Normalization-Part III

Hi! Now we are going to discuss the reasons for the higher normal forms like fourth and fifth normal form.

MULTIVALUED DEPENDENCIES

Recall that when we discussed database modelling using the E-R Modelling technique, we noted difficulties that can arise when an entity has multivalue attributes. It was because in the relational model, if all of the information about such entity is to be represented in one relation, it will be necessary to repeat all the information other than the multivalue attribute value to represent all the information that we wish to represent. This results in many tuples about the same instance of the entity in the relation and the relation having a composite key (the entity id and the mutlivalued attribute). Of course the other option suggested was to represent this multivalue information in a separate relation.

The situation of course becomes much worse if an entity has more than one multivalued attributes and these values are represented in one relation by a number of tuples for each entity instance such that every value of one the multivalued attributes appears with every value of the second multivalued attribute to maintain consistency. The multivalued dependency relates to this problem when more than one multivalued attributes exist. Consider the following relation that represents an entity employee that has one mutlivalued attribute *proj*:

emp (e#, dept, salary, proj)

We have so far considered normalization based on functional dependencies; dependencies that apply only to single-valued facts. For example, $c\# \rightarrow dept$ implies only one *dept* value for each value of *e#*. Not all information in a database is single-valued, for example, *proj* in an employee relation may be the list of all projects that the employee is currently working on. Although *e#* determines the list of all projects that an employee is working on, $c\# \rightarrow proj$ is not a functional dependency.

So far we have dealt with multivalued facts about an entity by having a separate relation for that multivalue attribute and then inserting a tuple for each value of that fact. This resulted in composite keys since the multivalued fact must form part of the key. In none of our examples so far have we dealt with an entity having more than one multivalued attribute in one relation. We do so now.

The fourth and fifth normal forms deal with multivalued dependencies. Before discussing the 4NF and 5NF we discuss the following example to illustrate the concept of multivalued dependency.

programmer (emp_name, qualifications, languages)

The above relation includes two multivalued attributes of entity *programmer*; *qualifications* and *languages*. There are no functional dependencies.

The attributes qualifications and languages are assumed independent of each other. If we were to consider *qualifications* and *languages* separate entities, we would have two relationships (one between *employees* and *qualifications* and the other between *employees* and programming *languages*). Both the above relationships are many-to-many i.e. one programmer could have several qualifications and may know several programming languages. Also one qualification may be obtained by several programmers and one programming language may be known to many programmers.

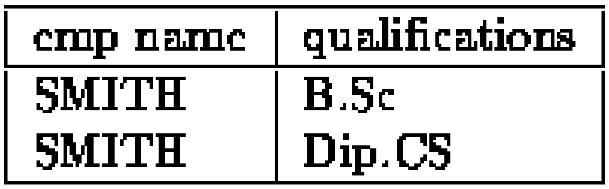
The above relation is therefore in 3NF (even in BCNF) but it still has some disadvantages. Suppose a programmer has several qualifications (B.Sc, Dip. Comp. Sc, etc) and is proficient in several programming languages; how should this information be represented? There are several possibilities.

emp name	qualifications	languages
SMITH	B.Sc	FORTRAN
SMITH	B.Sc	COBOL
SMITH	B.Sc	PASCAL
SMITH	Dip.CS	FORTRAN
SMITH	Dip.CS	COBOL
SMITH	Dip.CS	PASCAL

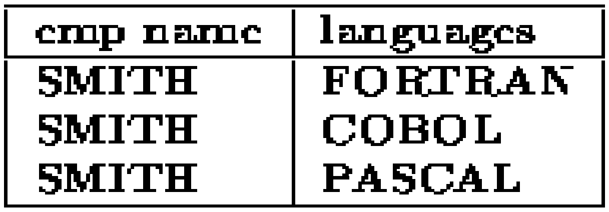
emp name	qualifications	language
SMITH	B.Sc	NULL
SMITH	Dip.CS	NULL
SMITH	NULL	FORTRAN
SMITH	NULL	COBOL
SMITH	NULL	PASCAL

emp name	qualifications	language
SMITH	B.Sc	FORTRAN
SMITH	Dip.CS	COBOL
SMITH	NULL	PASCAL

