International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

Study of Various Normal forms and Functional Dependency

Savita B. Chavan[1], Dr. B. B. Meshram[2]
[1]M.Tech Computer Engineering, V.J.T.I, Mumbai.
[2]Head of Computer Technology Department, V.J.T.I, Mumbai.
[1]savita.chavan08@yahoo.com
[2]bbmeshram@vjti.org.in

Abstract: **Normalization and functional dependency are most fundamental part in relational database. The normalization rules are designed to prevent update anomalies and data inconsistencies. The normal form defines in relation database theory represent the guidelines for record design. The guidelines corresponding to first through fifth normal form are presented. Normalization process depends on the single analytical tool called as functional dependency. The concept of functional dependency is useful in design and analysis of relational database. By applying the functional dependencies to relational database we can represent the relation in various normal forms.**

Keywords: Normalization, functional dependency , 1NF, 2NF, 3NF, Boyce-Codd normal form, 4NF, 5NF, multivalued dependency , joint dependency, full functional dependency, partial functional dependency.

I. INTRODUCTION

Normal forms defined in relational database theory represent the guidelines for record design [1]. The guidelines corresponding to first through fifth normal form. The normalization first proposed by codd [2] takes a relational schema through the series of test to certify where it satisfied the certain normal form [3]. Codd proposed three normal form which he called 1NF, 2NF, 3NF [1][2][3][4]. The strongest normal form 3NF is then defining as Boyce codd normal form [1][2][3]. He gives the ways to convert the relational schema not in given normal form into one of that by using normalization.

The all normal form are based on single analytical tool called functional dependency [1][2][3][4][5]. Later on fourth normal form (4NF)[1][2[3] and fifth normal form (5NF)[1][2][3] are dependent on the multivalued functional dependency[1][3] and joint functional dependency respectively[3][4]. The functional dependency is the properties of semantics. The database designer will use their understanding of semantics of attributes of relation i.e how they are related to one another to specify the functional dependency that should holds on all relation states of relational schema.

II. RELATED WORK

2. 1 Functional Dependency

We formalize the notation of FDs and their definition and also define the inference rules for functional dependency. A functional dependency is a constraint between two sets of attributes from the database.

Definition: A functional dependency (FD) is a statement of the form X → Y, where X and Y are sets of attributes. The FD X → Y is said to hold for a relation R if every pair of tuples of R that agrees on each of the attributes in X also agrees on the attributes in Y. That is, the FD X →Y holds for relation R if whenever s and t are tuples of R where s[X] = t[X], then s[Y] = t[Y].

Thus X functionally determines Y in the relation schema if and only if whenever two tuple of r(R) agrees on their x values, they must necessarily agree on their Y values. Note the following

- If the constraint on R states that there cannot be more than one tuple with given X values in any relation instance r(R)- that is X is the candidate key of R- this implies that X→Y for any subset of attribute Y of R .
- If X→Y in R, this does not say whether or not Y→X in R.

This can be proved by using example given below

Consider relation schema EMP_PROJ

From semantic of attribute we know the following functional dependencies hold

a) Ssn→Ename
b) Pnumber→{Pname, Plocation}
c) {Ssn,Pnumber}→Hours

2.2 Inference Rules for Functional dependency

Inference rules are means to construct these implied dependencies. A system of inference rules is said to be complete when every implied dependency can be derived by (repeated) applications of these rules. The following is a complete system of inference rules for functional dependencies:

- F1 (Reflexivity): If Y C X, then X→Y.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

- F2 (Augmentation): If Z C Wand X→ Y, then XW→ YZ.
- F3 (Transitivity): If X → Y and Y →Z, then X → Z.
- Other useful rules can be derived from these. For instance,
- F4 (Pseudotransitivity): If X → Y and YW →Z, then XW →Z.
- F5 (Decomposition): If X → Y, then X→ A for every A in Y.
- F6 (union, or additive, rule): {X →Y, X→Z} X →YZ.

III. Normal Form

In this paper we have discuss various types of norm normal form are as follows:

3.1 First normal Form

First normal form (1NF) is now considered to be part of the formal definition of a relation in the basic (flat) relational model, historically; it was defined to disallow multivalued attributes, composite attributes, and their combinations. It states that the domain of an attribute must include only atomic (simple, indivisible) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a single tuple.

The table cells must be of single value.

- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Example: Consider the relational schema student relational schema

Student

Student#	Student_name	Advisor	Class
1001	John Smith	jones	{101-2,112-01, 155-01}
1002	Cal smith	bob	{101-2, 102-3}

Fig 1: Student Relation schema

The above relational schema is not in 1NF because one student has several classes. To convert it into 1NF these classes should be display in separate table. The 1NF of above relation is

Student#	Student_name	Advisor	Class
1001	John Smith	jones	101-2
1001	John Smith	jones	112-01
1001	John Smith	jones	155-01
1002	Cal smith	bob	101-2
1002	Cal smith	bob	102-3

Fig2: Student Relational Schema in 1NF

3.2 Second Normal Form

Second Normal form based on the concept of full functional dependency.

