency equivalence” and “instance data equivalence”. Based on this, it is stated that it is possible for the database designer to select the one with the highest form for subsequent implementation.

Hawryszkiewycz (1984) stated that the n-ary relationships have a number of undesirable features. One feature is that it is no longer meaningful to model the ER diagram with 1:N, N:M or N:1 relationships. Another undesirable feature is that n-ary relationships can become clumsy, especially if a very large number of entities participate in a relationship. He provided solutions for translating a ternary relationship into a set of binary relationships. This technique can be used to translate the ERD into a binary representation of the ERD. In the conversion approach, the ternary relationship is converted into a new entity type, and a binary relationship is
created between the new converted entity type and each participating entity type where the cardinality of those binary relationships is M:1.

Teorey et al (1986) have reported that the basic transformation has been developed for the three types of relations namely entity, extended entity and relationships relations. A practical step-by-step methodology for relational database design can be derived using a variety of extensions to the ER model. To attain the desired highest degree of normalization, further analysis and modifications were done on candidate relations. They have proposed that the cardinality ratio of the ternary relationship can be one to one to one (1:1:1), one to one to many (1:1: M), one to many to many (1: M: N), and many to many to many (M: N: P). The functional dependencies that can be derived for the various cardinalities of a ternary relationship have all been discussed. The same is discussed in chapter one with examples.

Song et al (1995) compared ten different ERD notations, widely used in ER diagram. They are from database books and CASE tools used for the design of relational databases. They have listed the details of the books and tools discussing n-ary models. According to their investigation, they have found that ERD notations differ based on seven perspectives. Notation vary depending on whether they allow n-ary relationships, whether they allow attributes in a relationship, how they represent cardinality and participation constraints, the place where they specify constraints, whether they depict overlapping and disjoint subclass entity-types, whether they show total or partial specialization, and whether they model the foreign key at the ERD level.

Jones et al (1996) stated that it is impractical and inappropriate to consider ternary relationship decomposition under the umbrella of higher
normal form decomposition. Higher normal forms involve the decomposition of composite key relations into simple relations, eliminating the inherent duplication found in many composite key tables. Due to the nature and composition of such relations, comparisons are often made between these types of relations and ternary relationships. N-ary relationships are formed from a semantic understanding of the data environment and the need for subsequent data retrieval. Decomposing ternary relationship involves testing preservation of functional dependencies, and considering all constraints are equally enforced to reflect the implicit dynamics of the ternary relationship. Normalization investigates the functional and multi-valued dependencies objectively. This is not a sufficient basis when dealing with ternary relationship. They concluded that attempting to compare the structure of higher normal forms with a ternary relationship is not correct.

Jones and Song (1996) demonstrated that certain binary relationship cardinalities are permitted within the ternary relationships during ER modeling by providing the comprehensive set of rules namely implicit binary cardinality rule, explicit binary permission rule and implicit binary override rule. The implicit binary cardinality rule states that in any given ternary relationship, regardless of ternary cardinality, the implicit cardinalities between any two entities must be considered M:N, provided that there are no explicit restrictions on the number of instances that can occur. The explicit binary permission rule states that for any given ternary relationship, a binary relationship cannot be imposed where the binary cardinality is less than the cardinality specified by the ternary for any specific entity. The implicit binary override rule states that given the imposition of a permitted binary relationship on a ternary relationship, the cardinality of the binary relationship determines the final binary cardinality between the two entities involved. It is
proved that the ternary cardinalities of M:N:P and M:N:1 can never have 1:1 relationship between any of the two entities involved.

Bock (1997) recalls in general that there are two types of errors in ER modeling namely incomplete data model error and mis-modeled problem domain error. When the system analysts are tasked to build a computer-based information system that is limited in scope, the incomplete data model error tends to occur. When the system analysts lack a complete understanding of the domain, problems such as modeling an attribute as single valued when it is multi-valued or mis-modeling the connectivity or degree of relationship, the mis-modeled error tends to occur. He illustrates that one of the errors is due to mis-modeled problem domain. Modeling a relationship as ternary that should have been modeled as two or more binary relationships and vice-versa will trigger aforesaid problems. He further suggested a verification procedure to check all ternary relationships for multi-valued dependencies and multi-valued attributes that are independent of one-another. To avoid ER modeling errors, one of the guidelines is to develop a sufficiently comprehensive conceptual model by gathering additional end-user views of the data and to model a slightly larger problem domain than that is required for the project.

McAllister and Sharpe et al (1998) presented a practical eight-step approach for analysis of n-ary relationships and decomposition into simpler relationships wherever appropriate since one of the most significant challenges in the use of ER data models is in deciding whether to use a single relationship among several entities or a set of simpler relationships to represent a complex association based on the relational database design concepts. By using two examples for N:N:1 ternary relationship cardinality, they demonstrated that one relationship must remain as ternary and other relationship can be losslessly decomposed into two binary relationships. They
used an extended specification of cardinality constraints to support the
decomposition approach and to ensure applicability to a variety of modeling
styles. A fundamental analysis skill for data modeling practitioner is defined
by them. Decision making is simplified by stating explicitly the questions
that must be analyzed during each step in the decomposition process. When
data analysts are able to make informed decisions about decomposition,
analysis of n-ary relationships is no longer perceived as a mysterious and
difficult skill.

Dey et al (1999) stated relationships involving more than two
entities are considered rare and have received less attention. Their research
provided a general framework for the analysis of relationships in which
common binary relationships become a special case. The framework helps a
designer to identify ternary and other higher-degree relationships that are
commonly represented, often inappropriately, as either entities or binary
relationships. Generalized rules are also provided for representing higher-
degree relationships in the relational model. They have demonstrated that the
selection of a higher-degree relationship or several binary degree relationships
is based on a real world environment and its implementation. They concluded
that their approach will assist the designer to extract more information from
real world environment and represent it accurately in a conceptual design.

Jones and Song (2000) in their work have provided an exhaustive
summary of ternary and binary combinations and lossless decomposition
strategies for the various cardinality combination outcomes. However, they
have concluded that fully equivalent decompositions do exist for certain
combinations of ternary or binary cardinalities, but the majority does not have
fully logical and practical equivalents. They reviewed the set of rules
established by them in their previous work (Jones and Song 1996) to identify
various ternary and binary combinations that can be combined simultaneously and investigated the application of these rules for decomposing ternary relationships to binary relationships and provided legitimate decomposition strategies by a construct called Constrained Ternary Decomposition (CTD) rule which governs the potential lossless decomposition of ternary relationships. The CTD rule states that any given ternary relationship cardinality can be losslessly decomposed to two binary relationships provided that at least one 1:1 or 1:M constraint has been explicitly imposed between any two of the participating entities. They demonstrated with a set of instances in the Ternary relationship 1:M:N with 1:M constraint and 1:1:1 with M:N constraint and provided the proof and identified certain ternary relationships have true, fully equivalent, binary equivalents of lossless decompositions, preservation of functional dependencies and finally the ability to preserve update constraints.