Other variations are possible (we remind the reader that there is no relationship between qualifications and programming languages). All these variations have some disadvantages. If the information is repeated we face the same problems of repeated information and anomalies as we did when second or third normal form conditions are violated. If there is no repetition, there are still some difficulties with search, insertions and deletions. For example, the role of NULL values in the above relations is confusing. Also the candidate key in the above relations is (emp name, qualifications, language) and existential integrity requires that no NULLs be specified. These problems may be overcome by decomposing a relation like the one above as follows:



emp name	qualifications
SMITH	B.Sc
SMITH	Dip.CS



emp name	languages
SMITH	FORTRAN
SMITH	COBOL
SMITH	PASCAL

The basis of the above decomposition is the concept of multivalued dependency (MVD). Functional dependency $A \rightarrow B$ relates one value of *A* to one value of *B* while multivalued dependency $A \twoheadrightarrow B$ defines a relationship in which a set of values of attribute *B* are determined by a single value of *A*.

The concept of multivalued dependencies was developed to provide a basis for decomposition of relations like the one above. Therefore if a relation like *enrolment(sno, subject#)* has a relationship between *sno* and *subject#* in which *sno* uniquely determines the values of *subject#*, the dependence of *subject#* on *sno* is called a *trivial* MVD since the relation enrolment cannot be decomposed any further. More formally, a MVD $X \twoheadrightarrow Y$ is called trivial MVD if either *Y* is a subset of *X* or *X* and *Y* together form the relation *R*. The MVD is trivial since it results in no constraints being placed on the relation. Therefore a relation having non-trivial MVDs must have at least three attributes; two of them multivalued. Non-trivial MVDs result in the relation having some constraints on it since all possible combinations of the multivalue attributes are then required to be in the relation.

Let us now define the concept of multivalued dependency. *The multivalued dependency* $X \twoheadrightarrow Y$ *is said to hold for a relation R(X, Y, Z) if for a given set of value (set of values if X is more than one attribute) for attributes X, there is a set of (zero or more) associated values for the set of attributes Y and the Y values depend only on X values and have no dependence on the set of attributes Z.*

In the example above, if there was some dependence between the attributes *qualifications* and *language*, for example perhaps, the language was related to the qualifications (perhaps the qualification was a training certificate in a particular language), then the relation would not have MVD and could not be decomposed into two relations as abve. In the above situation whenever $X \twoheadrightarrow Y$ holds, so does $X \twoheadrightarrow Z$ since the role of the attributes *Y* and *Z* is symmetrical.

Consider two different situations.

(a) *Z* is a single valued attribute. In this situation, we deal with *R(X, Y, Z)* as before by entering several tuples about each entity.
(b) *Z* is multivalued.

Now, more formally, $X \twoheadrightarrow Y$ is said to hold for *R(X, Y, Z)* if *t1* and *t2* are two tuples in *R* that have the same values for attributes *X* and therefore with *t1[x] = t2[x]* then *R* also contains tuples *t3* and *t4* (not necessarily distinct) such that

t1[x] = t2[x] = t3[x] = t4[x]
t3[Y] = t1[Y] and *t3[Z] = t2[Z]*
t4[Y] = t2[Y] and *t4[Z] = t1[Z]*

In other words if *t1* and *t2* are given by

t1 = [X, Y1, Z1], and
t2 = [X, Y2, Z2]

then there must be tuples *t3* and *t4* such that

t3 = [X, Y1, Z2], and
t4 = [X, Y2, Z1]

We are therefore insisting that every value of *Y* appears with every value of *Z* to keep the relation instances consistent. In other words, the above conditions insist that *Y* and *Z* are determined by *X* alone and there is no relationship between *Y* and *Z* since *Y* and *Z* appear in every possible pair and hence these pairings present no information and are of no significance. Only if some of these pairings were not present, there would be some significance in the pairings.

Give example (instructor, quals, subjects) --- explain if subject was single valued; otherwise all combinations must occur. Discuss duplication of info in that case.

(Note: If *Z* is single-valued and functionally dependent on *X* then *Z1* = *Z2*. If *Z* is multivalue dependent on *X* then *Z1* <> *Z2*).

