On mapping an Enterprise Class Model directly into Third Normal Form(3NF) integrated Database

Amita Jain

SR.LECTURER, BHAGWAN PARSHURAM INSTITUTE OF TECHNOLOGY, GGSIPU, DELHI

AMITA_JAIN_17@YAHOO.COM

Devendra K. Tayal

ASSOCIATE PROFESSOR, DEPTT. OF CSE, GGSIPU, KASHMERE GATE, DELHI

DR.TAYAL@IPU.EDU

Phase-II: Empirical Article

ABSTRACT

• One of the most crucial steps in any ERP implementation is the creation of an Integrated data model. The company uses this integrated data for analysis and decision making. The normalization of integrated databases, in particular, relational schemas is the cardinal part of relational database design. The relational schema generated from the conceptual model is rendered normalized so that the redundancy in the data is removed and the scope of propagation of anomalies is reduced. Earlier the Entity-Relationship Diagram(ER Diagram) and subsequently the Extended Entity Relationship Diagram(EER Diagram) was used for conceptual modeling of the database. But now a days UML 2.0 Class Model is considered as the de-facto standard in the software industry for conceptual-modeling of the integrated data model of an enterprise. The conceptual model is then mapped to the relational database schema in a relational database. We present a formal algorithm to map the class diagram into relational schema . This schema then has to undergo various normalization checks based on different types of data dependencies, the primary of them being the functional dependency. In this paper we consider the functional dependencies and analyze why unnormalized relations are created when the integrated conceptual model of the enterprise(represented using class model) is mapped to integrated database schema. We propose the changes in the class model itself at the time of conceptual modeling so that the relational schema so generated will always be in 3NF. Our results will thus eliminate the need of database normalization upto 3NF which is otherwise a hectic and time consuming activity.

KEYWORDS

• Normalization
• Class –Model
• Functional dependency .

Introduction

The relational database model used for developing an integrated database for an enterprise relies on developing relational schemas which can support efficient query processing and are free from various anomalies like insertion, deletion and update anomalies. Basically two approaches for database design have been proposed in the literature viz. bottom-up-methodology and decomposition-methodology. A bottom-up design methodology would consider the basic relationships among individual attributes as the starting point and it would use those to build up relations. Other than the binary relational model, this approach is still not popular and suffers from the problem of collecting a large number of binary –attribute relationships at the starting point [4] and Universal relation problem[6]. In contrast top-down design methodology would start with a number of grouping of attributes into relations that have already being obtained from conceptual design and mapping activities. Design by analysis is then applied to the relations individually and collectively, leading to further decomposition until all desirable properties are met. The relational schema so obtained from the conceptual schema have to therefore go through rigorous normalization checks based on different types of data dependencies occurring. Although more than 100 data dependencies are known today but the primary of them are functional dependencies which are basis of First Normal Form upto Boyce Codd Normal Form, the other being Multivalued Dependencies(Fourth Normal Form) and Join Dependencies (Fifth Normal Form) etc. Practically every relational database used in an Enterprise is rendered at least in Third Normal Form. Since first three normal forms are based on the functional dependencies, the designer has first the responsibility of identifying the initial set of functional dependencies occurring on the relational schemas. Since this set may be incomplete at initial stage and similarly may also contain some redundant functional dependencies(FD), therefore the closure of a set of FDs is computed and then its minimal cover is obtained, which is practically a time consuming and hectic procedure. Further these relational schemas so generated then undergo normalization checks and consequently decomposed into multiple schemas. The aim of this paper is to remove this hectic process of database designing. In this paper we propose a database designing methodology which makes the need of normalization redundant. We propose some changes in the conceptual database schema designing so that the relational schemas so obtained from the conceptual schema are themselves in third normal form and therefore there is no need to go for normalization checks.

For logical database design, there have been many methodologies, tools and notations that best model, design and build DBMS applications from analysis through to database implementation, the most famous and highly applicable was the ER Model[2] and subsequently the EER Model[4]. However some of the above methodologies were very strict in the process and very tool sensitive. The upper CASE tools & lower CASE tools were also used and some of the organizations preferred the best- of-breed solutions. The CASE tools followed a strict process and the best-of-breed solutions lacks in providing integration and information sharing. The UML had both the flavors, it is flexible enough and also support the teams involved to work together one way and to do their own part as needed [8]. UML has therefore become the de-facto standard in the Software industry for DBMS development. We use Class Model (also called Class-Diagrams) proposed in UML 2.0 for representing the conceptual schema of the database.

We first discuss a formal algorithm to map a class-diagram into relational schemas and then analyze the causes of the violation of the various normals forms based on functional dependencies forms in the relational schemas. We argue that since the conceptual schema is generating un-normalized relations therefore there is a scope of some improvement in the conceptual schema designing itself so that it may always generate normalized relations only. We propose rules to change the conceptual schema itself so that when relational mapping algorithm is applied the result is the normalized schema up to 3NF.