Camps (2002) has discussed that the transformation from n-ary relationships to a relational database schema has never been fully analyzed and has presented one of the several ternary cases ignored while transforming the ER model into relational model. The common belief, given a set of functional dependencies over a table resulting in a non-3NF situation, it is always possible to obtain a fully equivalent set of 3NF tables, without adding restrictions other than candidate keys and inclusion dependencies, is not actually true. If a correct database state has to be guaranteed, more complex restrictions have to be used.

Dullea et al (2003) explores the factors that contribute to validity of modeling structures within the ER diagrams. Their study has yielded a comprehensive set of decision rules namely, validity rules for recursive relationships, validity rules for ternary relationships, validity corollaries for
ternary relationships, validity rules for binary relationships and validity corollaries for binary relationships. They have stated that regardless of the complexity of the problem, these rules can be readily applied to real world database modeling and designing process or extended into case tool implementations. Both maximum and minimum cardinality constraints were used. They furnished the above set of rules based on impacts of constraining relationships, path connectivity and cardinality constraints to determine the structural validity of an ER diagram containing binary and ternary relationships.

Hartmann et al (2003) has presented a sound and complete system of inference rules for a class of generalized cardinality constraints containing both, participation constraints and Chen-style constraints. Cardinality constraints are those of the basic constituents of the entity-relationship approach to database design. When modeling the ER diagram, there is an extraordinary amount of disagreement among data modelers because of Chen notation and Merise notation. It is otherwise called look-across approach and look-here approach respectively. For binary relationship types, the difference between these two approaches can easily be translated to each other by exchanging the labels at the links. For ternary relationship types, however, the problem becomes more crucial since both types of the constraint namely Chen constraints and participation constraints are shown by look-across and look-here respectively. This can be avoided by restricting entity-relationship modeling to binary relationships types or by having lossless decomposition of an n-ary relationship type to binary relationship types or by transforming n-ary relationship type into an entity type and linked to its former components via n new binary relationship types. Their work justifies the simultaneous usage of two popular kinds of cardinality constraint
classes used in data modeling namely Chen-style constraint and participation constraint.

Badia (2004) revisited the ER models and investigated the ways in which ER models could be extended. To support more complex applications in database, it is necessary to capture more semantics in the conceptual model at the requirement specification phase. To explore enhancements and extensions to the ER model, research on relationships over relationships, attributes over attributes and relationships over attributes are suggested as promising approaches. The investigation concluded that no conceptual model will ever capture all the semantics of an application, and trade off between expressiveness and complexity is needed and revisiting ER model is often necessary.

Moody (2005) described some initial efforts towards developing a common standard for data model quality by addressing the theoretical and practical issues for establishing a conceptual model quality framework. He stated that international standards are available for evaluating the quality of software products but for measuring quality of conceptual models, equivalent standards are not available. The current practice of the evaluation of conceptual models is by nature a social rather than a technical process. To progress from an “art” to “engineering” discipline, quality standards need to be defined, agreed and applied in practice for conceptual modeling. For the progress of the field, it is argued the researchers and practitioners should work together to establish a common standard.

Teorey et al (2006) discussed all four cardinalities and functional dependencies about ternary relationships with examples. They explained the transformation of ternary relationship to relation is for all four cardinalities.
They have taken ternary relationship for M:N:P cardinalities between EMPLOYEE, SKILL and PROJECT and illustrated them with three types of business rules namely Skill_required, Skill_in_common and Skill_used with three different set of tables. All three ternary relationships namely Skill_required, Skill_in_common and Skill_used will be having the cardinality ratio of M:N:P. The business rule regarding relationship skill-required is defined as “An employee must have all the required skill needed for a project to work on that project. The business rule regarding the relationship skill-in-common is defined as “The employee must apply the intersection of his or her available skills with the skills needed to work on certain projects”. The business rule regarding the relationship skill-used is defined as “We can selectively record different skills that each employee applies to working on individuals projects”

They demonstrated that the Skill-required relation used for Skill-required ternary relationship may be decomposed by a two-way lossless decomposition when projected over (EMPLOYEE, SKILL) and (SKILL, PROJECT) not lossless when projected over (EMPLOYEE, SKILL) and (EMPLOYEE, PROJECT) and a three-way lossless decomposition occurs when projected over (EMPLOYEE, SKILL), (SKILL, PROJECT) and (EMPLOYEE, PROJECT). They demonstrated that the Skill_in_common relation used for Skill_in_common ternary relationship will have a three-way lossless decomposition when projected over (EMPLOYEE, SKILL), (SKILL, PROJECT) and (EMPLOYEE, PROJECT). They explained that the skill_used relation used for Skill-required ternary relationship cannot have lossless decomposed either by two way or three way decomposition. They stated by taking examples that a many-to-many-to-many ternary relationship is BCNF if it can be replaced by two binary relationships, 4NF if it can only
be replaced by three binary relationships and 5NF if it cannot be replaced in any way and thus is a true ternary relationship. Based on the other literature available we infer that Teorey et al (2006)’s concept may not suit for real world scenarios.

Hoffer et al (2008) stated that a ternary relationship is not equivalent to three binary relationships and recommended to convert all ternary or higher relationships to associative entities. They stated that by converting to an associative entity, it is possible to accurately represent the participation constraints. Further, in many ER diagramming tools and case tools, it is not possible to represent ternary relationship. Although not semantically accurate these tools must be used to represent ternary relationship with an associative entity and three binary relationships which have cardinality as a mandatory association with each of the related entity types. If it is to be drawn as three binary relationships, it is stated that names of the binary relationships should not be shown.

Genero et al (2008) in their work have defined a set of metrics to measure the structural properties of ER diagrams. They serve as indicators of the understandability of the diagram which is a key factor in determining the maintainability. In the highly dynamic business environment that we experience these days, database and data model evolution involve significant problems since conceptual data models which constitute the foundation of database design should be sufficiently flexible to incorporate changes easily and smoothly and it is a main concern of information managers. To support the rapidly changing data requirement and incorporate changes in the conceptual data model, it is necessary to understand the factors that drive the maintainability of conceptual model. They defined twelve metrics for various ER modeling constructs to determine ER diagram’s structural complexity.
These metrics are namely, the number of entities, the number of attributes, the number of derived attributes, the number of composite attributes, the number of multi-valued attributes, the number of relationships, the number of M:N relationships, the number of 1:N relationships, the number of n-ary relationships, the number of binary relationships, the number of reflexive relationships and the number of IS_A relationships.

Cuadra et al (2010) in their work focused on errors made in the design of ternary relationships like cardinality mistakes, representing a ternary relationship by means of binary relationships, lack of one or more attributes in the relationships and existence of redundancies. In conceptual modeling, the abstraction process creates some misunderstandings for novice designers, such as difficulties in modeling some constructs and in understanding the semantics that are present. Conceptual model must be able to provide direct mapping without distortion between the perceived real world and its representation. To aid the novice designer, they proposed six heuristics, two to avoid cardinality mistakes, one to avoid representation with binary relationships, one to avoid lack of one or more attributes in the relationships and two to avoid existence of redundancies. They concluded that the proposed heuristics help the designer in two activities of abstraction process namely the detection and the use of ternary relationships.

