

DIAGRAMMATIC APPROACH FOR COMPLETE AUTOMATION OF RELATIONAL DATABASE NORMALIZATION AT CONCEPTUAL LEVEL

Dr. M. S. Patwardhan, P. S. Dhabe, Asavari A. Deshpande, Sandhya G. Londhe,

M. L. Dhore, and H. K. Abhyankar.

Department of Computer Engineering,
Vishwakarma Institute of Technology, Pune-411037,
Maharashtra State, India.

ABSTRACT

AER (Articulated Entity Relationship) diagrams proposed by us, are the extension of Entity Relationship (ER) diagrams to accommodate functional dependencies (FDs) as their integral part. Accommodation of FDs in ER diagram helps to achieve total and unconditional automation of relational database normalization. In this paper we have implemented a software Integrated Development Environment (IDE), called AER IDE, designed by extending the Graphical Editing Framework (GEF) and Draw2d plug-in functionality of Eclipse. The AER IDE helps to draw, validate, store and normalize AER diagrams. An AER IDE has five modules. AER Editor allows creation and edition of AER diagrams using Model View controller (MVC) approach. AER diagrams are translated and stored as an xml file. AER Validator provides the facility of AER diagram validation based on a set of proposed rules. AER Normalizer allows normalization of an AER diagram, in one go, with the help of the proposed AER normalization rules and algorithms. Thus, it allows the total and unconditional automation of relational database normalization up to a given normal form; as an integral part of AER IDE. It also serves as a visual aid for the normalization process which is always easy to understand and interpret rather than theoretical approach. AER-XML Bidirectional Translator provides the import and export functionality of AER diagram, to and from an XML template making it compatible with the other toolset. AER-SQL Generator generates DDL scripts in the SQL schema format. The AER IDE is tested with ten distinct AER diagrams with all possible combinations of AER features and validation rules.

KEYWORDS

Entity Relationship Diagram, Automatic Normalization, Database Schema refinement, Relational Database Design.

[1] INTRODUCTION

Normalization is more complex specifically if the number of relations and number of attributes in each relation is high. Normalization when carried out manually can be time consuming, prone to errors and costly, since it needs high skilled personnel [1]-[2]. Thus, automating the process of normalization is the only solution to eliminate the drawbacks of manual normalization. The automation of normalization implemented by normalization tools such as JMathNorm [4], Micro [1], Normalizer [5], Web based Normalization Tool [6], NORMIT [7]. These Normalization tools consider database relations and FDs as two distinct inputs and thus lead to the Partial and Conditional Automation [3].

In [8], the authors propose bubble diagram to give pictorial representation of FD and leads up to 4th normal form at diagram level. The proposed approach bubble diagram is constructed from ER diagram by identifying the primary key attribute of the entity and replaces whole entity set by primary key bubbles and all other attributes are connected to the key bubble. The relations are constructed from the bubble diagram by identifying relationship and dependencies such as FDs, transitive dependencies and multivalued dependencies. The normalization rules are applied at diagram level to get normalized relations as bubble diagrams. However, bubble diagrams are difficult to understand for database developer's community. The approach does not provide an implementation of bubble diagram as a normalization tool and thus it is doubtful if the concept can be applicable for industry benefits.

In [9], the author's states that normalization process is not required at least till BCNF, if the real world problems and its constraints are properly presented in the ER model. The author then suggests how an ERD should be improved depicting a better conceptual model which when transformed to a relational schema using mapping algorithms results in normalized relations. The approach expects the database designer to be aware of the normalization concepts at the time of conceptual design and incorporate those concepts manually as a part of the ER diagram to make it normalized. Normalization process requires not only very good understanding of the organization of semantic/domain information but also how the schema can be organized to take care of these semantic dependencies. Having all this knowledge at the time of conceptual design is difficult. As it is a manual approach, it is prone to errors, time consuming and costly for today's enterprise applications.

As a part of our previous work, we propose an Articulated Entity Relationship (AER) diagram [3], which is an extension of Chen Model of ER diagram [2] to accommodate the FD information [10] as its integral part. In this paper, we develop a software IDE called AER IDE as a modeling tool for ARE diagrams implementing the philosophy discussed in our previous work [3]. AER IDE is an Eclipse plug-in and thus can be accommodated as a part of the Eclipse framework. Eclipse is an open source development platform for constructing customized IDEs. It can be used to create applications such as web sites, embedded Java programs, C++ programs, and Enterprise JavaBeans (EJBs) [11] and also works as a collection of "places-to-plug-things-into" (extension points) and "things-to-plug-in" (extensions) [12]. The AER IDE implemented by extending Graphical Editing Framework (GEF) and Draw2d, [13] -[14] which are the plug-ins of Eclipse, allowing us to develop graphical editor for AER model. The AER IDE helps to draw, validate, store and normalize AER diagrams.

By offering a facility to normalize an AER diagram in one click, AER Normalizer module of AER IDE not only allows total and unconditional automation of normalization, but also lets the database designer to have this automation performed at a very early stage of database design i.e. at conceptual level. To implement AER Normalizer, we have proposed normalization rules and algorithms on entity sets to get normalized entity sets up to BCNF. Thus, AER Normalizer provides pictorial approach for the automatic normalization of relational databases.

In further sections, we provide detailed information about implementation of AER IDE. Remaining part of this paper is organized as follows. Section 2 provides information about related work and comparison of various tools. Section 3 throws light upon the AER IDE system architecture along with its implantation details. Section 4 depicts the conclusion and future work.

[2] RELATED WORK

Several tools are available which allow database designer to draw and model ER diagram. In this section we provide an overview of the toolset which we considered for extending the ER Model to define AER diagrams [3].

A comparison of all the ER drawing and modeling tools which use Chen like model for ER definitions is provided in Table 1. We compare tools based on the criteria such as if the tool is open source or need to buy, drawing or modeling or both, if it provides the facility to generate SQL script, whether source code is available for extension purposes and the allowed export formats.