The paper is organized as follows: After discussing the introductory concepts in section 1, we present a formal class diagram to relational mapping algorithm in section 2. In section 3 ,we analyze the various causes of violation of normal forms and propose the rules to modify the conceptual schema so that the result generated is always in the respective normal form.

2. CLASS DIAGRAMS TO RELATIONAL MAPPING

Earlier the relational database design approach used the Entity-Relationship Diagram [2] to represent the logical Model. It was further extended to Extended Entity Relationship Diagram (EER Diagram)[4] to represent the concepts in a better way. Elmasri and Navathe[4] discuss the mapping rules to derive a relational schema from the conceptual relational schema. However no formal algorithm is available in the literature to map the class diagram into relational schema, although some informal rules have been discussed. In this paper we provide a formal algorithm to map the cardinal constructs of class diagram into relational schema. In this section we present a formal algorithm to map the important constructs of the class diagrams into database relations based on the approaches discussed in [3,8].

ALGORITHM:

1. For each class in the class diagram create a relation. Add simple attributes of the class as the attributes in the relation.. If primary key attribute is identified in the class, define it as the primary key attribute in the newly created relation otherwise designate one attribute (or a set of attributes) from the class as the primary key of the relation which is capable of identifying every row of the table uniquely.

2. For every multi-valued attribute of a class , create a separate relation. This relation will have a joint key composed of the primary key of the class along with the multi-valued attribute.

3. For every 1:1 association relationship R between the two classes C & D , include the primary key of C as a foreign key in the relation corresponding to class D.

4. For every 1 to many(0……*) association relationship R between the two classes C & D with C on '1' side and D on (0….*) side, include the primary key of C as a foreign key in the relation corresponding to class D.

5. For every many (o….*) to many(o…..*) relationship create a new relation . This relation will

have a joint key composed of keys of relations corresponding to both the participating entity types.

6. For every participation constraint in the class diagrams, create assertion to represent the constraint.

7. For every recursive relationship R with parent and child roles, create a new relation. The key of the new relation will be its child key.

8. For every n-ary association relationship, create a new relation R. This new relation will have a joint key composed of primary keys of all the relations corresponding to the participating classes.

For sake of brevity we are not providing here an implementation example which shows execution of the above algorithm , but it can easily be executed as given in [4,8].

3. EXTENDING CLASS DIAGRAMS TO AVOID VIOLATION OF NORMAL FORMS

In this section we discuss the changes in the class diagrams needed at the time of designing so that the relational schema generated after applying the mapping algorithm described in section -3 always generates a normalized schema . We analyze the causes of the violation of normal forms by discussing every normal form individually.

3.1: NON-FIRST NORMAL FORM CLASS DIAGRAM

The relational database theory mentions that a relation violates the First Normal Form if
(i) it contains an attribute which can have multiple values i.e it is a multivalued attribute.
(ii) it is a composite attribute.
(iii) it contains relations within relations.

As far as violation due to (i) condition is concerned, it will never occur because multivalued attributes are mentioned explicitly in the class diagrams and mapped accordingly (Rule 2 of Algorithm).

Secondly, in business class diagrams, the attribute types usually correspond to units that make sense to the likely readers of the diagram (i.e., minutes, dollars, etc.). However, a class diagram that will be used to generate code needs classes whose attribute types are limited to the types provided by the programming language, or types included in the model that will also be implemented in the system[1]. Assuming that the database is implemented in a programming language which does not support aggregate data structures, we find that the composite attributes (ii) will also never propose any violation of 1NF as any element belonging to the data-type is considered atomic. Moreover since there is no provision of creating class within a class in Class-Diagrams, so the question of existence of relation within a relation does not arise (condition (iii)). Therefore we find that the relation schema generated from a class diagram after applying the mapping rules will always be in First Normal Form.

3.2: NON-SECOND NORMAL FORM CLASS DIAGRAM

Since the class diagram generates relations, we have to ascertain that the relations generated by the class do not contain any functional-dependency which violates the 2NF. In this section we analyze those functional dependencies whose presence violates the satisfaction of 2NF criteria.
Let R(A1,A2…………………….. Ak,Ak+1,………………………An) be a relational schema corresponding to a class C in the class diagram such that A1,A2…………………….. Ak is the primary key/ candidate key denoted as "PK" of R and Ak+1,………………………An are the non-prime(NP)attributes of R. The second normal form is violated when there exists atleast one partial functional dependency $X \rightarrow Y$ such that $X \subset$ A1,A2…………………….. Ak and $Y \subseteq$ Ak+1,………………………An
. This shows that since a subset of PK i.e X is capable of functionally determining $X \cup Y$. Logically this must correspond to a separate class in the class diagram , then only the real world constraint specified by the functional dependency $X \rightarrow Y$ will be satisfied. So we create a new class say C' containing the attributes $X \cup Y$ such that X is a full key of C'. Also the class C will have attributes (PK – X) $\cup$ (NK-Y). The new representation will provide a better logical design as all the relational schemas will now be in 2NF. We formally state the rule as follows:

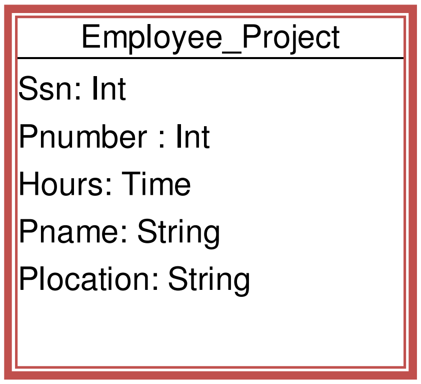
RULE 1:

For every functional dependency $X \rightarrow Y$ where $X \subset PK$ and $Y \subseteq NK$,

(i) create a new class $C^{'}$ such that :
Attributes $(C^{'}) = X \cup Y$
and $PK(C^{'}) = X$

(ii) Set Attributes(C)=(PK-X) $\cup$ (NK-Y)

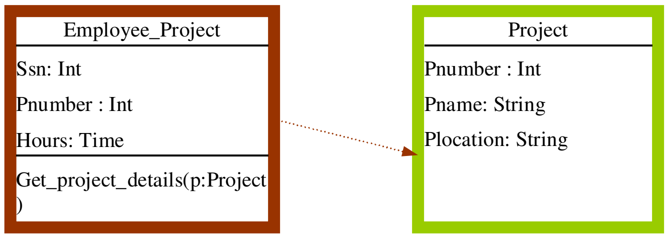
We illustrate our rule using the following example:
Let EMPLOYEE_PROJECT be class in the conceptual schema which contains information



about the employee working on different projects
In the above class the attribute set (Ssn,Pnumber) is the primary-key and the following functional dependencies hold on it.

Ssn,Pnumber → hours
Pnumber → Pname
Pnumber → Plocation

Here both the functional dependencies Pnumber → Pname and Pnumber → Plocation violate the 2NF condition as both of them are partial dependencies, so according to Rule 1 we will create another class PROJECTS with the attributes Pnumber, Pname and Plocation with Pnumber as the primary key. The



class EMPLOYEE_PROJECT will contain the attributes Ssn, Pnumber, Hours with (Ssn,Pnumber) as the primary key. Create a "dependency-relationship" between the class PROJECTS and EMPLOYEE_PROJECT as the class EMPLOYEE_PROJECT may use the class PROJECT to get the details of the projects in its operation therefore the class EMPLOYEE_PROJECT is dependent on EMPLOYEE.
Note that the relational schema obtained after applying the mapping algorithm will always fetch relational schema in 2nf.

3.3: NON-THIRD NORMAL FORM CLASS DIAGRAM

In this section we ascertain that the relations generated from the class-diagrams do not contain any functional dependency which violates the third-normal form.
Again, let R(A1,A2…………………….. Ak,Ak+1,………………………An) be a relational schema corresponding to a class C in the class diagram such that A1,A2…………………….. Ak is the primary key / candidate key (PK) of R and Ak+1,………………………An are the non-prime(NP)attributes of R.
Then R violates the 3NF when there exists atleast one functional dependency X→Y such that X,Y NP.
According to the given FD X→Y, the set of attributes X functionally determine the attribute set Y irrespective of R-XY. This shows that there should exist a separate class C' in the class-diagram such that the attribute set X is the primary key of the relation corresponding to C' which is denoted by RC' so that by using the definition of the primary key we get, X→XY i.e X→RC' .
We should therefore create a new class C' corresponding to this functional dependency and exhibit a 1:M relation between these two classes C and C'. The rule can be formally stated as below:

RULE 2:

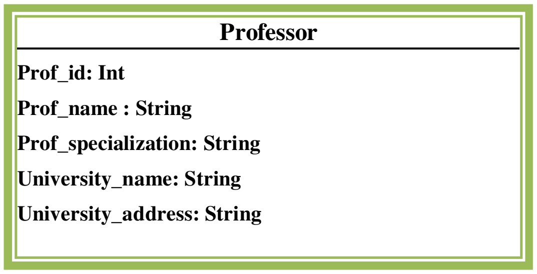
For every FD X→ Y ,where X,Y⊆ NP

(i) Create a new class C' in the class diagram with attributes X ⋃ Y and assign attribute set X as the key attribute of C' .