Elmasri and Navathe (2010) stated that the information represented by a ternary relationship type is different from the information represented by three binary relationship types. They explained that the representation of ternary relationship may be shown in different ways: as a weak entity, as an associate entity and three binary relationships. Many database design tools permit only binary relationships. In this case, ternary relationship must be represented as a weak entity type with no partial key and with three
identifying relationships. The three participating entity types are owner entity
types. An entity in the weak entity type is identified by the combination of its
three owner entities. It is also possible to represent the ternary relationship as
a regular entity type by introducing an artificial or surrogate key. A surrogate
key could be used for this new entity type and for converting it into a regular
entity types. Three binary 1:N relationships relate ternary relationship to the
three participating entity types. They conclude that “It is often tricky to decide
whether a particular relationship should be represented as a relationship type
of degree n or should be broken down into several relationship types of
smaller degree”. The designer must base this decision on the semantics or
meaning of the particular situation being presented. The typical solution is to
include the ternary relationship plus one or more of the binary relationships, if
they represent different meanings and if all are needed for application. The
schema designer must analyze the meaning of each specific situation to
decide which of the binary and ternary relationship types are needed.

The existence of three relationship instances in three binary
relationship types does not necessarily imply a corresponding existence of
instance in ternary relationship, because the meaning is different. They stated
that although in general three binary relationships cannot replace a ternary
relationship, under certain additional constraints it is possible to replace. They
provided a single guideline saying that a ternary relationship can be
decomposed if it contains an embedded binary relationship with a 1:1
cardinality.

Cuadra et al (2012) in their work provided a methodological
framework to inspire the database designer to use ternary relationship, the
constraint which database designers find it very difficult to detect, represent
and manage according to the domain requirements. The three approaches are
taken namely Chen’s approach, Merise’s method and their proposed strategy consisting of a combination of these approaches. Calculation of cardinality constraints in binary and ternary relationships is shown in their method. Based on the study, it is proved that their approach has a high level of participant confidence.

Shoval et al (2004) proposed a method for creating a Hierarchy of Entity-Relationship Diagrams (HERD) to large-scale applications. In case of a large-scale application, end users and managers find it difficult to use ERD due to increase in size and complexity since the comprehensibility and maintainability of the specification degrades rapidly. They stated that clustering an existing ER diagram method has a major drawback because the entity has many relationships which generate many diagrams at the end of the process, one for each refinement. Furthermore, there is no limitation with respect to the participation of an entity or an entity cluster in the different refinements, and therefore there is no way to find all the locations of an entity or an entity cluster. Thus, the data model can become very complex and complicate maintainability of the model. Therefore, a mechanism is proposed to improve ERD comprehensibility, to simplify its maintainability and effectively apply the ER model to large-scale applications. Their method is based on packaging operations which group entities and relationships according to certain criteria. They performed a comparative study about the order of learning and time of learning. With respect to order of learning, they stated that flat ER is good and with respect to time of learning, they stated more time on learning and exercising with flat ER diagrams is required. They had mainly viewed user comprehension and their preference point.

Khaled et al (2004) have proposed a new construct to specify the business rules and constraints along with the systems requirements in
database conceptual modeling. Their proposed construct is capable of enhancing the modeling capabilities and expressiveness of entity-relationship model to incorporate attribute oriented business requirements and constraints. They discussed about analytical approach to define rules for attribute-oriented BRCs and to define construct and appropriate parameters to capture business requirements and constraints to integrate the new construct with existing ER diagram. They listed five categories of binary relationships between two entities as attributes of only one entity, attributes of both entities, attributes of the relationship itself, attributes of the relationship itself plus attributes of only one participating entities and attributes of the relationship itself plus at least one attribute of both participating entities. They stated that database designers can be benefitted with this construct in several ways as to specify various attribute-oriented business rules and constraints along with the systems requirements, to identify potential conflict between BRCs, and between BRCs and SRCs at an early analysis stage, consistent implementation of SRCs and BRCs with the design artifact, better understanding of BRCs by all stakeholders and the design decision at the analysis level could be revisited and re-consulted with at the implementation level if any confusion occurs since BCRs are readily available along with the conceptual model in their approach. According to them the issues that still remain unsolved are: Propagation of the defined BRC in conceptual model to the database implementation Handling BRCs for ternary relationship type Graphical notations to accommodate BRCs in ER and other approaches, Formalize BRCs as data model.

Dhabe et al (2010), in their paper proposed an Articulated Entity-Relationship (AER) diagram which is an extension of Entity-Relationship (ER) diagram to accommodate the functional dependency (FD) information as
its integral part for complete automation of normalization. As the proposed AER diagram is designed by taking into account the normalization process, normalization up to BCNF becomes an integral part of conceptual design. Any modifications made to the AER diagram will automatically be reflected in its FD information. FDs are diagrammatically represented using two types of connectors: attribute connectors and functional dependency connectors. They have shown complete AER diagram for banking enterprise. They concluded that AER diagrams could be extended to include multi-valued dependencies and join dependencies. They stated that it should be possible to the domain and key constraints to automate normalization up to DKNF.

De Lucia et al (2010) in their work aimed to analyze the comprehensibility ER diagrams and UML class diagrams in data model maintenance. They performed three sets of controlled experiments. They stated that the results demonstrate that using UML class diagrams achieved better comprehension levels and the support given by the two notations during maintenance activities are same, while in general UML class diagrams provide a better support during verification activities.

Martin (1994) has presented a technique for extracting an entity-relationship schema from a relational database through reverse engineering. His work is based on an analysis of data manipulation statements in the code of an application using a relational database management system. A database application typically includes data manipulation statements embedded in forms, reports and application code. Semantically related attributes in different relations are identified through the analysis of join clauses in the data manipulation statements of the application. Attributes representing references between relations in the relational schema and possible keys are determined by an analysis of join conditions in queries and view definitions.
Knowledge about which attributes link relations is used to investigate the database extension in a selective manner. The information resulting from this analysis is represented in a connection diagram with nodes representing relational schemas, and where an edge between two nodes $R_1$ and $R_2$ implies that there is a join condition involving attributes in $R_1$ and $R_2$.

Ramanathan and Hodges (1996) have presented an object-centered approach for schema mapping. The approach follows a procedure that maps a 3NF relational schema into an object-oriented schema without the explicit use of inclusion dependencies and thus provides a greater scope for automation. Their object-oriented approach carries out the mapping process using the information from the primary and foreign keys of the relations because, in most RDBMSs, the information about the keys can be extracted automatically from the database catalogue. Their procedure has three steps: First, identification of relations corresponding to object-classes, second, identifying all manner of relationships and finally establishment of cardinalities. User’s input is required to confirm certain extracted object-oriented constructs and to give names to discovered associations. The class diagrams are obtained through reverse engineering of relational schemas.

