An Ameliorated Approach towards Automating the Relational Database Normalization Process

P.B. Alappanavar, Dhiraj Patil, Radhika Grover, Srishti Hunjan, Yuvraj Girnar

Abstract— In the field of database systems in general and relational systems in particular, database design problem is of utmost importance as it helps to decide on a suitable logical structure for a given data to be represented in a database and concentrates on the fact that which relations are needed and what their attributes should be. Designing a database is a complex task and the normalization theory is a useful aid in the design process. In this paper, a tool is presented which aims to handle the normalization process up to third normal form(3NF) and even imparts the normalized relations to its users in different formats. The objective of the system is to provide an effective and efficient e-Learning system to individuals to learn normalization in an easy and quick manner and an impeccable solution for organizations dealing with excessive data.

Keywords— Database, Database Management System, Functional Dependencies, Normalization, Relations, Relational Model.

I. INTRODUCTION

In relational data model, the prime objective of a good database design is to create an accurate representation of the data, its relationship and its constraints. Database Normalization, a well developed field since the introduction of E.F. Codd's influential work on normal forms in 1970, assists to achieve this objective by organizing the data in a database by reducing data redundancies, inconsistent dependencies and eliminating data anomalies within the database.

In a nutshell, Normalization Theory assists in achieving the trademarks of a good database design which is mandatory for the long run success of any database being used by an organization- large or small scale.

II. MOTIVATION

Database Normalization is a fundamental topic in database theory but is unfortunately considered to be dry, ideological and purely theoretical by students. Consequently, the subject is not very well understood by them. A proficient and extremely competent staff is required to explain this process to students or even to manually carry out the normalization of databases in organizations.

Manuscript received April, 2013.

P.B. Alappanavar, Assistant Professor, Department of IT, STES's Sinhgad Academy of Engineering Pune-48, Maharashtra, India.

Dhiraj Patil, Department of IT, STES's Sinhgad Academy of Engineering Pune-48, Maharashtra, India.

Radhika Grover, Department of IT, STES's Sinhgad Academy of Engineering Pune-48, Maharashtra, India.

Srishti Hunjan, Department of IT, STES's Sinhgad Academy of Engineering Pune-48, Maharashtra, India.

Yuvraj Girnar, Department of IT, STES's Sinhgad Academy of Engineering Pune-48, Maharashtra, India.

Almost all the organizations experience the need to expand their databases over time. Thus, it becomes very difficult, time-consuming and impractical to manually handle the normalization process for expanding databases.

These scenarios have motivated us to develop a user-friendly, flexible and an efficient tool which will not only provide an interactive learning environment to students by giving a hands-on experience of the database normalization process but will also act as an asset for the Database Design Personnel of the leading organizations by automating the normalization process for their ever-growing databases. It provides organizations with database in a normalized form in an easy, quick and efficient manner. As a result, the data can be restructured and the database can grow without forcing the rewriting of application programs, which is of prime importance because of the excessive and growing costs of maintaining an organization's application programs and its data from the disrupting effects of database growth.

Our elaborate literature survey has proved that our system has an edge over all the other existing tools as none of them work on real world schemas and relationships or even provide their users with ready-to-use normalized relations whereas our tool not only lets the user define his own schema but also provides normalized relations to its users in three different formats viz. text file, SQL file and Entity-Relationship Diagram.

III. BACKGROUND

A. Functional Dependency

Functional Dependency (FD) is a term derived from the mathematical theory that underpins relational database theory[1]. It is a many-to-one relationship from one set of attributes to another within a given relation such that the values of one attribute depend on or are determined by the values of another attributes.

Given a relation R, attribute Y of R is functionally dependent on attribute X of R if and only if each X-value in R has associated with it precisely one Y-value in R at any one time. The left-hand side of the FD is called the determinant, and the right-hand side is the dependent[2]. The concept of FD is crucially important to a number of issues in the context of database design theory and is elaborated with the help of a simple real life scenario such as, a car's manufacturer and model can be determined if the serial number on the car is known.

Thus, manufacturer and model depends on the serial number and exhibits the following FD: serial number→ (manufacturer, model).

But in case of FDs, it is not necessary that the relationship works in reverse as well such that if it is known that a car has a given manufacturer, that information is not enough to determine its serial number since there are many individual cars, with different serial numbers, all made by the same manufacturer[3].

B. Types of Functional Dependency

The intricacies of Functional Dependency can be better understood with the help of a simple example. Consider the following relation,
REPORT (Student#, Course#, CourseName, IName, Room#, Marks, Grade)[4]
where,

- Student#- Student Number.
- Course#- Course Number.
- CourseName - Name of the course.
- IName- Name of the instructor who delivered the course.
- Room#- Room number which is assigned to respective instructor.
- Marks-Scored in Course-Course# by student-Student #
- Grade –Obtained by student-Student# in course-Course #

The set of functional dependencies(FDs) associated with the above relation are as:

Student#Course# → Marks
Course# → CourseName
Course# → IName (Assuming one course is taught by one and only one instructor)
IName → Room# (Assuming each instructor has his /her own and non-shared room)
Marks → Grade

Full Functional Dependency

In a relation, the attribute(s) Y is fully functional dependent on X if Y is functionally dependent on X, but not on any proper subset of X such that removal of any attribute or set of attributes 'Z' from 'X' doesn't make the dependency hold good[5].

In the above example, marks is fully functional dependent on Student#Course# as marks cannot be determined either by Student# or Course# alone. It can be determined only by using Student# and Course# together[4].

On the other hand, CourseName is not fully functionally dependent on Student#Course# as its subset Course# can individually determine the CourseName and Student# does not have any role in deciding the CourseName[4].

Partial Functional Dependency

In a relation, the attribute(s) Y is partially functional dependent on X if Y is functionally dependent on X and the dependency holds good even on removal of any attribute or set of attributes 'Z' from 'X'.