(ii) Set Attributes(C)=R-(X ⋃ Y). The PK attribute of the class C will remain the same.

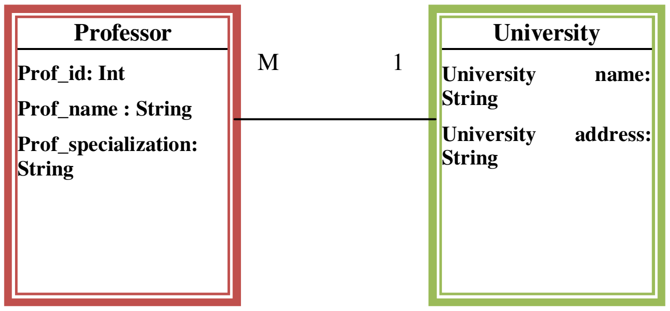
(iii) Set a 1:M relationship between the class C and C' such that class C is on the M side and class C' is on the 1 side.

Let us consider the following example from an Academic- system in which a class Professor is shown such that the following FDs hold on the class:



Prof_id → Prof_name
Prof_id → Prof_specialization
Prof_id → University_name
Univeristy_name → University_address

Here Prof_id is the primary key and let us assume that there is no candidate key. It can be easily noted that the FD University_name → University_address violates the definition of 3NF. Also note that the “University” is a separate entity and has its own existence . Therefore using Rule 2 , we create a new class “University” with the attributes University_name(PK) and University_address . The original class Professor will contain attributes Prof_id(PK) , Prof_name and Prof_specilization. We will also set a 1:M association relationship between the two classes as shown in the following Figure:



The new class diagram correctly represents the constraint that a professor in only one university but a university may employ more than one professors. We may also note that the relational mapping algorithm (section-2) will now create relations which are already in 3NF.

CONCLUSION

In this paper we have analyzed the causes of violation of the First three normal forms when a class model is mapped into database relations. We found that if aggregate data structures are avoided and multivalued attributes are mapped according to the relational mapping algorithm proposed by us then the relational schema so generated will always be in First normal form. We also found that if partial dependencies are analyzed carefully and the class model is changed suitably so as to accommodate some new classes and relationship (RULE 1), then the 2NF may always be achieved from the class model. We also found that our rule (RULE 2) will always lead to a 3NF normalized relational schema from a class model. We therefore conclude that if the methodology proposed by us is applied carefully at the time of conceptual modeling, then there is no need of the cumbersome normalization process based on the functional dependencies, and the time spent in normalizing relational schemas can be saved significantly.

FUTURE SCOPE

This paper proposes a methodology to eliminate the need of normalization process in relational database design for integrated DBMS of an Enterprise. In the present paper we have proposed two rules which eliminate the need of normalization upto 3NF based on the functional dependencies. But since BCNF is also based on 3NF, our focus in the future will be to propose a rule for eliminating the need of BCNF and also the other normal forms based on multivalued dependency(4NF) and join dependency(5NF). If we are successful in implementing these results then it may revolutionize the relational database theory as there will be no need left to do the normalization of the database.

Volume-1 Issue-2
July 2009-December 2009
Phase-II: Empirical Article

REFERENCES

[1] D. Bell, "UML Basics : *The class Diagram- An introduction to structure diagrams in UML 2.0*, IBM, http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/index.html

[2] P.P.Chen, " The Entity Relationship Model: towards a unified view of data", *ACM Transactions on Database Design*, Vol . 1, No. 1 , 1976.

[3] S. Dietrich, S.D.Urban, "An Advance course in Database Systems : Beyond Relational Databases", First Edition, Morgan Kaufmann Publication, 2005.

[4] R Elmasri and S. Navathe, "*Fundamentals of Database Systems*, Third Edition, Pearson Education Asia,2000.

[5] T. Hussain, S.Shamail and M.A. Awais " *Eliminating Process of Normalisation in Relational Database Design*", In Proc: 7th International INMIC Multitopic Conference, December 2003.

[6] W. Kent , "Consequences of assuming a universal relation. *ACM Transactions on Database Systems*, Vol. 6, No.4 ,1981.

[7] W. Kent, "The Universal relation revisited,*ACM Transactions on Database Systems*, Vol.8, No.4, 1983.

[8] E. J. Naiburg and R.A.Maksimchuk " *UML for database design*", First Edition, Addision Wesley,2001.

[9] J. Rumbaugh, G. Booch and I. Jacobson " *Unified Modeling Language: The User Guide*", Addision Wesley, 1999.

[10] Alexis Leon, "ERP Demystified", Second Edition, Tata Mcgraw Hill Publishing Company,2008.