Table1. Summary of ER Modeling Tools

Sr. No.	ER Tools	Drawing /Modeling	SQL Script	Export	Free/Open Source/Buy	Extendable
1.	Concept Draw PRO 8.0.7.4 [15]	Drawing	No	.pdf,.html	Buy	No
2.	Smart Draw 2010[16]	Drawing	No	.doc,.emf,jpeg, .tiff,.wmf,.ppt	Buy	No
3.	Edrawz Max 5.1[17]	Drawing	No	.html, .pdf, .doc, .ppt, .svg, .tiff, Any graphics format (.jpeg, .gif, etc.)	Buy	No
4.	Eclipse ERD[18]	Both	Yes	.java, .xml	Open Source	Yes
5.	DDS Pro[19]	Both	Yes	NA	Buy	No
6.	ERDraw[20]	Both	Yes	.erml	Free	No

From the Table 1, the eclipse platform and its plug-in Eclipse ERD [18], satisfies all criteria such as if the tool is open source or need to buy, drawing or modeling or both, if it provides the facility to generate SQL script, whether source code is available for extension purposes and the allowed export formats of our comparison and thus is chosen for the implementation of AER diagram. The implementation details of various parts of AER IDE discussed in the further sections.

Our study of already existing automatic normalization tools such as JMathNorm [4], Micro [1], Normalizer [5], etc, show that, no tool exists which allows automation of normalization at conceptual level. On the other hand, these tools perform normalization at relational level. For example, in [1], a system called as “Micro” is proposed which uses two-linked list, one for storing attributes and other for storing FDs. User has to enter attributes, keys, and FD information using a GUI interface and can normalize a relation up to BCNF. However, this approach provides automation of normalization only at the relational level. Also, the tools take every relation as individual input and thus, is incapable of taking the whole relational schema as a single input to normalize it in one go.

Some toolsets such as NORMIT [7], Web based Normalization Tool [6], etc, are specifically designed for learning/teaching/understanding the process of normalization, since the process is difficult to understand and theoretical and thus it is difficult to motivate students. But these tools do not provide visual aid for normalization. Our approach provides visual aid to normalization process which is always easy to understand and interpret rather than theoretical approach.

[3] AER IDE SYSTEM ARCHITECTURE

This section provides architecture of the AER IDE tool depicted in Figure 1. The AER IDE consists of five modules i.e., AER Editor, AER Validator, AER-XML Bidirectional Translator, and AER-SQL Generator. We provide detail description of these modules in the following subsections.

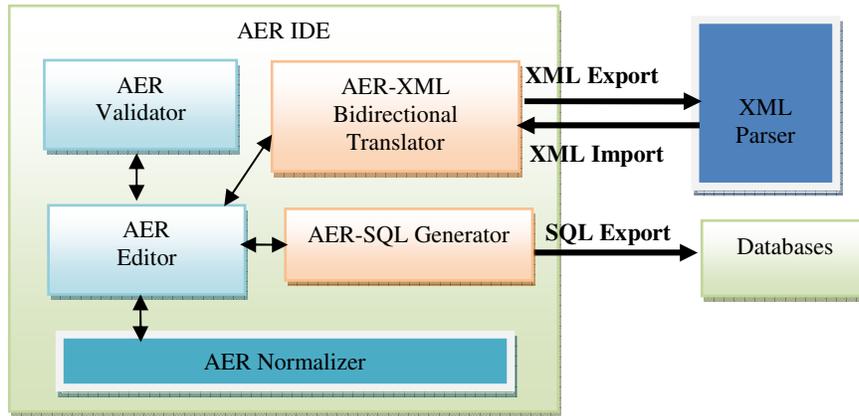


Figure.1 System Architecture of AER IDE

3.1.AER Editor

In this section, we discuss in detail the AER editor which helps in drawing and editing AER Diagrams. The AER Editor is composed of various views such as graphical editor along with palette view, property view as shown in Figure 2. We have implemented the definitions of the AC and FDC connectors and the Attribute Articulation Point (AAP) as AER features, which can be displayed and used using AER Editor.

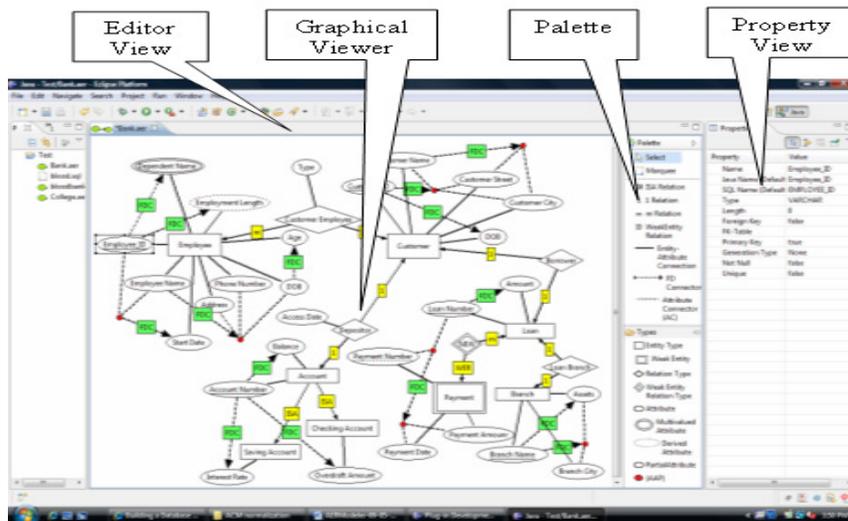


Figure.2 AER Editor GUI View

We have added the implementation of missing features of ER Model such as multi-valued attribute, derived attribute, partial key attribute, weak entity, weak entity relation, and weak entity relation connector to the basic ER drawing IDE. Along with these missing ER features, we have coded the newly introduced AER features allowing the AER editor to support the database designer to incorporate FDs as a part of database design in the form of AC, FDC, and AAP implementation.

The AER IDE is implemented by using Graphical Editing Framework (GEF) and Draw2D. The GEF adds editing support to graphical applications by implementing MVC (Model-View-Controller) framework [14]. The implementation of MVC as far as the AER IDE is concerned is depicted in Figure. 3.