In the above relationship CourseName, IName, Room# are partially dependent on composite attribute Student#Course# as Course# can individually define the CourseName, IName, Room#[4].

Transitive Functional Dependency

A transitive dependency is an indirect functional dependency, one in which X→Z only by the virtue of X→Y and Y→Z[6].
In the above relationship, transitive dependency is exhibited as- Room# depends on IName and in turn depends on Course#. Hence, Room# transitively depends on Course#. Similarly, Grade depends on Marks which in turn depends on Student#Course# hence Grade transitively depends on Student#Course#[4].

C. Closure

Closure-Set of FDs

Certain FDs imply others[7]. From a given set of FDs F, the closure F^+ of that set is the set of all FDs that can be derived by the FDs in F by the application of Armstrong's Axioms. F^+ is necessarily a superset of F.
Armstrong's Axioms provide a complete system of inference rules for FDs[8]. They are as:

F1(Axiom of Reflexivity): If $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$.

F2 (Axiom of Augmentation): If $\alpha \rightarrow \beta$ then $\gamma.\alpha \rightarrow \gamma.\beta$

F3 (Axiom of Transitivity): If $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$.

Other useful rules can be derived from these. For instance,
F4 (Axiom of Pseudo transitivity): If $\alpha \rightarrow \beta$ and $\beta.\gamma \rightarrow \delta$, then $\alpha.\gamma \rightarrow \delta$.

F5 (Axiom of Decomposition): If $\alpha \rightarrow \beta$, then $\alpha \rightarrow \gamma$ for every γ in β. Thus, if $\alpha \rightarrow \delta\mu$, $\alpha \rightarrow \delta$ and $\alpha \rightarrow \mu$ holds true.
F6 (Union or Additive rule): if $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$, then $\alpha \rightarrow \beta.\gamma$ holds true.

Closure-Set of attributes
Given a subset Z of the set of attributes of relation R and a set F of FDs that holds for R, the closure Z^+ of Z under F is the set of all attributes A of R such that FD $Z \rightarrow A$ is a member of F^+.

D. Super Key

A superkey is an attribute or a set of attributes that uniquely identifies tuples within a relation such that two distinct tuples within a relation will always have distinct value for superkey. If Z^+ consists of all attributes of R, Z is said to be a superkey for R[7].

E. Candidate Key

A candidate key is a minimal set of attributes that can uniquely identifies tuples within a relation such that it acts as a minimal superkey, that is, a superkey for which no proper subset of it is also a superkey[4].

Every relation does have at least one candidate key. If there are multiple candidate keys, the Database Designer can designate any one of them as the Primary Key[9]. Thus all the attribute combinations inside a relation that can serve as a primary key are candidate keys as they are candidates for primary key position.

F. Prime Attribute

An attribute which appears as an attribute for some candidate key (It needn't be the primary key or even any key of interest)[10].

G. Non-Prime Attribute

An attribute which is never included in any candidate key[10].

H. Canonical Cover

The set of functional dependencies may have redundant dependencies that can be inferred from the other dependencies. A canonical cover for F is a given set of FDs F_c such that

1. F and F_c should be equivalent such that F logically implies all dependencies in F_c and F_c logically implies all dependencies in F.
2. F_c should contain no redundancy i.e. no functional dependency in F_c should contain an extraneous attribute.
3. Each left side of functional dependency F_c in should be unique.

Canonical Cover is also well known as minimal cover and non-redundant cover.

IV. APPROACH FOLLOWED TO AUTOMATE NORMALIZATION

Step 1: To determine the candidate keys associated with the relational schema R.
From the given set of FDs provided by the user,
Step 1.1 Gather the attributes which are present only on LHS (Determinant) of each FD.
Step 1.2 Gather the attributes which are present only on RHS (Dependent) of each FD.
Step 1.3 Gather the attributes which are present only on both LHS and RHS of each FD.
Step 1.4 Apply the following rules in order to compute the candidate key.
Rule 1.4.1: Take in all the attributes of relational schema R obtained in Step 1.1 and mark this as set Z. For this set Z of attributes obtain the closure set Z^+. If this closure set Z^+ contains all attributes of the given relation schema R then this forms the super key of R.
Rule 1.4.2: Never take in any attribute obtained in Step 1.2 to compute the candidate key of R.
Rule 1.4.3: If Rule1 does not give candidate key of R then add each possible combination of attributes obtain in step 1.3 to set of attributes obtained in step 1.1 and figure out the closure for each newly added combination. Include only the combinations whose closure set derives all the attributes of given relation R. Mark them as possible keys.
Rule 1.4.4: Choose a key having minimum number of attributes from the above obtained set of keys as candidate key is nothing but minimal super key.
Step 1.5: In this particular step, choose the Primary key as the Candidate Key with minimum number of attributes.

Step 2: Segregate the functional dependencies into full and partial functional dependencies in order to place the relation in 2NF. It is ensured that the relation is in 1NF since the system has been designed such that for each composite attribute GUI asks for the set of atomic attributes corresponding to composite attribute. Thus, it is assured that atomicity is preserved.

Step 3: Transform the relations in to 2NF by using Heath's theorem. A relational schema R is in Second Normal Form(2NF) if and only if it is in 1 NF and every non-key attribute is fully dependent on the primary key such that relations contains no partial dependencies[11].
Ian Heath in 1971 proved the following theorem-
Let relation R having heading H and let X, Y and Z be the subsets of H such that the union of X, Y and Z is equal to H. Let XY denote the union of X and Y, and similarly for XZ. If r satisfies the FD X→Y, then R is equal to the join of its projections on XY and XZ[12].

Step 4: Transform the relations into 3NF. A relation R is said to be in Third Normal Form(3NF) if and only if it is in 2NF and every non-key attribute is non-transitively dependent upon the primary key such that relations contains no transitive dependencies[13].
The following procedure can be employed to transform the relations into 3NF.