Petit et al (1996) show how an Entity-Relationship schema can be derived from a relational schema, in third normal form, provided with key constraints and referential integrity constraints. Inclusion dependencies are extracted by analyzing the equijoin queries embedded in application programs and by querying the database extension. They have used a set of algorithms for deriving an Entity-Relationship schema from a relational schema. The ‘INS Discovery algorithm’ discovers the interrelation dependencies between attributes of the relational schema after which the ‘LHS-Discovery algorithm’ computes the set of candidate left hand sides of the functional dependencies.
and the set of hidden objects. Discovery of the right hand side of a functional
dependency was carried out by extracting the clues from application programs
by the RHS-Discovery algorithm. The ‘Restruct algorithm’ gives the main
steps to restructure a 1NF relational schema given the set of inclusion
dependencies, and the set of functional dependencies. Finally, the ‘Translate
algorithm’ maps the restructured relational schema into Entity-Relationship
structures. Their method is interactive and dependent on an expert user who
has to validate the presumptions on the elicited dependencies.

An et al (2010) presented a round-trip engineering framework and a
set of principles and procedures that automatically and incrementally maintain
conceptual-relational mappings as schemas. The first principle for mapping
maintenance under schema evolution is to locate the appropriate elements in
the conceptual model for adding new attributes. The location process is
guided by analyzing the key and foreign key information in the original and
new schemas. The second principle is to locate the anchors of the appropriate
skeleton trees for discovering or adding relationships. The location process is
guided by using key and foreign key structures in the schemas. The third
principle is to arrange the primary key and foreign key constraints in the
(new) schema with the cardinality constraints in the new conceptual model.
The authors also propose two procedures for maintaining the mappings. The
first procedure maintains the mappings when schemas evolve. Given a set of
consistent conceptual-relational mappings as input, the procedure gives a
synchronized conceptual model and a set of updated mappings as output. The
second procedure obtains a set of consistent conceptual-relational mappings
between a conceptual model and a relational schema as input and gives an
updated new set of mappings. The conceptual model obtained can be
considered equivalent to ER model.
2.2 DATABASE NORMALIZATION

The process of normalization was first formalized by Codd (1970). Codd (1970) in his classic paper has introduced the relational model. Functional dependencies were originally introduced by Codd (1970). Normalization is performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form. Three normal forms called first (1NF), second (2NF), and third (3NF) normal forms were initially proposed. The original definitions of first, second, and third normal form were also defined in Codd (1972), where a discussion on update anomalies can be found. An amendment was later added by Codd (1974) to the third normal form called Boyce-Codd Normal Form (BCNF).

Fagin (1977) has introduced “multivalued dependencies” which are a generalization of the well-known functional dependencies for relational databases. It significantly extends the understanding of the logical design of the relational database. This concept of multivalued dependencies leads to the “fourth” normal form and it is strictly stronger than Codd’s “improved, stronger version of third normal form” or “Boyce-Codd normal form”. It has been proved that multivalued dependencies provide a necessary and sufficient condition for a relation to be decomposable into two of its projections without loss of information.

Rissanen (1977) stated that in a multiattribute relation or, equivalently, a multicolumn table a certain collection of the projections can be shown to be independent in much the same way as the factors in a Cartesian product or orthogonal components of a vector. In this concept, a precise notion of independence for relations is defined and studied by him. The main result states that the operator which reconstructs the original relation from its
independent components is the natural join, and that independent components split the full family of functional dependencies into corresponding component families. These give an easy-to-check criterion for independence.

Fagin (1979) has discussed the relationship between normal forms in a relational database and an allowed set of relational operators. Project-join normal form (PJ/NF) was defined when only projection and join are allowed. It was noted that PJ/NF could logically be called as “fifth normal normal form”, since it is stronger than fourth normal form.

Codd (1979) discussed extending the relational model to incorporate more meta-data and semantics about the relations. The result is a model with a richer variety of objects than the original relational model, additional insert-update-delete rules, and some additional operators that make the algebra more powerful. A data model should act as a conceptual framework for defining a wide class of formatted databases and a mediator between stored representations and user views. It should probably have at least four personalities: a tabular personality (e.g., the extensions of relations in the relational model), a set-theoretic personality (e.g., the relational algebra), an inferential string-formula personality (e.g., predicate logic in modern notation), and a graph-theoretic personality (e.g., labeled, directed hypergraphs for relations). He also proposed a three-valued logic to deal with uncertainty in relations and incorporating NULLs in the relation algebra.

Aho et al (1979) have proposed efficient algorithms to determine whether the join of several relations has the intuitively expected value of lossless or has a subset with a lossy join. These algorithms will be useful to test lossless join decomposition in the presence of functional and multivalued
dependencies are giving the expected value to answer queries in a relational database.


Boyce-Codd normal form was defined in Codd (1974). The alternative definition of third normal form is given in Ullman (1988), as is the definition of BCNF. Many of the theorems and proofs concerning relational database design algorithms, functional dependencies and other dependencies are explained in Ullman (1988), Maier (1983), and Atzeni and De Antonellis (1993).

The theory of dependency preservation and lossless joins is given in Ullman (1988), and the lossless join property is analyzed in Aho et al. (1979).

Armstrong and Deobel (1980) made a general study of two basic integrity constraints on relations: functional and multivalued dependencies. The concept of multivalued dependencies are studied by an equivalent concept of decompositions. A model is constructed for any possible combination of functional dependencies and decompositions. The model embodies some decomposition as unions of relations having different schemata of functional dependencies. This suggests a new, stronger integrity constraint, the degenerate decomposition. More generally, the theory demonstrates the importance of using the union operation in database design and of allowing different schemata on the operands of a union. Techniques
based on the union lead to a method for solving the problem of membership of decomposition in the closure of a given set of functional dependencies and decompositions. The concept of antiroot is introduced as a tool for describing families of decompositions, and its fundamental importance for database design is indicated.

Fagin (1981) has defined a normal form, called domain-key normal form (DK/NF), which is based only on the primitive concepts of domain and key, along with the general concept of a “constraint”, unlike previously defined normal forms in terms of functional, multivalued, or join dependencies. A formal definition of an insertion anomaly and a deletion anomaly are presented. He stated that definition of “anomaly” is different from that of Codd. He has shown that a satisfiable relation schema is in DK/NF if and only if it has no insertion and deletion anomalies. Unlike previous normal form definitions, the definition of DK/NF is operator free.

Zaniolo (1982) is a source for simpler definitions of 3NF and BCNF. In this work, definition of another normal form, elementary key normal form (EKNF) which is between 3NF and BCNF and possesses some of the better qualities of both is given. This new normal form is well suited for designing schemata according to the representation principle. This work also proved that Bernstein’s schema design algorithm (Bernstein 1976) produces schemata that satisfy this new definition. The new normal form is developed in the framework of functional dependencies.

Kent (1983) presented the guidelines corresponding to first through fifth normal forms that do not require an understanding of relational theory to emphasize their generality and also to make them easier to understand. They stated that their presentation conveys an intuitive sense of the intended
constraints on record design, although in its informality it may be imprecise in some technical details. The normalization rules are designed to prevent update anomalies and data inconsistencies. With respect to performance tradeoffs, these guidelines are biased towards the assumption that all non-key fields will be updated frequently.