Full Functional Dependency

A functional dependency X→ Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more; that is, for any attribute A ϵ X, (X - {A}) does not functionally determine Y.

Partial Functional Dependency

A functional dependency X → Y is a partial dependency if some attribute A X can be removed from X and the dependency still holds; that is, for some A ϵ X, (X - {A})→Y.

A relation schema R is in second normal form (2NF) if every nonprime attribute A in R is not partially dependent on any key of R. The relation is in first normal form and all nonkey attributes are functionally dependent on the entire primary key. The new term in the preceding is functionally dependent, a special relationship between attributes. To test for 2NF involves the testing for functional dependencies whose left hand side attributes are the part of primary key. If primary key contain single attributes the test need not be applied at all.

Example: Consider the following inventory record:

Part	Warehouse	Quantity	Warehouse_address

Fig 3: Inventory relational schema

The key here consists of the PART and WAREHOUSE fields together, but WAREHOUSE-ADDRESS is a fact about the WAREHOUSE alone. The basic problems with this design are:

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

- The warehouse address is repeated in every record that refers to a part stored in that warehouse.
- If the address of the warehouse changes, every record referring to a part stored in that warehouse must be updated.
- Because of the redundancy, the data might become inconsistent, with different records showing different addresses for the same warehouse.
- If at some point in time there are no parts stored in the warehouse, there may be no record in which to keep the warehouse's address.

To satisfy second normal form, the record shown above should be decomposed into (replaced by) the two records:

Part	Warehouse	Quantity

Warehouse	Warehouse_address

Fig 4: Relation in 2 NF

The above relational schema is in 2NF.

3.3 Third Normal form

Third normal form (3NF) is based on the concept of transitive dependency.

Transitive Functional Dependency: A functional dependency X→ Y in a relation schema R is a transitive dependency if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R and both X→ Z and Z→Y hold.

A relation schema R is in third normal form (3NF) if whenever nontrivial functional dependency X→A holds in R either

(a)X is the super key of R

(b)A is prime attribute of R

Third normal form is violated when a non key field is a fact about another non key field. A relation is 3NF if every attribute transitively dependent on a key is a key attribute.

The theoretical definition of third normal form says: The relation is in second normal form and there are no transitive dependencies.

Example: Consider employee relational schema which in 2NF

Emplyee no	Name	Department	Location

Fig 5: Relational schema not in 3NF

The EMPLOYEE field is the key. If each department is located in one place, then the LOCATION field is a fact about the DEPARTMENT in addition to being a fact about the EMPLOYEE. The problems with this design are the same as those caused by violations of second normal form:

- The department's location is repeated in the record of every employee assigned to that department.
- If the location of the department changes, every such record must be updated.
- Because of the redundancy, the data might become inconsistent, with different records showing different locations for the same department.
- If a department has no employees, there may be no record in which to keep the department's location.

To satisfy third normal form, the record shown above should be decomposed into the two records:

Employee_dept

Employee id	Name	Department

Dept_location

Depatment	location

Fig6: Relation in 3NF

3.4 Boyce-codd Normal form

Boyce-codd normal form was proposed as simpler of 3NF but it was found stricter than 3NF. A relation schema r is in BCNF if whenever a nontrivial functional dependency X→A holds in R then X is a super key of R. That is every relation in BCNF is also in 3NF but a relation in 3NF is not necessary in BCNF. A relation schema R is in BCNF if whenever a nontrivial functional dependency X→A holds in R then X is a super key.

Example: Consider the student relational schema as given below.

Student id	Major	Advisor
100	Math	Hilbert
150	History	Jung
200	Math	Couran
300	History	Ruth
300	communication	vasilaky

Fig 7: Student Relational Schema not in BCNF

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

Since Student_id repeats it can't be the Primary Key. We can choose either Student_id or Major as the Primary Key or Student_id and Advisor as Primary Key. Say we choose Student_id and Major as the Primary Key. But that means that the remaining field Advisor is a fact about both Student_id and Advisor. So we know that Hilbert is an advisor for math majors and advises student 100.

This table is in 3NF but it still has anomalies (inconstancies). It's in 2NF because the advisor is a fact about both the student advised and the major he/she advises. It's in 3NF because advisor is a fact only about the primary key (Student_id, Major).