1. Determine the canonical cover.
2. Mark relations for each individual FDs. i.e. if Fc={AB->C, C->D} then make R1(ABC) & R2(CD).
3. Check if candidate key is contained in any one relation. If yes then it is in 3NF and if no then add one more relation that contains only that candidate key.

In this manner, the relations are normalized till 3NF. All the normal forms can be summarized as:

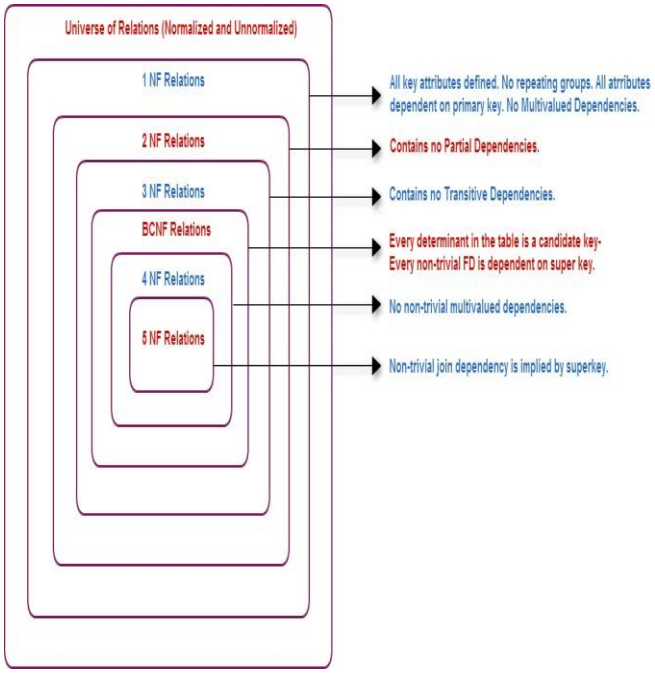

Fig.1 Different Normal Forms

V. SYSTEM FUNCTIONALITES AND FEATURES

Different strategies have been followed to make the web based system adaptable in all scenarios by giving students the ability to easily and efficiently test their knowledge of the different normal forms in practice and practitioners to use it during the actual implementation. Currently, the system has been designed for the two most frequently used relational databases-Oracle and MySQL.

1. The HOME page gives an outline of the system and a brief introduction on the fundamental concepts of Relational Database, Database Normalization and the various normal forms. This should immensely assist the users, especially the novice ones, in building their knowledge.
2. New users need to register with the system and the existing users login to the system in order to access the system features.
3. After successful login, the user needs to navigate to the 'Normalize' tab. The following details need to be

provided by the users as input in order to get the normalized relations:

- Relational Schema name.
- Attributes details- Name, Datatype and Constraints associated (if any).
- All the functional dependencies associated with the relational schema.

4. Employing a rigorous algorithm considering all the functional dependencies associated with the relational schema and other necessary details, the normalized relations are promptly computed up to third normal form (3NF).
5. The user instantly gets access to normalized relations in three different formats-Text file, SQL file and ERD.
6. A Feedback page has been employed to gather users' feedback which will assist us in creating the best user experience and help with future planning for improvements.

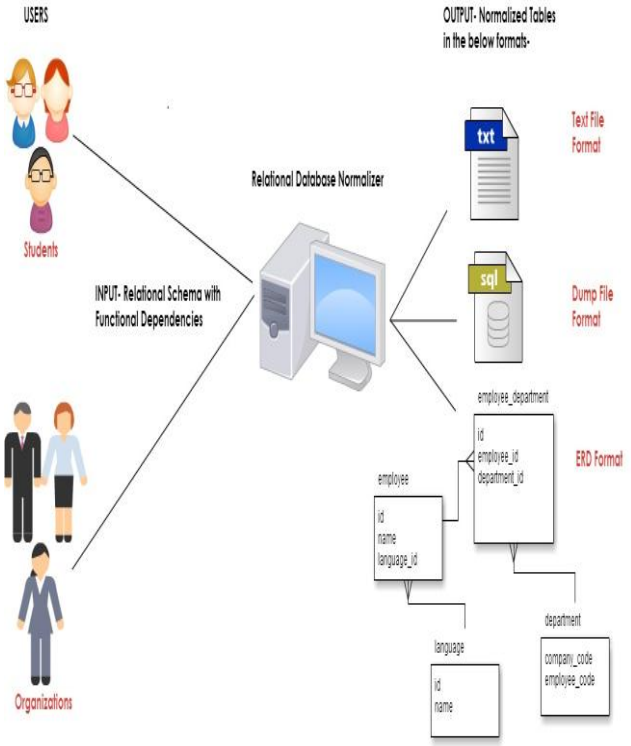

Fig.2 Top-Level Functionality Diagram

VI. TECHNOLOGIES USED

A. .NET Framework

.Net Framework, a Microsoft programming structure is a platform or development environment to seamlessly build, deploy and run web-applications and web services. The .NET Framework supports cross-language interoperability and consists of the common language runtime (CLR) and the .NET Framework class library, which includes classes, interfaces, and value types that support an extensive range of technologies[14]. The .NET Framework provides a managed execution environment, simplified development and deployment, and integration with a variety of programming languages, including Visual Basic and Visual C#[14].
Our system has been developed using Microsoft .NET Framework 4.5.

B.ASP.NET

ASP.NET (Active Server Pages .NET) is a web development technology from Microsoft. Part of the .NET framework, it provides developers with a free web framework for building great web sites and web applications using complied languages like VB.NET and C#, HTML, CSS and JavaScript.
In order to establish our system, ASP.NET 4.0 has been employed.