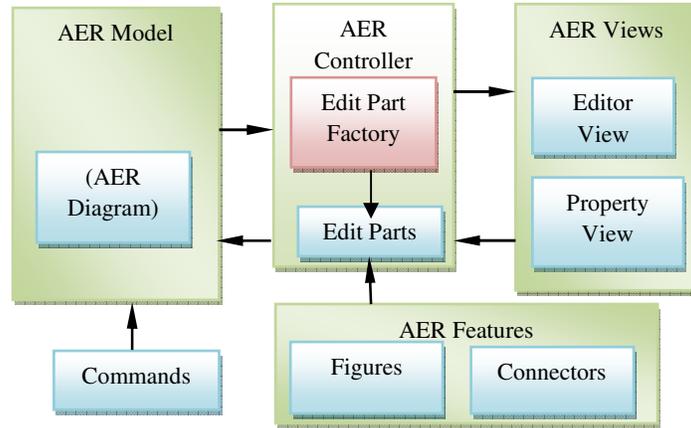


Figure. 3 MVC schematic for AER Editor

3.2.AER Validator

The AER validation is the process of checking correctness of the AER diagram by implementing the validation rules. The validation rules makes sure that the AER diagram follows the AER Model definition and thus helps the user to draw correct AER diagram for better automation. In this section, we have discussed the validation rules for AER diagram and their implementation by using Eclipse plug-in AER Validator

3.2.1.Validation Rules

The validation rules discussed in this section are categorized mainly into two parts. First part provides information about the rules defined by Chen like model and implemented by us. These set of rules are enlisted in Table 2. The second part throws light upon the rules proposed by us for valid representation of AER given by Table 3, Table 4, and Table 5.

Table 2. Validation rules for Attribute Connector (AC)

Sr. No.	Rule Description	Graphical Representation				Implementation Details
		Valid	Invalid			
1	<p>Attribute Entity Connection</p> <p>Explanation :</p> <p>An attribute must be connected to an Entity</p>					Invalid message on saving the digram with a standalone attribute

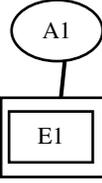
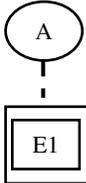
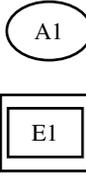
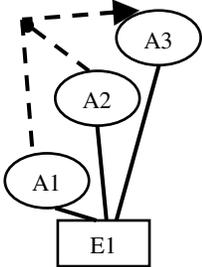
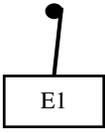
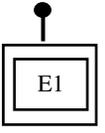
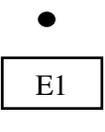
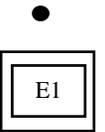
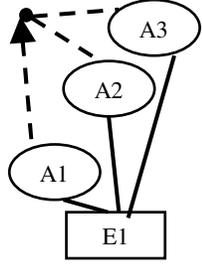
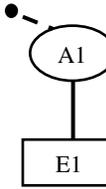
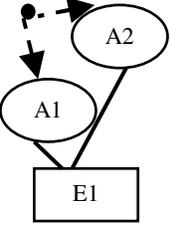
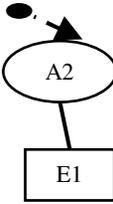
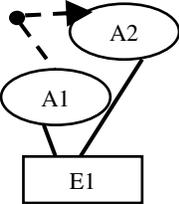
	<p>or Weak Entity by using Attribute Entity Connector (AEC). It cannot stand on its own without a connection to an Entity.</p>					<p>Not allowing the connection between an attribute and an entity using AC,FDC while drawing the diagram.</p>
--	--	---	---	--	---	---

Table 3. The validation rules for Attribute Articulation Point

Sr. No.	Rule Description	Graphical Representation		Implementati on Details	
		Valid	Invalid		
1.	The AAP cannot connect to an Entity.	<p>FD: $A1, A2 \rightarrow A3$</p> 			<p>Unable to connect and AAP to an antity by using AEC, AC or FDC while drawing the diagram.</p>
2.	AAP cannot stand on its own.				<p>Invalid message on saving the diagram in case of stand alone AAP.</p>
3.	The AAP must be connected through one and only one FDC.	<p>Example FD: $A1 \rightarrow A2, A3$</p> 			<p>Invalid message on saving the diagram if AAP is connected using less than 2 ACs</p>
4.	The AAP must be connected through one and only one FDC.				<p>Invalid message on saving the diagram, if AAP is conned through more than one</p>

				FDCs
<p>5. The AAP cannot connect to the attributes of two distinct non weak entities. Example E1: FD: A1,A2 →A3 E2: FD: A21 →A22</p>				<p>Unable to connect FDC to attributes of distinct non-weak entities while drawing the diagram.</p>
<p>6. The AAP and Partial Key connection rule Explanation: AAPs can connect to prime key attribute of strong entity and partial key attribute of weak entity through AC and attribute of the same weak entity through FDC. Example for E3 FD: A11,A31 →A32</p>				<p>Unable to connect FDC while drawing the diagram.</p>

Table 4. Validation rules for Functional Dependency Connector (FDC)

Sr. No.	Rule Description	Graphical Representation		Implementation Details
		Valid	Invalid	
1.	The FDC cannot connect to an Entity.	<p>FD: $A1, A2 \rightarrow A3$</p>		Unable to connect FDC to an entity while drawing the diagram.
2.	For FDC source must be an AAP or an Attribute and Target must be an AAP or an Attribute.			
3.	FDC cannot connect attributes of two distinct entities except ISA relation Example FD: $A1 \rightarrow A2, A3$			Unable to connect FDC to the attributes of two distinct entities while drawing the diagram.
4	FDC to ISA relation connection rule Explanation : In ISA relation the FDC used to connect primary key attribute (determiner) of parent entity to attribute (dependent) child entity to represent FD. The vice versa is invalid.			Unable to connect FDC while drawing the diagram.