Vincent (1997) made an investigation about the adequacy of fifth normal form in relational database design and its relation to PJNF. It is demonstrated that every attribute is a candidate key and forms a stringent condition and it is impossible to attain in practical database design and it also fails to generalize 4NF. As stated in the 5NF definition, all derived JD's including those with removable components should satisfy the condition that every component is a super key. Vincent has defined corrected 5NF called as 5NFR (Reduced - 5NF) by relaxing the 5NF stating that only derived JD's which are strongly reduced in which no removable components are permitted satisfy the condition that all of their components are superkeys which also has the desired property of generalizing fourth normal form. The relationships between PJNF, 5NF and 5NFR are investigated and concluded that 5NF is strictly in a stronger condition than PJNF and that PJNF is strictly stronger than 5NFR.

Levene and Vincent (2000) identified three problems when designing databases in the presence of FDs and Inclusion Dependencies (INDs), apart from update anomalies and redundancy problems. The notions of entity integrity and keys are generalized by functional dependencies and the notion of referential integrity and foreign keys are generalized by inclusion dependencies. The semantic justification of the normal forms in the presence of FDs is well understood, but little research has been carried out and the important of INDs as integrity constraints. Hence three problems are
identified when designing databases in the presence of FDs and INDs. The first problem is the attribute dependency; second problem is potential violation of entity integrity when doing insertion operation and the third problem is avoiding the interaction between FDs and INDs. To solve these entire problems, they proposed IDNF (Inclusion Dependencies Normal Form). IDNF is justified as a robust normal form that eliminates both redundancy and update anomalies from the database schema.

Connolly and Begg (2008) synthesize two general perspectives for normalization, namely, normalization as a bottom-up approach to create set of relations for database design and normalization as a top-down approach used as a validation technique to check the structure of relations which are created using ER modeling. The opportunity to use normalization as a bottom-up technique is limited by the level of detail that the database designer has to manage. This limitation is not applicable when normalization is used as a validation technique as a database designer focuses on only part of the database, such as a single relation, at any one time. Normalization can be usefully applied irrespective of the size or complexity.

According to Stephens and Plew (2001) normalization is the process of reducing the redundancy of data in relational database. Although in most cases redundant data is not completely eliminated, the greater the reduction of redundancies, the easier data integrity is to maintain. Normalization improves on the structure of a database, but unfortunately, has consequences. Performance degradation is inevitable when normalizing database.

Halpin (2001) argues that using normalization as a check on conceptual modeling may at times be helpful, but use of normalization alone
as a design method would be inadequate. He states that normalization theorists have been able to rigorously prove several interesting results, by limiting the scope of normalization to a small but important class of constraints and ignoring semantic transformation.

Date (2001) provides a retrospective review and analysis of the relational data model on a historical account and assessment of E.F. Codd’s contribution to the field of database technology from 1969 to 1979. Several of Codd's original papers have since become hard to find, and some of his writings were somewhat difficult to read and understand. This book aims to clearly evaluate Codd's original ideas and relate them to today's database society.

Khodorovskii (2002) introduced a new normal form, namely Object Normal Form. He stated that the existing definition of the fifth normal form is unsatisfactory and the correct definition is given for the first time. For improving the representation of data in a database with a 5NF schema, the notion of negating relation is introduced. It is shown that the standard order of steps of the normalization procedure should be changed; first, the specific requirements of the fourth normal form should be satisfied and only then the requirement of the second normal form and so forth. Maier (1983), more or less accurately discussing the 5NF theory, gives two definitions of the form. The first definition states that, a schema R of a relation r(R) is in the 5NF if, for every join dependency *[R_1, R_2, ..., R_p] in the relation r(R), either the dependency is trivial or every Ri is a superkey of the relation r(R). Then, Maier (1983) gives another “revised” definition. This is, in fact, the first definition of the 5NF proposed by Fagin (1977). A schema R of a relation r(R) is in the 5NF if every join dependency *[R_1, R_2, ..., R_p] in r(R) is deducible from key functional dependencies in r(R). Satisfactory definition of
fifth normal form. A schema $R$ of a relation $r(R)$ is in the 5NF if, for any irredundant join dependency $*[R_1, R_2, \ldots, R_p]$ in the relation $r(R)$, every $R_i$, $i = 1, 2, \ldots, p$, is a superkey of the relation $r(R)$. A database schema $R$ is in the 5NF if every schema $R$ from $R$ is in the 5NF.

Date et al (2003) defined sixth normal form as a normal form for databases based on an extension of relational algebra. In this work, the relational operators, such as join, are generalized to support a natural treatment of interval data, such as sequences of dates or moments in time. Sixth normal form is then based on this generalized join as: A relvar $R$ is in sixth normal form if and only if it satisfies no nontrivial join dependencies at all. A join dependency is trivial if and only if at least one of the projections involved is taken over the set of all attributes of the relvar concerned. Any relation in 6NF is also in 5NF. It is stated that sixth normal form is intended to decompose relation variables to irreducible components may be relatively unimportant for non-temporal relation variables and can be important when dealing with temporal variables or other interval data.

There is no universal agreement as to which features would disqualify a table from being in 1NF. Most notably 1NF, as defined by some authors, Elmasri and Navathe (2003) excludes relation-valued attributes (tables within tables); whereas 1NF as defined by other authors and Date permits them. According to Date’s definition of 1NF, a table is in 1NF if and only if it is "isomorphic to some relation" (2003). Hugh Darwen and Date (1992, 2003) have suggested that Codd's concept of an "atomic value" is ambiguous and that this ambiguity has led to widespread confusion about how 1NF should be understood. Date (2003) suggests that the notion of atomicity has no absolute meaning and a value may be considered atomic for some purposes, but may be considered an assembly of more basic elements
for other purposes. If this position is accepted, 1NF cannot be defined with reference to atomicity. Columns of any conceivable data type whether it is from string types, numeric types, array types and table types are acceptable in an 1NF table although not always desirable. Date (2003) argues that relation-valued attributes by means of which a field within a table can contain a table are useful in rare cases.

Kung et al (2008) proposed a simple and straightforward normalization algorithm, the 5C data modeling method to assist the sophisticated end-users to develop their own database in a better way without understanding technical terminology that may confuse the user-developer. They stated that end-users frequently experience difficulties in producing quality databases since they do not possess sound data modeling training or skills. The proposed method focuses on easy-to-follow steps and eliminates references to technical terminology. Count, Comply, Compare, Consolidate and Convert are the representations of 5C in the 5C data modeling method. The assumptions made are that universal relation covers a single business process domain; universal relation is in the first normal form (1NF) and (3) the set of functional dependencies are given and are in closure. Using normalization and ERD (entity-relationship diagram) and making 5 steps they created this method. The Steps are Count Number of LHS and RHS attributes of a functional dependency (FD), Keep LHS attributes intact, Compare the no. of LHS and RHS attributes, Consolidate RHS attributes, and Convert the FD to relations. Simple and straightforward teaching of bottom-up database design can be achieved by this method.