C. Visual C#

C# is a programming language developed by Microsoft Corporation as part of their .NET initiative in response to the success of Sun Microsystems' Java programming language[15]. It is designed for building a variety of applications that run on the .NET Framework[16]. C# is simple, powerful, type-safe, and object-oriented[16]. The many innovations in C# enable rapid application development while retaining the expressiveness and elegance of C-style languages as C# is primarily derived from the C, C++, and Java programming languages with some features of Microsoft's Visual Basic in the mix.

D. Microsoft Silverlight

Silverlight is a Microsoft technology aimed to help developers create rich interactive Web applications with the best user interface features and functionalities[17]. It's available as a plug-in for almost all famous browsers available today, and it's used to deliver the next generation media and Web applications[17]. It's said to be cross-platform, cross-browser, and cross-device. It can run on Windows, Linux, and even Mac, it can run on Internet Explorer, Mozilla Firefox, Google Chrome, and many others, and it also can run on PCs, mobile devices, and handhelds[17].

The ERD of the normalized relations has been developed using Microsoft Silverlight 5.0 which accounts for a rich and interactive look provided by the system.

VII. METHODOLOGY EMPLOYED

Methodology is a system of broad principles or rules from which specific methods or procedures may be derived to interpret or solve different problems within the scope of a particular discipline[18]. It is a set of practices. Our system uses Iterative Model as its methodology. The basic idea of Iterative model is that the software should be developed in increments, each increment adding some functional capability to the system until the full system is implemented[19]. It provides the fundamental structure that enables the effective management of software development.
The Iterative approach is widely accepted and has been chosen as it provides the following key benefits:

1. Quickly getting a useful system into the hands of users and the project team can learn along the way.
2. Clearer and more meaningful progress indicators which assists in better progress tracking and predictability resulting in a higher quality product with fewer defects.
3. Changing requirements and tactics can be more easily accommodated.
4. Increased Reusability.
5. Risks are mitigated earlier, because elements are integrated progressively[20].
6. Improving and refining the product is facilitated, resulting in a more robust product with better overall quality.

Iterations for the developed system

Iteration	Tasks Performed
1	1. Acceptance of input from the user. The following details are accepted as input from the user- • Relational Schema name along with number of attributes associated with it. • Attributes details- Name, Datatype and Constraints associated (if any). • All the functional dependencies associated with the relational schema. 2. Ensuring that the relations are in 1NF with the help of GUI. 3. Provision of normalized relations for the users in text file format.
2	1. Transforming the relations into 2NF by eliminating all the partial dependencies. 2. Provision of normalized relations for the users in text file and dump file format.
3	1. Transforming the relations into 3NF by eliminating all the transitive dependencies. 2. Provision of normalized relations for the users in text file and dump file format
4	Generating normalized relations for the users in ERD format using animation in Silverlight to enhance understand ability and reusability.

VIII. CASE STUDY

Consider the following relational schema 'Supply' as Supply(supplier_no, status, city, part_no, quantity) with a set of FDs satisfying on Supply given by-

i. supplier_no, part_no→ quantity.

ii. supplier_no → status.

iii. supplier_no→ city.

iv. city → status.

The various steps followed to achieve normalization is as shown below:

Step 1: The user provides the Relational Schema Name.

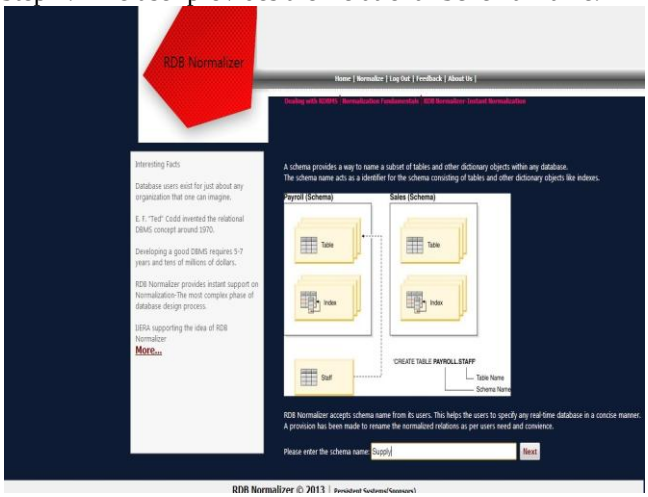

Fig.3 Acceptance of Relational Schema Name

Step 2: The user provides all the attributes associated with the Relational Schema 'Supply'

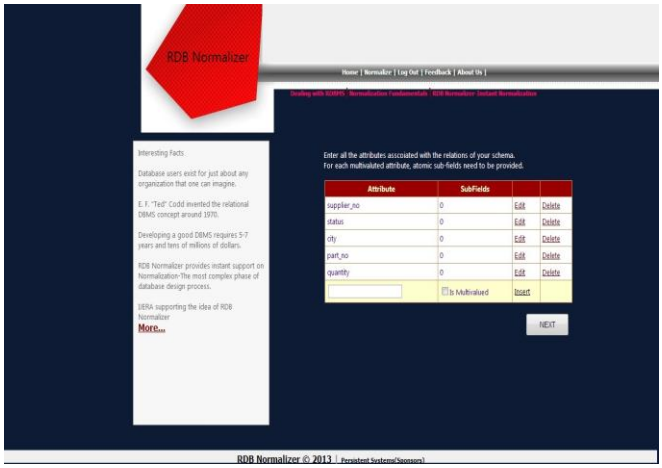

Fig.4 Acceptance of all the attributes associated with the Relational Schema

if the relational schema comprises of any multivalued attribute, all the sub-fields for the multivalued attribute are accepted in order to ensure atomicity of attributes. Thus, it is assured that the relational schema follows the norms of 1 NF. The user can further specify the datatype and constraints associated(if any) with the attributes. This feature assists the user to describe the relational schema in a flexible and efficient manner.