Table 5. Validation rules for Attribute Connector (AC)

Sr. No	Rule Description	Graphical Representation		Implementation Details
		Valid	Invalid	
1.	<p>The AC cannot be connected to an Entity.</p> <p>FD: $A1, A2 \rightarrow A3$</p>			<p>Unable to connect AC to an entity while drawing the diagram.</p>
2	<p>For source of AC must be an AAP and Target must be an Attribute.</p> <p>Example E1: FD: $A1, A2 \rightarrow A3$ E2: FD: $A21 \rightarrow A22$</p>			<p>Unable to connect FDC if the source and target requirements are not matched while drawing the diagram.</p>

3.2.2. Implementation of AER Validator

This section illustrates the implementation of the above discussed validation rules for AER model using AER Validator module of AER IDE. The AER Validator performs the validation of an AER diagram when user uses AER editor to draw an AER diagram. The validation rules for AC, FDC, and AAP get automatically enforced either at the time of saving of AER diagram or online while drawing the diagram. The database designer is informed of any violation of the validation rule either in the form of a message at the time of saving the diagram or restricting the designer from contradictory action while drawing AER diagram. In case of violation along with the message we are highlighting the invalid elements in AER diagram by using orange colour. For example, Figure.4 depicts the contraction of the rule of AER diagram, saying that the highlighted attribute cannot be standalone.

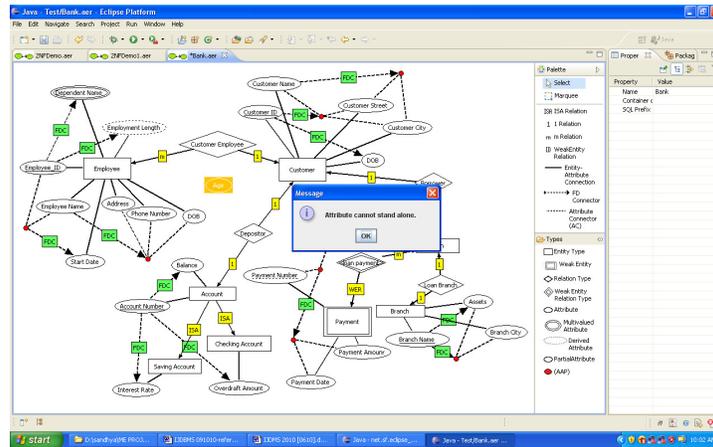


Figure.4 Validation Rule implementation

3.3.AER Normalizer

AER Normalizer is the module of AER IDE which takes the Entity and the Attributes of AER diagram as the input and normalizes that entity using the proposed AER normalization rules. The normalization rules (1NF, 2NF, 3NF, BCNF) are defined in terms of AER diagram model elements viz. entity, attributes, FDCs, ACs, and AAPs. The nomenclature and the definitions provided in set theoretic notations for the AER model elements are provided in Table 6.

Table 6. Nomenclature for AER Normal form Violation Rules

Notation	Description	Set Theoretic Definition
A	Attribute	--
FDC	Functional Dependency Connector	$FDC = Source(FDC) \cup Target(FDC)$
AC	Attribute connector	$AC = Source(AC) \cup Target(AC)$
AAP	Attribute articulation point	$AAP = \{FDC\} \cup \{AC_1, AC_2, \dots, AC_n\}$
Source(AC)	Source of attribute connector	$Source(AC) = AAP$
Target(AC)	Target of attribute connector	$Target(AC) = A$
Source(FDC)	Source of Functional dependency connector	$Source(FDC) = A \vee AAP$
Target(FDC)	Target of Functional dependency connector	$Target(FDC) = A \vee AAP$
E	Entity	$E = \{A_1, A_2, \dots, A_n\} \cup \{FDC_1, FDC_2, \dots, FDC_n\} \cup \{AC_1, AC_2, \dots, AC_n\} \cup \{AAP_1, AAP_2, \dots, AAP_n\}$
PK(E)	Primary key of Entity	$PK(E) = \{A_1, A_2, \dots, A_n\}$
CK(E)	Candidate key of the Entity	$CK(E) = \{A_1, A_2, \dots, A_n\}$
SOA(E)	Set of Attributes of Entity E	$SOA(E) = \{A_1, A_2, \dots, A_n\}$

SOFDC(E)	Set of functional dependency connectors of entity E	$SOFDC(E) = \{FDC_1, FDC_2, \dots, FDC_n\}$
AER	AER diagram	$AERDiagram = \{E_1, E_2, \dots, E_n\} \cup \{R_1, R_2, \dots, R_n\}$
SOE(AER)	Set of entities of AER Diagram	$SOE(AER) = \{E_1, E_2, \dots, E_n\}$
SOR(AER)	Set of relations of AER Diagram	$SOR(AER) = \{R_1, R_2, \dots, R_n\}$
Decompose (E,E ₁ ;E ₂)	Decompose entity E into new entities E ₁ and E ₂	$SOE(AER) = (SOE(AER) \cup \{E_1, E_2\}) - E$ $SOR(AER) = SOR(AER) \cup \{R\}$
R	Relation between Entities E ₁ , E ₂ ... E _n	$R = \{E_1, E_2, \dots, E_n\}$
Degree(R,E)	Degree of relation R with respect to Entity E	---

3.3.1. Normal Form violation Rules

3.3.1.1. AER 1NF Normalization Rule

The AER 1NF normalization rule makes sure that every attribute belonging to an entity or relationship is atomic. i.e.

$$\forall A \in E \quad (A.multivalued = false) \\ \wedge (A.composite = false)$$

The following is the violation algorithm of AER 1NF Normalization Rule for composite attributes:

$$\text{If } (A \in SOA(E)) \wedge (A.composite = true) \\ \wedge (A = \{A_1, A_2, \dots, A_n\}) \text{ then} \\ SOA(E) = SOA(E - A) \cup \{A_1, A_2, \dots, A_n\} \\ \text{If } A = \text{Target}(FDC) \text{ then} \\ \{A_1, A_2, \dots, A_n\} = \text{Target}(FDC)$$

If the attribute is composite, decompose it into multiple attributes (represented by a comma separated string) belonging to the same entity or relationship to which the composite attribute belonged.