Chen et al (2011) reported the problems of functional entanglements and have given the solution to eliminate some basic data anomalies on account of functional entanglements. By identifying functional
entanglement in a database it is stated that practitioners can greatly improve data quality. They introduced two methods to identify functional entanglements by detecting sub domain dependency and restricted domains. Two methods of eliminating functional entanglements at the design level in a normalized database namely, field-level disentanglement and horizontal decomposition are examined. Three other practical approaches for restricting the effects of functional entanglements with RDBMS tools are analyzed as building lookup relations, utilizing check constraints, and implementing database triggers. Based on the strengths and shortcomings, it is stated that practitioners should carefully evaluate the requirements at hand and apply the most appropriate methods to deal with potential data anomalies.

Delobel and Adiba (1985) have stated that in the set of unirelational dependencies, around two-thirds correspond to functional dependencies (FDs) and one-quarter correspond to multivalued dependencies (MVDs) in practice. Wu (1992) has stated that MVDs are frequently exhibited in database applications. Hartmann and Link (2012) have investigated the implication problem for classes of data dependencies over SQL table definitions. They have established an axiomatization and algorithms to decide the implication problem for the combined class of functional and multivalued dependencies in the presence of NOT NULL constraints and have shown that arbitrary null-free subschemata (NFS) has a significant impact on the implication problem of the combined class of FDs and MVDs. They have established NOT NULL constraint as an effective mechanism to balance the expressiveness and efficiency of entailment relations for significant classes of unirelational dependencies that arise in practice.

Date (2012) has given definition of 6NF that works for “regular”, nontemporal-data as: A relvar R is in sixth normal form if and only if the only
JDs that hold in R are trivial ones. In other words, the only JDs that hold in R are of the form *{.....,H,....}, where H is the heading. He has stated that we can never get off trivial dependencies; thus, a relvar in 6NF cannot be nonloss decomposed at all, other than trivially. 6NF relvar is sometimes said to be irreducible and that every 6NF relvar is certainly in 5NF and 6NF is always achievable.

Darwen et al (2012) introduced a new normal form, called Essential Tuple Normal Form (ETNF) which is strictly between fourth normal form and fifth normal form. Normalization can be described as a process that leads to a design in which the database schema is redundancy-free. In this work, they defined the meaning for a tuple in a relation to be redundancy-free, or “essential”; and defined the normal form ETNF where every tuple is essential. They showed it is exactly as effective as 5NF in eliminating redundancy of tuples. They argued that if the goal of the database designer is to prevent redundancy of tuples, ETNF is necessary and sufficient for the purpose. They provided a syntactic characterization of ETNF, which says that a relation schema is in ETNF if and only if it is in Boyce-Codd Normal Form and some component of every explicitly declared join dependency of the schema is a super-key. They showed the relationship between ETNF and other normal forms in the literature that are strictly between 4NF and 5NF, namely Super-Key Normal Form (SKNF) and Redundancy-Free Normal Form (RFNF). They stated that the normal form given by erroneous definition of 5NF including Date (2004) is called SKNF by Normann (1998) and Reduced Fifth Normal Form (5NFR) by Vincent (1997). Date (2004) 5NF says that every component of every irreducible Join Dependency (JD) is a super-key. They also stated syntactic characterization of ETNF is similar to
Vincent (1995) RFNF. They proved that $5N \Rightarrow SKNF \Rightarrow RFNF \Rightarrow ETNF \Rightarrow 4NF$ and that none of the reverse implications hold.

2.3 REFACTORING TECHNIQUES

Fowler et al (1999) made an invaluable contribution to object oriented software development by shedding lights on the refactoring process. They explained the principles and best practices of refactoring. They also stated when to apply and not to apply refactoring to the source code but failed to state where to apply refactoring and complicated this by stating that refactoring are based on human intuition and on subjective perceptions. They further stated that no set of metrics can challenge informed human intuitions. They specified what kind of source code should be restructured and proposed the concept of bad smells. They proposed and described 22 bad smells in object-oriented systems. They also associated refactoring rules with these bad smells, suggesting how to resolve these bad smells. According to them the main risk of refactoring is that existing working code may break due to the changes being made. They concluded by suggesting two golden rules namely, refactor in small steps and have test scripts available to existing functionality to mitigate this risk. As a noun refactoring means a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior. As a verb refactor means restructuring software by applying a series of refactorings without changing its observable behavior.

Simon et al (2001) in their work proposed a metric-based visualization tool to support the identification of source code components that needs refactoring. They defined a distance-based cohesion metric which measures the cohesion between attributes and methods. The calculated
distances are visualized in a three dimensional perspective. They showed that these special kinds of metrics can support the subjective perceptions and thus can be used to get support for the decision, what technique of refactoring is to be applied and where it can be deployed in an effective and efficient way. In their work they demonstrated the concept for four usual refactorings namely move class, move attribute, extract class and inline class by presenting a tool supporting the identification and explaining the real world application with this concept. The drawback of Simon’s approach is that the calculated distances are visualized in a three-dimensional perspective, forcing the developer to manually identify refactoring opportunities. So the visual interpretation of distance in large systems can be a difficult and subjective task. Furthermore the approach proposed by Simon et al., did not suggest any restructuring automatically; it just helps the developer to identify the restructuring opportunities.

Kamiya et al (2002) devised a clone detection algorithm and implemented a tool named CCFinder (Code clone finder) which focuses on analyzing large-scale systems with a limited amount of language dependence. A code clone is a code portion in source files that is identical or similar to another. CCFinder aims to effectively identify code portions of renaming variables or editing pasted code after copy-and-paste which makes a slightly different pair of code portions. The tool makes a token sequence form the input code through a lexical analyzer, and applies the rule-based transformation to the sequence. The purpose is to transform code portions into a regular form to detect clone code portions that have different syntax but have similar meaning.

Tahvildari and Kontogiannis et al (2003), in their work related to enhance design quality through meta-pattern transformation, have proposed a
framework where a catalogue of object oriented metric is used as an indicator for automatically detecting where a particular transformation can be applied so as to improve the design quality of the system. They have used an object-oriented metrics suite consisting of complexity, coupling, and cohesion metrics to detect classes where a particular transformation can be applied to improve the quality of the system. In particular, they identified the design flaws at class level by quality design heuristics namely key classes and one class-one concept and these heuristics lies at the extremities. They have also categorized the design flaws into three categories namely architectural, behaviour and structural. By assessing which classes of the system exhibit problematic metric values they select an appropriate meta-pattern that will potentially improve the corresponding metric values.

They have established a cause effect relationship between some combination of metrics and poor design quality. This approach indicates the kind of the required transformation but does not specify on which specific methods or classes this transformation should be applied. This process requires human interpretation. Moreover, in case of multiple potential suggestions, the approach does not evaluate their effect in order to rank. They have applied the re-engineering strategy for design flaws on Java Expert System Shell (JESS). They have identified six meta-patterns namely abstraction, extension, movement, encapsulation, buildrelation and wrapper and their effect on the metric values. They have increased the value of TCC in order to decrease the value of DAC and RFC by applying the meta-pattern transformation called WRAPPER. WRAPPER wraps an existing receiver class with another class in such a way that all requests to an object of the wrapper class are passed to the receiver object it wraps and similarly any results of such requests are passed back by the wrapper.
Du Bois et al (2004) proposed a concept of refactoring with metric oriented approach, which helps to improve metrics rather than apply them as an identification criterion. The best and worst-case impact of refactoring on coupling and cohesion dimensions are theoretically analyzed. The refactoring, they studied are extract method, move method, replace method with method object, replace data value with object method, and extract class method. They demonstrated that by exploiting the results from coupling or cohesion impact analysis, it is possible to achieve quality improvements with restricted refactoring efforts. This effort is restricted to the analysis and resolution of a limited set of refactoring opportunities which are known to improve the associated quality attributes.