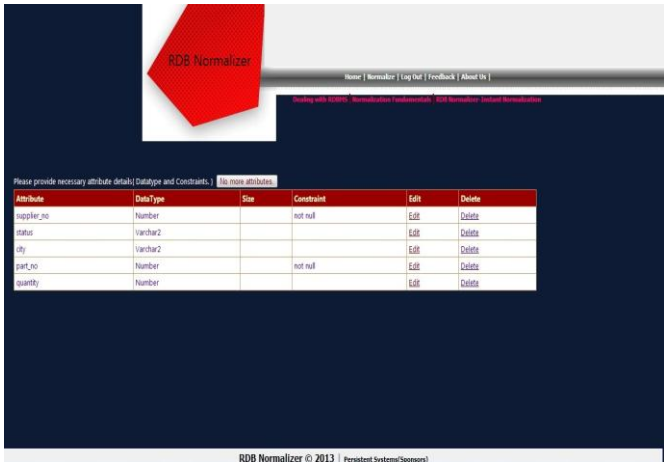

Fig.5 Acceptance of Datatype and the constraints associated with the attributes of Relational Schema.

Step 3: The user provides all the functional dependencies associated with the relational schema.

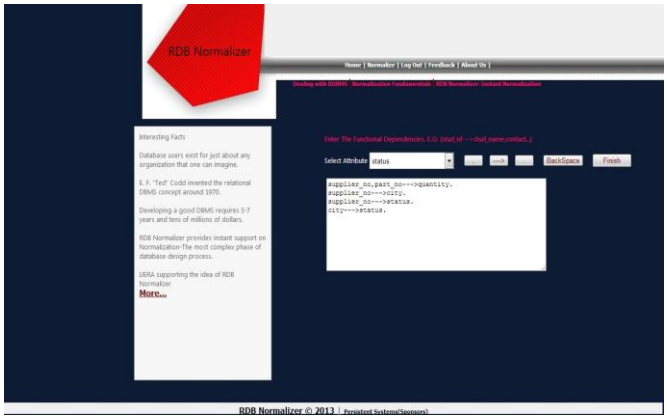

Fig.6 Acceptance of FDs from the user

Results: On the basis of FDs provided by the user, the relations are normalized upto 3NF. The normalized relations

are provided to the users in three different formats- Text File, SQL file and Entity Relationship Diagram.

Decomposition in 2NF-

In the Supply relational schema, non-key attributes are not mutually independent (city→ status) and are further not fully functionally dependent on the primary key (i.e. status and city are dependent on just part of the key, namely supplier_no)[13].

Thus, the decomposed relations are as-

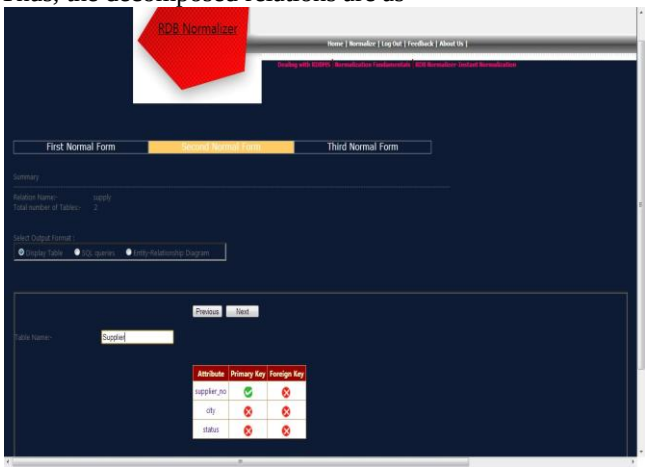

Fig.7 Normalized Relation(2NF)- Supplier

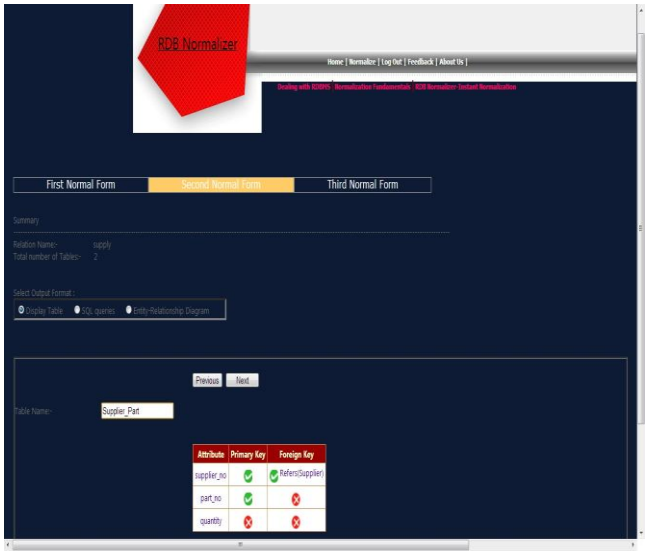

Fig.8 Normalized Relation(2NF)- Supplier_Part

```
---NORMALIZATION TOOL OUTPUT---
SQL FILE FOR YOUR NORMALIZED DATABASE

create or replace table Supplier( supplier_no number not null,city varchar2 ,status varchar2 ,primary key(supplier_no));

create or replace table Supplier_Part( supplier_no number not null,part_no number not null,quantity number ,foreign key(supplier_no) references Supplier( supplier_no )
,primary key(supplier_no),primary key(part_no));
```

Fig.9 SQL file for 2NF relations

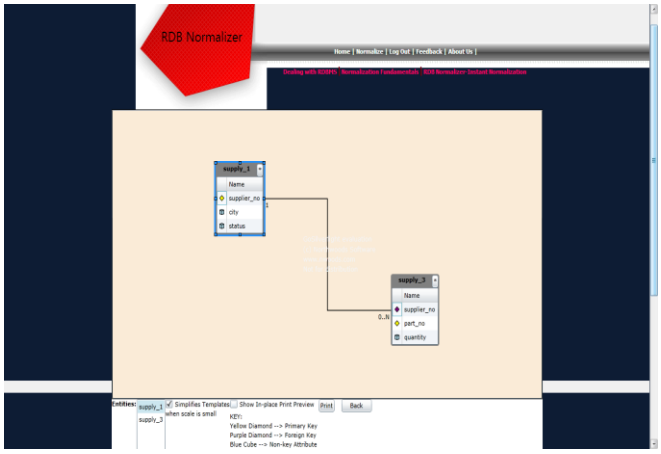

Fig.10 ERD for Relational Schema in 2NF

Decomposition in 3NF-

Supplier relation is not in 2NF as it lacks mutual independence among non-key attributes[13]

Mutual dependence is reflected in the transitive dependencies:

supplier_no→ city.

city → status.