The following is the violation algorithm of AER 1NF Normalization Rule for multi-valued attributes:

$$\text{If } (A \in E) \wedge (A.multivalued = true) \text{ then} \\ \text{Decompose}(E, E_1; E_2) \\ SOA(E_1) = SOA(E) - A \\ SOA(E_2) = PK(E) \cup A \\ PK(E_1) = PK(E) \\ PK(E_2) = PK(E) \cup A$$

If the attribute is multivalued then decompose the entity to which the attribute belong into two entities with the help of following algorithm.

3.3.1.2. AER 2NF Normalization Rule

The AER 2NF normalization rule states that every entity is in 2NF, if it first fulfils the requirements to be in 1NF and every FDC of that entity should not have its source as subset of the key of that entity (prime).i.e.

$$\forall FDC \in (E) \quad Source(FDC) \not\subset CK(E)$$

The following is the violation algorithm of AER 2NF Normalization Rule:

```

 $\forall FDC \in E$ 
begin
  If  $(Source(FDC) = A_s) \wedge (A_s \subset CK(E))$  then
    begin
      If  $(Target(FDC) = A_t) \wedge (A_t \not\subset CK(E))$  then
        Call Decomposition_Algorithm(E, FDC)
      If  $(Target(FDC) = AAP) \wedge (Target(AC) \not\subset CK(E),$ 
        where  $Source(AC) = AAP)$  then
        Call Decomposition_Algorithm(E, FDC)
    end
  If  $(Source(FDC) = AAP) \wedge (Target(AC) \subset CK(E),$ 
    where  $Source(AC) = AAP)$  then
    begin
      If  $(Target(FDC) = A_s) \wedge (A_s \not\subset CK(E))$  then
        Call Decomposition_Algorithm(E, FDC)
      If  $(Target(FDC) = AAP) \wedge (Target(AC) \not\subset CK(E),$ 
        where  $Source(AC) = AAP)$  then
        Call Decomposition_Algorithm(E, FDC)
    end
end
    
```

3.3.1.3. AER 3NF Normalization Rule

The AER 3NF normalization rule makes sure that every entity or relation of AER diagram is in 2NF and every FDC belonging to an entity is trivial or should have its source as a whole key and target as non-key attribute (key dependency) or should have its source as non-key attribute and target as key (prime) attribute. i.e.

$$\forall FDC \in (E)$$

$$(Target(FDC) \subset Source(FDC)) \vee$$

$$(Source(FDC) = CK(E) \wedge Target(FDC) \not\subset CK(E)) \vee$$

$$(Source(FDC) \not\subset CK(E) \wedge Target(FDC) \subset CK(E))$$

The following is the violation algorithm of AER 3NF Normalization Rule:

3NF is violated if there exist transitive functional dependency such as

$$\exists FDC \in (E) ((Source(FDC) \not\subset CK(E))$$

$$\wedge (Target(FDC) \subset CK(E)))$$

3NF violation algorithm is given below:

```

 $\forall FDC \in (E \vee R)$ 
begin
If (Source(FDC) =  $A_s$ )  $\wedge$  ( $A_s \notin CK(E)$ ) then
begin
If (Target(FDC) =  $A_t$ )  $\wedge$  ( $A_t \notin CK(E)$ ) then
Call Decomposition_Algorithm(E, FDC)
If (Target(FDC) = AAP)  $\wedge$  (Target(AC)  $\notin CK(E)$ ),
where Source(AC) = AAP) then
Call Decomposition_Algorithm(E, FDC)
end
end
If (Source(FDC) = AAP)  $\wedge$  (Target(AC)  $\notin CK(E)$ ),
where Source(AC) = AAP) then
begin
If (Target(FDC) =  $A_t$ )  $\wedge$  ( $A_t \notin CK(E)$ ) then
Call Decomposition_Algorithm(E, FDC)
If (Target(FDC) = AAP)  $\wedge$  (Target(AC)  $\notin CK(E)$ ),
where Source(AC) = AAP) then
Call Decomposition_Algorithm(E, FDC)
end
end;

```

3.3.1.4. AER BCNF Normalization Rule

The AER BCNF normalization rule makes sure that every entity or relation of AER diagram is in 3NF and every FDC of an entity should have its source as candidate key i.e.

```

 $\forall FDC \in (E)$ 
( $Target(FDC) \subset Source(FDC)$ )  $\vee$ 
( $Source(FDC) = CK(E) \wedge Target(FDC) \notin CK(E)$ )

```

The following is the violation algorithm of AER BCNF Normalization Rule:

```

 $\forall FDC \in (E \vee R)$ 
begin
If (Source(FDC) = A)  $\wedge$  (A  $\neq CK(E)$ ) then
Call Decomposition_Algorithm(E, FDC)
If (Source(FDC) = AAP)  $\wedge$  (Target(AC)  $\neq CK(E)$ ),
where Source(AC) = AAP) then
Call Decomposition_Algorithm(E, FDC)
end;

```

3.3.2. Decomposition Algorithm

In this section we explore the algorithm to decompose entities and algorithm to create relationship between decomposed entities along with relationship cardinality. Algorithm to determine