Suppose student 300 drops out of school. If we delete student 300 we lose the fact that

Dr. Ruth Advises psychology. This is a deletion anomaly. Also how can we know that Dr. Freedman advises Economics until student major in Economics? This is an insertion anomaly. So we have inconsistent dependency. An attribute is a determinant if it determines another attribute. For example Student_id determines Major. Advisor determines the major she/he advises.

A table is in BCNF if it's in 3rd NF and every determinant can be used as a Primary Key.

In our example Advisor attribute determines Major but is not a possible Primary Key. Student_id and Advisor together is a possible (candidate) Primary Key.

Student

Student_id	Advisor
100	Hilbert
150	Jung
200	Couran
300	Ruth
300	vasilaky

Advisors

Advisor	Major
Hilbert	Math
Jung	History
Couran	Math
Ruth	History
Vasilaky	communication

Fig 8: Relation in BCNF

IV. Higher Normal Forms

4.1 Multivalued Dependencies and Fourth Normal Form

So far we have discussed only functional dependency, which is by far the most important type of dependency in relational database design theory. However, in many cases relations have constraints that cannot be specified as functional dependencies. In this section, we discuss the concept of multivalued dependency (MVD) and define fourth normal form, which is based on this dependency. Multivalued dependencies are a consequence of first normal form (1NF) which disallowed an attribute in a tuple to have a set of values. If we have two or more multivalued independent attributes in the same relation schema, we get into a problem of having to repeat every value of one of the attributes with every value of the other attribute to keep the relation state consistent and to maintain the independence among the attributes involved. This constraint is specified by a multivalued dependency.

4.1.1 Formal definition of multivalued Dependency

Definition: A multivalued dependency X→→ Y specified on relation on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that t1[X]=t2[X], then two tuples t3 and t4 should also exist in r with the following properties where we use Z to denote (R-(X U Y)) that is Z is shorthand of the attributes in R after the attributes in (X U Y) are removed from R.

1)t3[X]=t4[X]=t1[X]=t2[X]
2)t3[Y]=t1[Y] and t4[Y]=t2[Y]
3)t3[Z]=t2[Z] and t4[Z]=t1[Z]

Whenever X→→Y holds, we can say that x multidetermines Y. because of the symmetry in the definition, whenever X→→Y holds in R, so does X→→Z. Hence, X→→Y implies x→→Z and therefore it is sometimes written as X→→Y|Z.

4.1.2Fourth Normal Form:

We now present the definition of **fourth normal form (4NF),** which is violated when a relation has undesirable multivalued dependencies, and hence can be used to identify and decompose such relations. A relation schema R is in 4NF with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every nontrivial multivalued dependency X →→Y in F^+, X is a super key for R.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

Example:
The EMP relation of Fig 9 below is not in 4NF because in the nontrivial MVDs ENAME PNAME and ENAME DNAME, ENAME is not a super key of EMP. We decompose EMP into EMP_PROJECTS and EMP_DEPENDENTS, shown in Figure 10. Both EMP_PROJECTS and EMP_ DEPENDENTS are in 4NF, because the MVDs ENAME PNAME in EMP_PROJECTS and ENAME DNAME in EMP_DEPENDENTS are trivial MVDs. No other nontrivial MVDs hold in either EMP_PROJECTS or EMP_DEPENDENTS. No FDs hold in these relation schemas either.

EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	Y	John
Brown	X	Jim
Brown	X	Jim
Brown	Y	Joan

Fig 9: EMP Relation not in 4NF

EMP_PROJECT

Ename	Pname
Smith	X
Smith	Y
Brown	X
Brown	Y

EMP_DEPENDENT

Ename	Dname
Smith	John
Smith	Anna
Brown	Jim
Brown	Joan

Fig 10: Two corresponding 4NF relations EMP_PROJECT and EMP_DEPENDENT

4.2 Join Dependencies and Fifth Normal Form
We know that LJ1 and LJ1' give the condition for a relation schema R to be decomposed into two schemas and, where the decomposition has the lossless join property. However, in some cases there may be no lossless join decomposition of R into two relation schemas but there may be lossless join decomposition into more than two relation schemas. Moreover, there may be no functional dependency in R that violates any normal form up to BCNF and there may be no nontrivial MVD present in R either that violates 4NF.

We then resort to another dependency called the join dependency and if it is present, carry out a multiway decomposition into fifth normal form (5NF).
A **join dependency** (**JD**), denoted by JD (R1, R2 …Rn), specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have lossless join decomposition into R1, R2 …Rn; that is, for every such r we have
$*(\Pi_{R1}(r), \Pi_{R2}(r),\ldots\ldots\ldots\ldots.., \Pi_{Rn}(r)) = r$

An MVD is a special case of a JD where
n = 2. That is, a JD denoted as JD (R1, R2) implies an MVD (R1∩ R2)→→ (R1 – R2). A join dependency JD (R1, R2 …Rn), specified on relation schema R, is a **trivial JD** if one of the relation schemas in JD (R1, R2 …Rn), is equal to R. Such a dependency is called trivial because it has the lossless join property for any relation state r of R and hence does not specify any constraint on R. We can now define fifth normal form, which is also called project-join normal form.