Therefore, relations in 3NF are as-

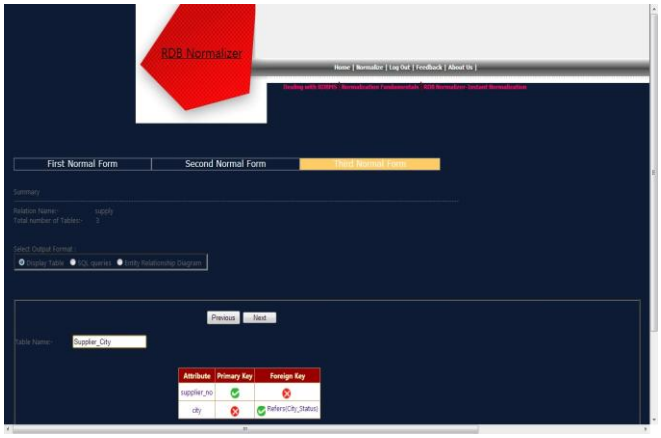

Fig.11 Normalized Relation(3NF)- Supplier_City

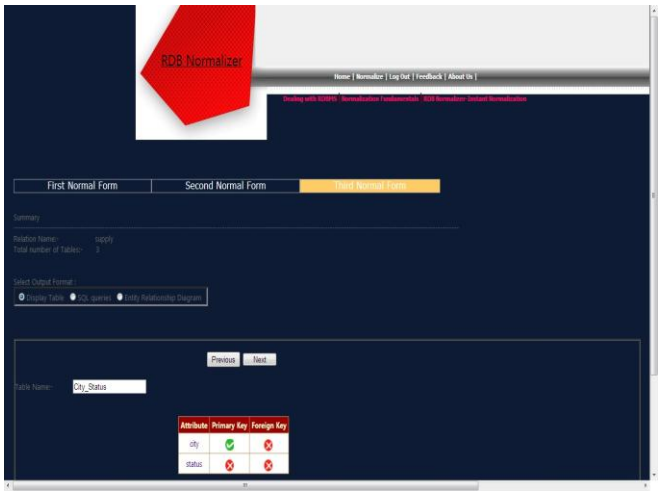

Fig.12 Normalized Relation(3NF)- City_Status

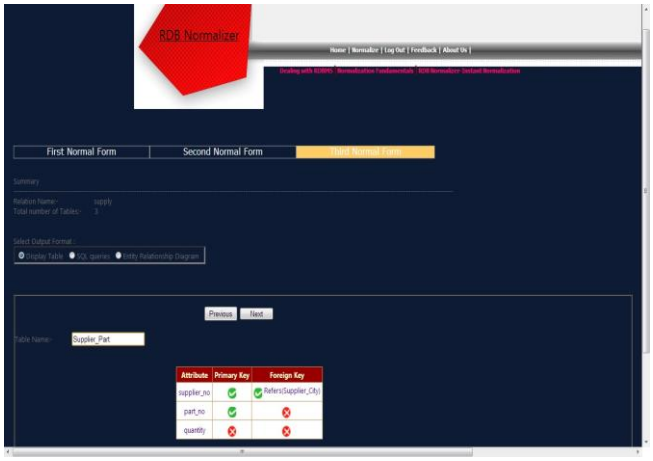

Fig.13 Normalized Relation(3NF)- Supplier_Part

Fig.14 SQL file for 3NF relations

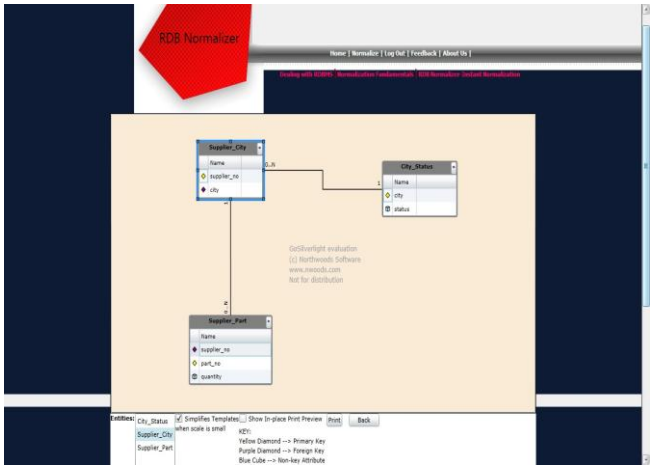

Fig.15 ERD for Relational Schema in 2NF

IX. EXPERIMENTATION

The dominant functionality of our system has been tested using several relations from distinct domains to ensure good performance and reliability. Few examples are shown below.

A. Description of standard relations used for experimentation

1. Supply
Relation Name: Supply
Attributes List: (supplier_no, status, city, part_no, quantity)[13]
No. of Attributes: 5
FDs:

1. supplier_no, part_no → quantity.
2. supplier_no → status.
3. supplier_no → city.
4. city → status.

No. of FDs: 3

2. Beer_Relation[21]
Relation Name: Beer_Relation

Attributes List: (beer, brewery, strength, city, region, warehouse, quantity)
No. of Attributes: 7
FDs:

1. beer → brewery, strength.
2. brewery → city.
3. city → region.
4. beer, warehouse → quantity.