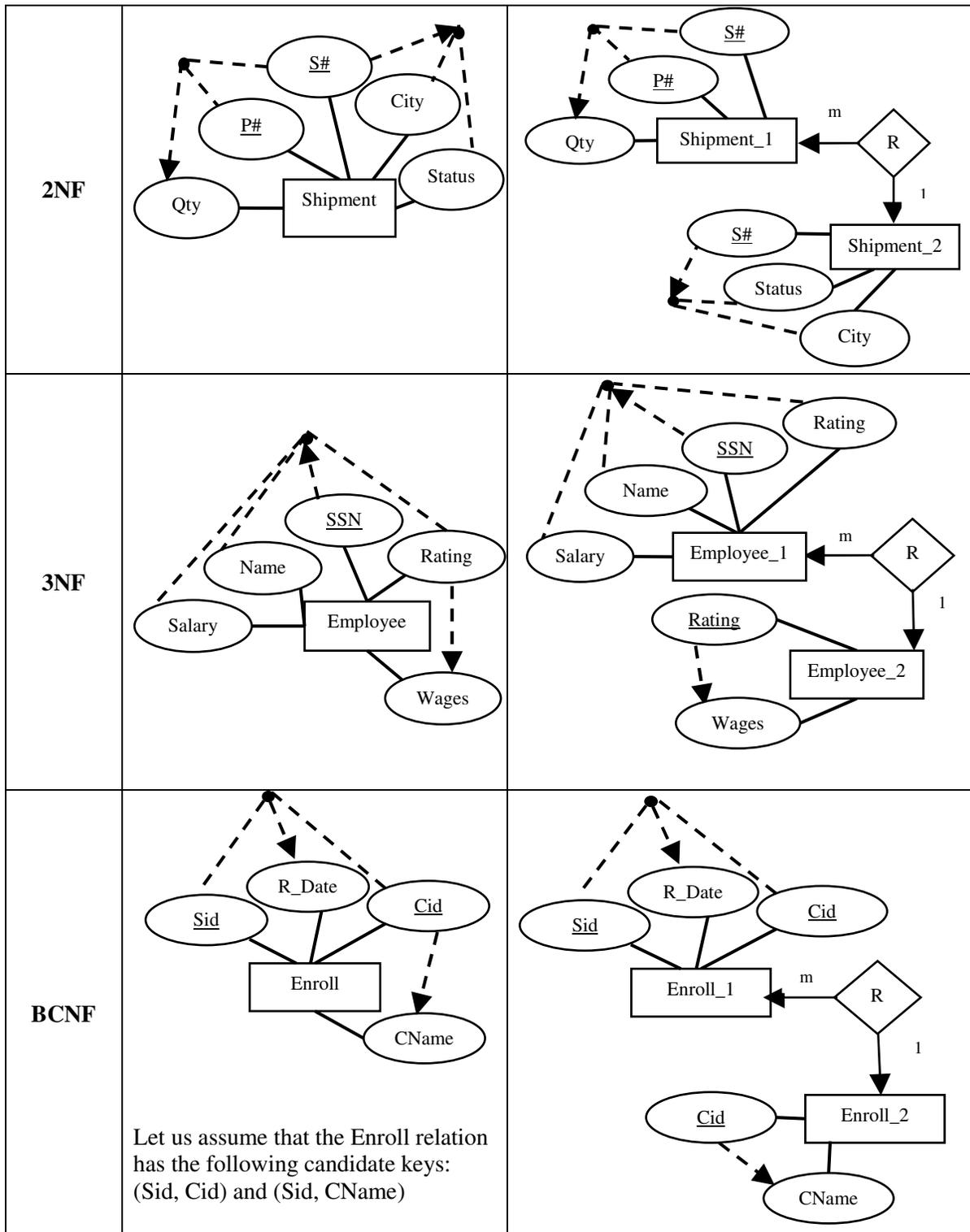
relationship cardinality is proposed in [24]. This algorithm is defined in terms of AER model elements.

```

Algorithm Decomposition _ Algorithm(E, CurrentFDC)
begin
    Decompose(E, E1; E2)
    E1.name = E.name _1
    E2.name = E.name _2
    SOA(E1) = SOA(E) - T arg et(CurrentFDC)
    SOA(E2) = Source(CurrentFDC) ∪ T arg et(CurrentFDC)
    PK(E1) = PK(E)
    PK(E2) = Source(CurrentFDC)
    FDC(E1) = FDC(E) - CurrentFDC
    FDC(E2) = CurrentFDC
    R = {E1, E2}
    If (SOA(E1) ∩ SOA(E2)) = PK(E1) then
        Degree(R, E1) = 1 else Degree(R, E1) = m
    If (SOA(E1) ∩ SOA(E2)) = PK(E2) then
        Degree(R, E2) = 1 else Degree(R, E2) = m
    AER Diagram = (AER Diagram - {E}) ∪ {E1, E2} ∪ {R}
    SOE(AER) = (SOE(AER) ∪ {E1, E2}) - E
    SOR(AER) = SOR(AER) ∪ {R}
end;
    
```

Table7. AER diagram demonstrating Normal Forms

Normal Form	Graphical presentation	
	Input	Output
1NF	<p>Note: Address is composite attribute consists of City and Street</p>	

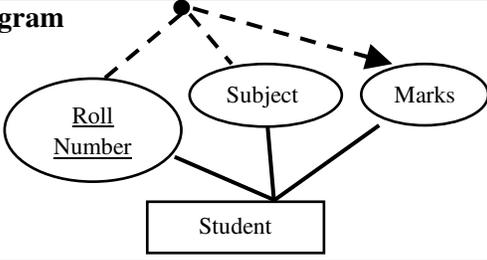


3.4.AER-XML Bidirectional Translator

This section illustrates our implementation of an extension point for the AER IDE tool, the AER XML translator. The translator performs bidirectional functionality such as XML import and XML export, translating the current AER diagram (.aer) in XML format.

In XML export functionality the AER XML translator helps in DTD generation of AER diagram with .xml extension and thus export diagram in XML format. On the save command, the current AER diagram is internally saved in XML format with .aer as an extension. On XML Export command the AER XML translator reads this .aer file and filters the graphical information from the file to generate DTD representation. The graphics information such as height, width, x and y coordinates of the figures are omitted and the AER representation details are maintained in the file. Also the identifiers of the model elements are removed and for the connector information the source and destination identifiers are replaced by the actual names of the model elements such as entity name, attribute name, etc. For example, in Table 8 column (A) represents the AER Diagram, Column (B) represents the .aer file which is generated after saving the diagram and column (C) shows the corresponding .xml file which is generated as a result of XML Export functionality.