Definition: A relation schema R is in **fifth normal form (5NF)** (or **project-join normal form (PJNF)**) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD (R1, R2 …Rn), in F^+ every R_i is a super key of R.
Example:
For an example of a JD, consider once again the SUPPLY all-key relation of Figure11. Suppose that the following additional constraint always holds: Whenever a supplier s supplies part p, and a project j uses part p, and the supplier s supplies at least one part to project j, then supplier s will also be supplying part p to project j. This constraint can be restated in other ways and specifies a join dependency JD (R1, R2, R3) among the three projections R1 (Sname, Part_name), R2 (Sname, Proj_name), and R3 (Part_name, Proj_name) of SUPPLY. If this constraint holds, the tuples below the dotted line in Figure (a) must exist in any legal state of the SUPPLY relation that also contains the tuples above the dotted line. Figure (b) shows how the SUPPLY relation with the join dependency is decomposed into three relations R1, R2, and R3 that are each in 5NF. Applying NATURAL JOIN to any two of these relations produces spurious
Tuples, but applying NATURAL JOIN to all three together does not.
The reader should verify this on the example relation of Figure (a) and its projections in Figure (b). This is because only the JD exists, but no MVDs are specified. The JD (R1, R2, and R3) is specified on all legal relation states, not just on the show in fig below.

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 7, July 2012)

SUPPLY

Sname	Part name	Proj name
Smith	Bolt	projX
Smith	Nut	projY
Adamsky	Bolt	projY
Walton	Nut	projZ
Adamsky	Bolt	projX
Adamsky	Nail	projX
Smith	Bolt	projY

Fig 11: SUPPLY Relation with no MVD's is in 4NF but not in 5NF

Decomposing the relation SUPPLY into 5NF R1, R2 and R3

R1

Part name	Proj name
Bolt	projX
Nut	projY
Bolt	projY
Nut	projZ
Nail	projX

R2

Sname	Proj name
Smith	projX
Smith	projY
Adamsky	projY
Walton	projZ
Adamsky	projX

R3

Sname	Part name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

Fig 12: Relation is in 5NF

V. CONCLUSION

Thus we have tried to present the various normal forms and functional dependencies in simple and understandable way; we are by means of suggesting a data design process is correspondingly simple. The design process involves many complexities that are quite beyond the scope of this paper. In first place an initial set of data element and records has to be developed as the candidate for normalization.

The factor affecting normalization are as follows:

- single valued and multivalued fact
- dependency on entire key
- independent versus dependent fact
- the presence of mutual constraint
- the presence of non unique and non singular representation

The desirable of normalization has to be assessed in term of its performance response on retrieval application

References:

[1]. Shamkant B. Navate, "Fundamentals of Database management system", Pearson Publication.
[2]. William ketty, "A Simple Guide to Five Normal Form in Relational Database Theory", Communications of the ACM February 1983 Volume 26 Number 2.
[3]. IMark Levene and Millist W. Vincent, "Justification for Inclusion Dependency Normal Form", IEEE transaction on knowledge and data engineering, VOL. 12, NO. 2, MARCH/APRIL 2000.
[4]. CARLO ZANIOLO ,Sperry Research Center, " A New Normal Form for the Design of Relational Database Schemata", ACM Transactions on Database Systems, Vol. 7, No. 3, September 1982
[5]. C. J. DATE and RONALD FAGIN," Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases", ACM Transactions on Database Systems, Vol. 17, No. 3, September 1992.
[6]. Y. V. Dongare, P. S. Dhabe and S. V. Deshmukh, "RDBNorma: - A semi-automated tool for relational database schema normalization up to third normal form", International Journal of Database Management Systems (IJDMS), Vol.3, No.1, February 2011.
[7]. Radhakrishna Vangipuram Raju velpula V.Sravya," A Web Based Relational Database Design Tool to Perform Normalization", International Journal of Wisdom Based computing, Vol. 1(3), December 2011.
[8]. S. Y. Liao, H. Q. Wang, and W. Y. Liu, " Correspondence", IEEE Transaction on fuzzy system, VOL. 7, NO. 1, FEBRUARY 1999.
[9]. RONALD FAGIN," Multivalued Dependencies and a New Normal Form for Relatknal Databases", ACM Transactions on Database Systems, Vol. 2, No. 3, September 1977.