No of FDs: 5

3. Invoice[21]
Relation Name: Invoice
Attributes List: (Order_ID, Order_Date, Customer_ID, Customer_Name, Customer_Address, Product_ID, Product_Description, Product_Finish, Unit_Price ,Order_Quantity)
No. of Attributes: 10
FDs

1. Order_ID → Order_Date, Customer_ID, Customer_Name, Customer_Address.
2. Customer_ID → Customer_Name, Customer_Address.
3. Product_ID → Product_Description, Product_Finish, Unit_Price.
4. Order_ID, Product_ID → Order_Quantity.

No of FDs: 10

4. Emp[22]
Relation Name: Emp
Attributes List: (emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrnname, skill_id, skill_name, skill_date, skill_lvl)
No. of Attributes: 10
FDs:

1. emp_id → emp_name, emp_phone.
2. emp_id → dept_name.
3. dept_name → dept_phone, dept_mgrnname.
4. skill_id → skill_name.
5. emp_id, skill_id → skill_date, skill_lvl.

No of FDs: 8

5. Project[21]
Relation Name: Project

Attributes List:(projectCode, project title, project manager, project budget, employeeNo, employeeName, deptNo, deptName, hourlyRate)
No. of Attributes: 9
FDs:

1. projectCode → project title, project manager, project budget.
2. employeeNo → employeeName, deptNo, deptName.
3. projectCode, employeeNo → hourlyRate.
4. deptNo → deptName.

No. of FDs: 8

6. Hospital[23]
Relation Name: Hospital
Attribute Details: (Patient_No, DrugNo, Start_Date, Full_Name, Ward_No, Ward_Name, Bed_No, Drug_Name, Description, Dosage, Method_Admin,
Units_Day, Finish_Date)
No. of Attributes: 13
FDs:

1. Patient_No → Full_Name.
2. Ward_No → Ward_Name, Bed_No.
3. Drug_No→ Drug Name, Description, Dosage, Method_Admin.
4. Patient_No , Drug_No, Start_Date → Units_Day, Finish_date, Ward_No.

No of FDs: 10

7. Report[24]
Relation Name: Report
Attributes List: (reportNo, editor, deptNo, deptName, deptAddress, authourId, authourName, authourAddress)

No. of Attributes: 8
FDs:

1. reportNo → editor, deptNo.
2. deptNo → deptName, deptAddress.
3. authourId → authourName, authourAddress.

No of FDs: 6

B. Experimentation Results

The output produced by our system examples collected from various research is shown below. Since the output generated matches with the expected results, our system is valid and works in the intended manner.

1. Supply
Decomposition in 2NF
i. Supplier(supplier_no, status, city)
ii. Supplier_Part(supplier_no, part_no, quantity)
Decomposition in 3NF
i. Supplier_City(supplier_no, city)
ii. City_Status(city, status)
iii. Supplier_Part(supplier_no, part_no, quantity)

2. Beer_Relation
Decomposition in 2NF
i. beer(beer, brewery, strength, city, region)
ii. beerwarehouse(beer, warehouse, quantity)
Decomposition in 3NF

i. beer(beer, brewery, strength)
ii. brewery(brewery, city)
iii. city(city, region)
iv. beerwarehouse(beer, warehouse, quantity)

3. Invoice
Decomposition in 2NF
i. OrderLine(order_id, product_id, ordered_qty)
ii. ProductID(product_id, product_desc, product_finish, unit_price)
iii. OrderID(order_id, order_date, customer_id, customer_name, customer_address)
Decomposition in 3NF
i. OrderLine(order_id, product_id, ordered_qty)
ii. Product(product_id ,product_desc, product_finish, unit_price)
iii. Order(order_id, order_date, customer_id)
iv. Customer(customer_id, customer_name, customer_address)

4. Emp
Decomposition in 2NF
i. empID(emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrnname)
ii. skill_id(skill_id, skill_name)
iii. emp_id skill_id (emp_id, skill_id,skill_date, skill_lvl)

Decomposition in 3NF
i. empID(emp_id, emp_name, emp_phone, dept_name)
ii. Dept(dept_name, dept_phone, dept_mgrnname)
iii. SkillID(skill_id, skill_name)
iv. EMP(emp_id, skill_id, skill_date, skill_lvl)

5. Project
Decomposition in 2NF
i. projectCode(ProjectCode, project title, project manager, project budget)
ii. employeeNo(employeeNo,employeeName,deptNo, deptName)
iii. projectCodeemployeeNo(projectCode, employeeNo, hourlyRate)
Decomposition in 3NF
i. projectCode(ProjectCode, project title, project manager, project budget)
ii. employeeNo(employeeNo,employeeName, deptNo)
iii. projectCodeemployeeNo(projectCode, employeeNo, hourlyRate)
iv. deptNo(deptNo, deptName)

6. Hospital
Decomposition in 2NF
i. Hospital (Patient_No, Drug_No, Start_Date, Ward_No, Ward_Name, Bed_No, Units_Day, Finish_Date)
ii. Drug(Drug_No, Name, Description, Dosage, Method_Admin)
iii. Patient(Patient_No, Full_Name)
Decomposition in 3NF
i. Hospital (Patient _No, Drug_No, Start_Date, Ward_No, Units_Day, Finish_Date)

ii. Drug (Drug_No, Name, Description, Dosage, Method_Admin)
iii. Patient (Patient_No, Full_Name)
iv. Ward (Ward_No, Bed_No, Ward_Name)

7. Report

Decomposition in 2NF

i. ReportNo(report_no, editor, dept_no, dept_name, dept_addr)
ii. Authorid(author_id, author_name, author_addr)

Decomposition in 3NF

i. ReportNo(report_no, editor, dept_no)
ii. DeptNo(dept_no, dept_name, dept_addr)
iii. Authorid(author_id, author_name, author_addr)

X. ADVANTAGES

Our tool aims to automate the most complex and elaborate phase of the database design process-Normalization, which will immensely help to achieve the trademarks of an acceptable database design.