Table 8. AER diagram example with XML export

<p>(A) The AER Diagram</p> 
<p>(B) The .aer file which is generated after saving the diagram :</p> <pre data-bbox="211 1155 1161 1848"> <? xml version="1.0" encoding="UTF-8"?> <diagram container="" lastSavePath="\Test\blood.sql" name="" sqlPre=""> <elements> <entity gen_position="0" id="2133325842" name="Student" pname="Student" sname="Student" sql_sequenz_name=""> <attribute fkeytbl="" foreignkey="false" genType="None" id="2041377926" length="0" name="Roll no" notnull="false" pname="Roll no" primarykey="true" sname="ROLL NO" type="VARCHAR" unique="false"> <position h="-1" w="-1" x="678" y="491"/> </attribute> <attribute fkeytbl="" foreignkey="false" genType="None" id="-1512400783" length="0" name="Subject" notnull="false" pname="NEW" primarykey="false" sname="NEW" type="VARCHAR" unique="false"> <position h="-1" w="-1" x="716" y="448"/> </attribute> <attribute fkeytbl="" foreignkey="false" genType="None" id="2076192327" length="0" name="Marks" notnull="false" pname="NEW" primarykey="false" sname="NEW" type="VARCHAR" unique="false"> </pre>

```

        <position h="-1" w="-1" x="704" y="192"/>
    </attribute>
</entity>
<articulatedp gen_position="0" id="47197359" name="NEW" pname=""
    sname="NEW" sql_sequenz_name="">
</elements>
<connections>
    <aec from="2041377926" to="2133325842"/>
    <aec from="-1512400783" to="2133325842"/>
    <aec from="2076192327" to="2133325842"/>
    <ac from="47197359" to="2041377926"/>
    <ac from="47197359" to="-1512400783"/>
    <fdc from="47197359" to="2076192327"/>
</connections>
</diagram>

```

(C) The .xml template file which is generated as a result of XML Export functionality :

```

<?xml version="1.0" encoding="UTF-8"?>
<AERDiagram>
<entity>Student
    <attribute length="10" type="primarykeyattribute" datatype="VARCHAR">RollNo </attribute>
    <attribute length="10" type="attribute" datatype="VARCHAR">Subject</attribute>
    <attribute length="10" type="attribute" datatype="VARCHAR">Marks</attribute>
</entity>
<FD determiner=" Roll no " dependent=" AAP1 "/>
<FD determiner=" Subject " dependent=" AAP1 "/>
<FD determiner=" AAP1 " dependent=" Marks "/>
</AERDiagram>

```

3.5.AER-SQL Generator

In this section we are considering the implementation details of AER SQL generator. It is an extension point of the AER IDE. The generator helps to generate relations from the AER diagram and saved with .sql file extension. The generator extracts the SQL script from the diagram by applying all the mapping rules [21] from ER diagram to relations. We are using the set of mapping rules which are already defined in the Eclipse ERD. Here we provide an example of a mapping rule specified for strong entities. In case of the equivalent representation of strong entities defined in an AER diagram into the relational schema, we need to create a new table (relation) for each strong entity and make the indicated key of the strong entity the primary key of the table. For example, the SQL script generated for the Table 8 column (A) by implantation of strong entity mapping rule, the following equivalent relational schema is generated:

```

CREATE TABLE STUDENT ( ROLLNUMBER VARCHAR (10), SUBJECT VARCHAR (10),
    MARKS VARCHAR (10), PRIMARY KEY (ROLLNUMBER));

```

[4] CONCLUSION AND FUTURE WORK

We have successfully implemented an Eclipse Plug-in called AER IDE which facilitates the drawing and validation of AER diagrams. AER drawing functionally provided by the AER Editor module allows database designer to accommodate the Functional Dependency information as an integral part

of the ER model in terms of the Attribute Connector (AC), Functional Dependency connector (FDC), and Attribute Articulation Point (AAP). Apart from AER drawing functionality, the AER validation rules are defined and implemented to maintain correctness of the AER diagram with the help of AER Validator Module. AER Normalizer allows normalization of an AER diagram, in one go, without any user intervention. The normalization of AER diagram is facilitated by the proposed set of AER normalization rules and algorithms.

AER IDE also provides additional features such as of XML translation and export and SQL generation. As a result of the XML export functionality, the relations and the FDs depicted through an AER diagram are embedded in an XML template omitting the graphics related information. This allows easy exchange of relational information with other database modeling and design toolsets. The AER-SQL Generation module acts as an extension point which generates the DDL script (.sql format) of the corresponding AER diagram, which can further be used to create database tables by using different software such as oracle, MySQL etc.

Further Extensions of AER IDE include incorporation of definitions of the multi-valued dependencies and joint dependencies as a part of AER diagram, which would lead to the automation of the process of normalization up to 5NF [22]. It can also be further extended to include the domain and key constraints to automate normalization up to Domain Key Normal Form [23]. But embedding this information by maintaining acceptable complexity of the AER diagram is a big challenge.

ACKNOWLEDGEMENT

We are thankful to University of Pune for providing financial support to the research project Titled “Implementing Articulated Entity Relationship Diagram (AER) for Complete Automation of Relational Database Normalization”, Inward Number 0910-0167. We are also thankful to the management Vishwakarma Institute of Technology, Pune for their encouragement and whole hearted support for this work. First four authors are thankful to Prof. S.Y. Prabhu, Dean, planning and Prof. S. N. Mali, Dean, student’s welfare, for their constant motivation and support.