Mens and Tourwe (2004) provided an extensive summary of existing research in the field of software refactoring. Their research compares and discusses different factors like refactoring activities that are supported, the specific techniques and formalisms that are used for supporting these activities, the types of software artifacts that are being refactored, the important issues that need to be taken into account when building refactoring tool support, and the effect of refactoring on the software process. A running example that illustrates a typical nontrivial refactoring of an object-oriented design is used throughout this work to explain and demonstrate the main concepts. It is stated that the most widespread approach to detect program parts that require refactoring is the identification of bad smells. They identified a need for tools that address refactoring in a more consistent, generic, scalable, and flexible way.

Gui and Scott (2006) provided new measures of coupling and cohesion developed to assess the reusability of Java components retrieved from the Internet by a search engine. They stated that these measures differ
from the majority of established metrics in two respects: they reflect the
degree to which entities are coupled or resemble each other, and they take
account of indirect couplings or similarities. An empirical comparison of the
new measures of their work with eight established metrics, Coupling Between
Objects (CBO), Response For a Class (RFC), Coupling Factor (CF), Data
Abstraction Coupling (DAC), Lack of Cohesion in Method LCOM, Lack of
Cohesion in Method variant 3 LCOM3, Runtime Lack of Cohesion in
Methods (RLCOM) and, Tight Class Cohesion (TCC) show the new measures
are consistently superior according to their reusability.

Van Rompaey et al (2007) have proposed a technique like fine-
grained defect detection technique. Accordingly, test coevolution forms an
additional burden on the software developer which can be tempered by
writing tests in a manner that makes them easier to change. So that they
concretely express what a good test is by exploiting the specific principles
underlying unit testing. Analogous to the concept of code smells, violations of
these principles are termed test smells. They clarified the structural
deficiencies encapsulated in test smells by formalizing core test concepts and
their characteristics. To support the detection of two such test smells, general
fixture and eager test, they proposed a set of metrics defined in terms of unit
test concepts. They compared their detection effectiveness using manual
inspection and through human reviewing. Their results indicate that human
reviewing is not a reliable means for test smell detection. This work stressed
the need for a more reliable detection mechanism and provided an initial
contribution through the validation of test smell metrics.

Joshi and Joshi (2009) presented a method based on concept
analysis which aims to identify less cohesive classes. Concept analysis for
class cohesion has come up with the concept called cohesion lattices that
captures the cohesiveness of the class and its members. Most metric based approaches for cohesion analysis compute a single value for the entire class and do not help to identify individual members responsible for the lack of cohesion. But the approach of Joshi et al. helps to identify the class members contributing to lack of cohesion. The concept based cohesion analysis gives a quick visual overview of class cohesion and helps guiding refactoring in localization of attributes, removal of instant variables and class extraction.

Fokaefs et al (2009) proposed a methodology which identifies extract class refactoring opportunities by a class decomposition method. An agglomerative clustering algorithm is used based on the Jaccard distance between class members. In terms of cohesion this work facilitates to identify new concepts and rank the solutions according to their impact on the design quality of the system. Specific kind of bad smells called “God Class” is considered for this purpose. Data god class and behavioral god class are defined. A class which has many system’s data in terms of number of attributes is called data god class. When it has greater portion of the systems functionality in terms of number and complexity of methods is called behavioral god class. Behavioral god class may be avoided by splitting the class by extracting a cohesive and independent piece of functionality. This refactoring is called “Extract Class”. Two projects namely eRisk, an electronic adaptation of the well known board game and SelfPlanner Refanidis et al (2008), an intelligent web based calendar application have been taken for this purpose and the result shows that this methodology identifies relatively large number of new concepts that can be potentially extracted in new classes.

Tsantalis and Chatzigeorgiou et al (2009) have proposed the placement of attributes or methods within classes in an object-oriented system
which is usually guided by conceptual criteria and aided by appropriate metrics. Moving state and behaviour between classes can help reduce coupling and increase cohesion, but it is nontrivial to identify. They proposed a methodology for the identification of move method refactoring opportunities that constitute a way for solving many common feature envy bad smells.

They generated an algorithm that employed the notion of distance between system entities (attributes or methods) and classes to extract a list of behaviour-preserving refactorings based on the examination of a set of preconditions. In practice, a software system may exhibit such problems in many different places. So, they measured the effect of all refactoring suggestions based on a novel entity placement metric that quantifies how well entities have been placed in system classes. They stated that the methodology could be regarded as a semi-automatic approach since the designer will eventually decide whether a suggested refactoring should be applied or not based on conceptual or other design quality criteria.

To perform clustering with different attribute weights, Alkhalid et al (2010) proposed Adaptive K-Nearest Neighbor (A-KNN) algorithm to facilitate the software developers in refactoring at the function or method level. Improvements of cohesion for ill-structured software entities are suggested by them. Software entities are code statements that need to be grouped. There are two types: executable and non-executable statements. Comments and declarations, that do not affect the functionality of the function; therefore, they are not used as entities. They are non-executable statements. The statements which include assignment, operation, and iteration statements are the executable statements. This A-KNN is compared with three other function-level clustering techniques like Single Linkage algorithm.
(SLINK), Complete Linkage algorithm (CLINK) and Weighted Pair-Group Method using Arithmetic averages (WPGMA) to prove the improved performance of proposed system. Each entity has properties, based on which the similarity matrix is constructed. A-KNN reduces the amount of required computations by using this similarity matrix. Code restructuring at class and package levels are not discussed in this work.

Bavota et al (2010) have proposed an approach using game theory to identify extract-class refactoring opportunities. Game theory techniques can often be used to deal contrasting goals. Game theory is successfully used to propose solutions to strategic situations, in which an individual’s success in making choices depends on the choices of others. It is very common in software engineering to find a solution often for problems having competing goals, like integrity versus efficiency, reusability versus reliability, reusability versus integrity, quality versus cost, cohesion versus coupling by the developers and managers. The extract-class refactoring problem can be modeled as a non-cooperative game involving two players. Given a class to be refactored, the two players contend with the methods of the original class to build two new classes with higher cohesion and lower coupling than the original class. Their approach considers S and T as two players in charge of building a new class selecting methods from the original class to be refactored. The process starts by assigning to S and T two methods that have the lowest structural and semantic similarity. S and T will then iteratively contend with the remaining methods of the class to be refactored. In single iteration, S or T could add at least one method to its class by considering the impact of adding the method on the cohesion and coupling of its class. The process stops when each method of the original class is assigned to either S or
T. The move to be performed during iteration is chosen by finding the Nash equilibrium in the payoff matrix.