1. It has a broader scope than any of the existing tools which aim to automate normalization as it caters to both individuals as well as the leading organizations.
2. It follows an extremely robust approach and even provides the normalized relations to the users in different formats acting as a boon for novice users.
3. Users can specify the entire schema in one go without having to break it down to tables. It also relieves the user from mentioning the primary, foreign and candidate keys as our efficient algorithm can calculate them automatically.
4. Overcoming the drawbacks of the previous systems, we provide the user with a practical output, unlike a simple text output displayed on the screen by the other systems.

XI. FUTURE SCOPE

The tool can be further elaborated to include features like-

1. Normalize the relations from the schema till 5NF in certain scenarios. Consequently, the tool will provide the following normal forms-
   i. First Normal Form (1 NF)
   ii. Second Normal Form (2 NF)
   iii. Third Normal Form (3 NF)
   iv. Boyce Codd Normal Form(BCNF)
   v. Fourth Normal Form (4 NF)
   vi. Fifth Normal Form (5 NF)

At present, the system has been restricted till third normal form (3NF) as it is quite sufficient for most business database design purposes and only few specialized applications may require the higher-level normalization. Although normalization is a very important database design component, always designing in the highest level of normalization is not desirable because at the physical implementation level, the decomposition of relations into higher normal form imply that more pointer movements are required to access and thus slower the response time of the database system[25]. This may conflict with the end user demand for fast performance[25]. Subsequently, the designer may sometimes have to denoralize some portions of a database design in order to meet performance requirements at the expense of data redundancy and its associated storage anomalies[25].

2. In order to use the system as Software-as-a-Service (SaaS), the application can be hosted on cloud. Saas is "pay-as-you-go" software delivery model in which software and the associated data is centrally hosted on the cloud[26]. It is extremely valuable as SaaS is a rapidly growing market as indicated in recent reports and will soon become commonplace within every organization.
3. To make the tool less time consuming for the users, we can add a provision for them to directly upload their schema instead of entering the data through the GUI.

XII. CONCLUSION

This tool has been developed keeping in mind the need of students and industrial personnel alike for a system which would ease the tedious process of Normalization by not only guiding and teaching the various normal forms but also accepting their real time schemas, normalizing it and also returning the decomposed tables in 3 formats viz. text file, SQL file and an ER Diagram. The system thus not only normalizes the user's schema but also aides in database designing.

The system helps to overcome all the shortcomings of all the existent/ previously developed systems by broadening its scope to database practitioners who would be using large real time schemas.

XIII. ACKNOWLEDGMENT

We are extremely thankful to our project guide Mrs. Pankaja Alappanavar and our mentors Mr. Paramananda Barik and Mrs. Varsha Pawar for their valuable guidance, advice, encouragement and whole hearted cooperation during this work. We are equally thankful to our HOD and staff members of Department of IT, STES's Sinhgad Academy of Engineering, Pune for their kind co-operation and support throughout which contributed tremendously to our work.

REFERENCES

[1] http://elm.eeng.dcu.ie/~ee221/EE221-DB-5.pdf
[2] http:///bccampus.pressbooks.com/dbdesign/chapter/chapter-11-functional-dependencies
[3] http://en.wikibooks.org/wiki/Relational_Database_Design/Normalization
[4] http://www.aliencoders.com/sites/default/files/normalization_resolved.pdf
[5] A Web Based Relational Database Design Tool to Perform Normalization- Radhakrishna Vangipuram(Associate Professor of CSE), Raju velpula (Department of CSE),V.Sravya (Department of CSE) -*International Journal of Wisdom Based Computing, Vol.1(3), December 2011*
[6] http://mytechnicalarticles.files.wordpress.com/2010/09/functional-dependencies-and-normalization-for-relation-databases.pdf
[7] An Introduction to Database Systems, Seventh Edition, C.J. Date
[8] http://toyhouse.cc/profiles/blogs/relational-database-design
[9] http://rdbms.opengrass.net/2_Database%20Design/2.1_TermsOfReference/2.1.2_Keys.html
[10] http://www.cs.miami.edu/~burt/learning/Csc598.073/notes/reldb.html
[11] http://www.cs.otago.ac.nz/cosc344/lectures/344-2012lect09.pdf
[12] Database Design and Relational Theory: Normal Forms and All That Jazz, By Chris Date
[13] http://www2.yk.psu.edu/~lxn/IST_210/normal_form_definitions.html
[14] http://msdn.microsoft.com/en-us/library/vstudio/w0x726c2.aspx
[15] http://en.csharp-online.net/CSharp_Overview
[16] http://msdn.microsoft.com/en-us/library/vstudio/kx37x362.aspx
[17] http://www.codeproject.com/Articles/200028/Introduction-to-Microsoft-Silverlight

[18] http://www.businessdictionary.com/definition/methodology.html
[19] An integrated approach to software engineering, third edition, By P. Jalote
[20] http://www.upedu.org/references/bestprac/im_bp1.htm
[21] RDBNorma: - A semi-automated tool for relational database schema normalization up to third normal formby Y. V. Dongare, P. S. Dhabe and S. V. Deshmukh- International Journal of Database Management Systems (IJDMS), Vol.3, No.1, February 2011
[22] Thomas C and Carolyn B (2005), "Database Systems", Pearson, third edition.
[23] O'nell P and O'nell E (2001), "Database Principles Programming and Performance", Harcourt, second edition.
[24] Teorey T. J.(2002), Database Modeling and Design", Harcourt, third edition.
[25] http://coronet.iicm.tugraz.at/Dbase1/scripts/rdbh04.htm
[26] http://www.saprock.com/saas_cloud_computing.html