REFERENCES

- [1] Du Hongbo & Wery Laurent, (1999) “Micro: A normalization tool for relational database Designers”, *Journal of Network and Computer Applications* 22, pp. 215-232.
- [2] Chen Peter & Pin-SHAN, (1976). “The entity-relationship model-toward a unified view of data”, *ACM Trans. on Database Systems*, Vol. No. 1, Pages 9-36.
- [3] P.S.Dhabe, M.S.Patwardhan & Asavari Deshpande, (2010) “Articulated Entity Relationship (AER) Diagram For Complete Automation Of Relational Database Normalization”, *International Journal Of Database Management Systems(IJDMS)*, Vol.2, No.2.
- [4] Yazici Ali & Karakaya Ziya., (2007) “JMathNorm: A database normalization tool using mathematica”, Springer-Verlag Berlin Heidelberg, pp. 186193
- [5] NABIL ARMAN, (2006). “Normalizer: A case Tool to normalize relational database schemas”, *Information Technology Journals* (2) 329-331 .ISSN 1812-5638.
- [6] Kung Hsiang-Jui., (2006) “A web-based tool to enhance teaching/learning database normalization”, *Proceedings of the Southern Association for Information Systems Conference* 251-258
- [7] Mitovic Antonija, (2002) “NORMIT: Web-Enabled Tutor for Database Normalization”, *Proceedings of IEEE International Conference on Computer Education(ICCE-02)*.

- [8] DAWSON KATHYN S.& PARKER LORRAINE M. PURGAILIS,(1998) “From Entity Relationship Diagram to Fourth Normal Form: A Pictorial Aid to Analysis”, The Computer Journal, Volume 31, Number 3
- [9] TAUQEER HUSSAIN*, SHAFAY SHAMAIL, & MIAN M. AWAIS(2003) “Eliminating process of normalization in relational database design”, Proceedings IEEE INMIC CODD E. F.,(1970) “A relational model of data for large shared data banks”, Communications of the ACM, vol. 13, No.6, pp. 377-387.
- [10] CODD E. F.,(1970) “A relational model of data for large shared data banks”, Communications of the ACM, vol. 13, No.6, pp. 377-387.
- [11]Daum Berthold, (2004). “Professional Eclipse 3 For Java Developers”, Wiley Publication
- [12]Gamma, E., Kent & B(2004). “Contributing to Eclipse: Principles, Patterns, and Plug-Ins”, Addison-Wesley.
- [13]Riel Bart van.,(2008) “MVC your graphics with Eclipse GEF”, Internet Document <http://www.tutorials.tfo-eservices.eu>
- [14] Moore Bill, Dean DAVID, GERBER ANNA, WAGENKNECHT GUNNAR, VANDERHEYDEN Philippe. IBM Redbooks _ GEF and EMF,(2004), [online] <http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246302.html>
- [15]ConceptDrawPro 8.0.7.4, <http://www.conceptdraw.com/en/products/cd5/main.php>.
- [16]SmartDraw2010,[online], <http://www.smartdraw.com/product/upgrade/>.
- [17]Edraw Max 5.2, [online] <http://www.edrawsoft.com/>.
- [18]Eclipse ERD, [online] <http://eclipse-erd.sourceforge.net/>
- [19]DDSPro2.21.1, <http://database-design-studio-professional-dds-pro.chilli-source.qarchive.org/>.
- [20]SHUYUN XU, YU LI & SHIYONG LU. “ERDraw: An XML-based ER-diagram Drawing and Translation Tool”, DBLP conference, 143-146.
- [21]BAGUI SIKHA & ERAP RICHARD. “Database Design Using Entity-Relationship Diagrams”, ISBN: 0849315484, Auerbach Publications
- [22]Date C. J,(2002). “An introduction to database systems”, Pearson Education, Seventh Edition
- [23]FAGIN RONALD. “A normal form for relational databases that is based on domains and keys”, ACM Trans. on Database Systems, Vol. 6, No. 3, Pages 387-415
- [24]Hsiang-Jui Kung., (2007) “COMPARING TWO BOTTOM-UP DATABASE DESIGN METHODS”, Proceedings of the 2007 Southern Association for Information Systems Conference

Authors

Dr. M. S. Patwardhan (manasi.kelkar@gmail.com) has perused a Ph.D. in Computer Science from University of Tulsa, OK, USA, in year 2007. She is presently working as associate professor in computer engineering at Vishwakarma Institute of technology, Pune from 2009. She was instrumental in playing a role of a researcher in KBSI, USA (2007-2009). Her areas of interest are database normalization, software modelling and computer security and parallel computing.



P. S. Dhabe (dhabe@vit.edu), has completed ME in computer technology from S.G.G.S college of engineering and technology, Nanded, Maharashtra, India in year 2002. He is presently perusing his Ph.D. in systems and control engineering, IIT Mumbai. He is presently working as assistant professor in computer engineering at Vishwakarma Institute of technology, Pune from 2005. He is principal investigator of research project funded by university of pune, Maharashtra, India, titled "Database normalization tool". His areas of interest are database normalization, fuzzy neural networks and pattern recognition.



Asavari A. Deshpande (asavari.deshpande@vit.edu) has perused B.E in Computer Science and Engineering from M.G.M. college of engineering , Nanded, Maharashtra, India in year 2004. She is presently perusing Masters in Computer engineering from 2008. Her area of interest is database normalization.



Sandhya G Londhe (londhesandhya@vit.edu) has perused B.E in Information Technology from Government college of engineering, Karad, Maharashtra, India in year 2006. She is presently perusing Masters in Computer engineering from 2009. Her area of interest is database normalization.



M.L.Dhore (manikrao.dhore@vit.edu) has completed ME in computer engineering from NITR, Chandigarh, India in 1998. Currently he is working as head and associate professor in computer engineering at Vishwakarma Institute of technology, Pune. Presently he is perusing his Ph.D. from University of Solapur, Maharashtra, India, in Web localization. His areas of interest are Webpage Localization and Computer Networking.



H.K.Abhaynkar (director@vit.edu) has completed ME in control systems from Shivaji University, Kolhapur, Mahatashtra, India in 1989. He is presently working as Director and Professor at Vishwakarma Institute of technology, Pune since 1994. He was member of Board of studies at Pune University and Shivaji University, Kolhapur, Mahatashtra ,India. He has awarded by ISTE as a best Engineering college Principal at National Level in 1999. He is well recognized academic leader by various organizations in India and abroad. Recently he has been awarded by PRAJ industries as a "MAHA Intrepeneur" in 2009.