Moha et al (2010) proposed three contributions to the research area related to code and design smells namely DÉCOR, a method that embodies and defines all the steps necessary for the specification and detection of code and design smells, DETEX, a detection technique that instantiates this method, and an empirical validation in terms of precision and recall of DETEX. Using DETEX, they specified four well-known design smells namely the antipatterns Blob, Functional Decomposition, Spaghetti Code, and Swiss Army Knife, and their 15 underlying code smells, and they automatically generated detection algorithms.

Bavota et al (2011) have presented an approach to improve class cohesion. To identify extract class refactoring opportunities, they have used the structural and semantic relationships between methods. Their approach takes a low cohesive class as input and parses the input class to extract a weighted graph representation of the class. The methods of the class form the nodes of the graph and the edges of the graph represent the interaction between the pair of methods that are connected by the edge. They have used MaxFlow-MinCut algorithm to obtain the partition of the original graph into two sub-graphs, by cutting the minimum number of edges with a low weight. The two sub-graphs can be used to create two classes that have cohesion higher than the original class without increasing too much coupling. Their implementation has not included classes belonging to hierarchies as splitting the descendant in a hierarchy might cause compilation error or change in the behaviour.
Liu et al (2012) proposed the schedule and resolution technique to reduce the human effort. When they resolve duplicated code and feature envy first, they found that once duplicated code and feature envy are cleaned, long method disappears automatically and the length of the method was reduced considerably. They identified the importance of resolution sequences of bad smells. They validated the effect of resolution sequences of bad smells on two nontrivial applications. Their evaluation results suggest that a significant reduction in effort ranging from 17.64 to 20 percent can be achieved when bad smells are detected and resolved using their proposed sequence.

Liu et al (2013) have proposed that software refactoring is an effective method for improvement of software quality while software external behaviour remains unchanged. They stated that software engineers with little experience in software refactoring might ignore a number of potential refactorings or may conduct refactorings later than expected and result in poor software quality, incur higher refactoring cost. They proposed a monitor-based instant refactoring framework to drive inexperienced software engineers to conduct more refactorings on time. In their work, changes in the source code are instantly analyzed by a monitor running in the background. If these changes have the potential to introduce code smells, i.e., signs of potential problems in the code that might require refactorings, the monitor invokes corresponding smell detection tools, and warns developers to resolve detected smells promptly. Feedback from developers, regarding acknowledgement of detected smells and resolution, is consequently used to optimize smell detection algorithms. Their proposed framework has been implemented, evaluated, and compared with the traditional human-driven refactoring tools. Evaluation results suggest that the proposed framework could drive inexperienced engineers to resolve more code smells promptly.
The average duration of resolved smells was reduced by 92 percent. Results also suggest that the proposed framework could help developers to avoid similar code smells through timely warnings at the early stages of software development, thus reducing the total number of code smells by 51 percent.

ISO 9126 proposes a standard which species six quality factors on areas of importance, for software evaluation as functionality, reliability, usability, efficiency, maintainability, and portability. Understandability is placed as a factor of software usability. Understandability is defined as a set of attributes of software that relate to the users’ effort for recognizing the logical concept and its applicability. Other factors of usability are learnability, operatability and compliance.

Boehm (1978) defined software understandability as a characteristic of software quality which means ease of understanding software systems (Boehm et al 1978). In his model, understandability is placed as a factor of software maintenance. Changes to software systems are called software evolution in the research field of software maintenance. Software understandability can be placed as a factor of software evolution in reuse or maintenance. In an experiment of code inspection, 60% of issues which professional reviewers reported were soft maintenance issues related to understandability (Porter 1997).

Grechanik and Perry (2004) stated that a significant number of failures of software projects are due to poor requirements gathering, errors in specifications, choosing an incorrect architecture, wrong design and development model, and incurring significant cost in the maintenance stage. Further they are based on an assumption that everyone involved in a software project is driven to make it successful. However, each team participant
views the project differently in terms of his or her personal goals. These different views may result in conflicting decisions by team participants that affect the overall success of the project. They analyzed software projects as non cooperative games and shown how to use the tools and techniques of game theory to uncover some hidden causes of failures of software projects and we suggest ways to fix them.

Oza (2006) developed the initial understanding of game theory focusing on software engineering community. Particularly risk, rationality, payoffs, and other elements of game theory are explored in his work in terms of how they affect offshore software outsourcing. He stated that game theory can be helpful in evaluating many of the software business related aspects such as value perspectives, risk sharing and contractual matters. The essential elements of game theory are often classified as PAPI - Players, Actions, Payoffs and Information. He concluded that Software engineering is a human oriented endeavour and therefore all three central aspects of game theory namely strategies, payoffs and rational attitude of the players are relevant to it.

Murphy and Black (2008) stated that refactoring tools can improve the speed and accuracy with which developers create and maintain software but only if they are used. In practice, tools are not used as much as they could be because they do not align with the way most programmers refactor. How to mix refactoring with programming tasks, and how frequently choose to refactor is described with the use a dental metaphor as “floss refactoring” which is characterized by frequent refactoring, intermingled with other kinds of program change and “root canal refactoring” which is characterized by infrequent, protracted periods of refactoring, during which programmers perform few if any other kinds of program changes.
On the basis of their research, they have developed five principles to help programmers to choose appropriate refactoring tools and tool designers to design products that fit programmers’ purposes as choosing the desired refactoring quickly, switch seamlessly between program editing and refactoring, view and navigate the program code while using the tool, avoid providing explicit configuration information, and access all the other tools normally available in the development environment while using the refactoring tool.

From the related works it has been inferred and carried out that data modeling has been done based on assumptions between the relationships among the data. Assumptions taken into consideration in most cases do not have a true value in the real world. The modeler while developing an ER model must take into consideration the business logic governing the data and relationship among the data. Taking into consideration, the business logic governing the data and the relationship among the data, will aid the modeler to identify the appropriate degree of relationship and cardinality ratio. The goal behind modeling the ER model is to identify the entities, their attributes and relationship among the entities. Once the relationship among entities is modeled, the right cardinality ratio must be identified. Identifying the right degree of a relationship and cardinality ratio has a severe impact on database design.

A modeling methodology for modeling multivalued attributes in the ER model there by eliminating the need for normal forms that deal with multivalued attributes and nontrivial multivalued dependencies are proposed.

Compared to other works in the literature, the work discussed in this research work is different in the following ways: First, researchers have
proposed methods for decomposing a ternary relationship to equivalent binary relationships. In this research work, an attempt is made to illustrate that decomposing a ternary relationship to equivalent binary relationships, transforming the binary relationships to relations and then performing a natural join between the relations, we will not be able to reconstruct the relations which are a result of transforming a ternary relationship to relations. Second, work an attempt is made to illustrate the need to stop the normalization process with Boyce-Codd Normal Form. Third, a methodology which database designers must follow during each phase of database design is discussed. We have also illustrated that identifying the right degree of a relationship has an impact on normalized design.

Existing literature on refactoring techniques does not suggest any restructuring method that can be directly applied for PL/SQL code to the best of our knowledge. In the proposed work, the concept of refactoring applied to object oriented systems is extended to PL/SQL code, which thereby improves the understandability and maintainability of PL/SQL code.