The theory of multivalued dependencies in very similar to that for functional dependencies. Given *D* a set of MVDs, we may find D^+, the closure of *D* using a set of axioms. We do not discuss the axioms here. (Interested reader is referred to page 203 Korth & Silberschatz or Ullman).

Multivalued Normalization --- Fourth Normal Form

Def: **A table is in 4NF if it is in BCNF and if it has no multi-valued dependencies.**

We have considered an example of *Programmer(Emp name, qualification, languages)* and discussed the problems that may arise if the relation is not normalised further. We also saw how the relation could be decomposed into *P1(Emp name, qualifications)* and *P2(Emp name, languages)* to overcome these problems. The decomposed relations are in fourth normal form (4NF) which we shall now define.

We are now ready to define 4NF. A relation *R* is in 4NF if, whenever a multivalued dependency $X \rightarrow Y$ holds then either

(a) the dependency is trivial, or
(b) *X* is a candidate key for *R*.

As noted earlier, the dependency $X \twoheadrightarrow \phi$? or $X \twoheadrightarrow Y$ in a relation *R(X, Y)* is trivial since they must hold for all *R(X, Y)*. Similarly $(X, Y) \rightarrow Z$ must hold for all relations *R(X, Y, Z)* with only three attributes.

In fourth normal form, we have a relation that has information about only one entity. If a relation has more than one multivalue attribute, we should decompose it to remove difficulties with multivalued facts.

Intuitively *R* is in 4NF if all dependencies are a result of keys. When multivalued dependencies exist, a relation should not contain two or more independent multivalued attributes. The decomposition of a relation to achieve 4NF would normally result in not only reduction of redundancies but also avoidance of anomalies.

<u>Fifth Normal Form</u>

Def: A table is in 5NF, also called "Projection-Join Normal Form" (PJNF), if it is in 4NF and if every join dependency in the table is a consequence of the candidate keys of the table.

The normal forms discussed so far required that the given relation *R* if not in the given normal form be decomposed in two relations to meet the requirements of the normal form. In some rare cases, a relation can have problems like redundant information and update anomalies because of it but cannot be decomposed in two relations to remove the problems. In such cases it may be possible to decompose the relation in three or more relations using the 5NF.

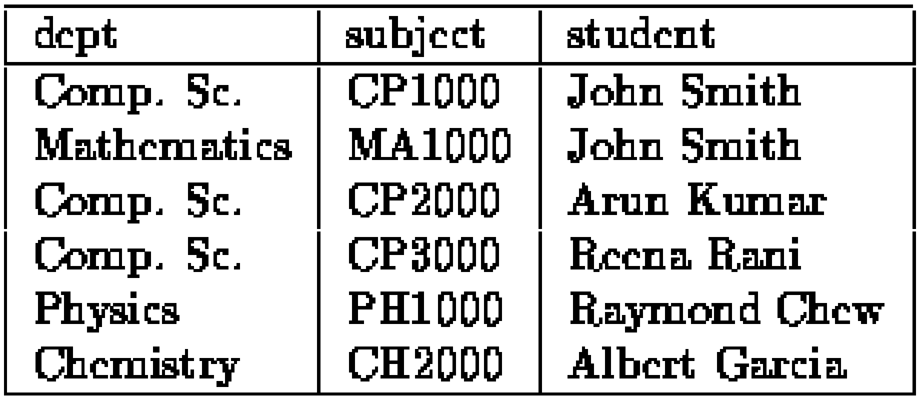
The fifth normal form deals with join-dependencies which is a generalisation of the MVD. The aim of fifth normal form is to have relations that cannot be decomposed further. A relation in 5NF cannot be constructed from several smaller relations.

A relation *R* satisfies *join dependency* *(*$R_1, R_2, \ldots R_n$*)* if and only if *R* is equal to the join of $R_1, R_2, \ldots R_n$ where R_i are subsets of the set of attributes of *R*.

A relation *R* is in 5NF (or project-join normal form, PJNF) if for all join dependencies at least one of the following holds.

(a) ($R_1, R_2, \ldots R_n$) is a trivial join-dependency (that is, one of R_i is *R*)
(b) Every R_i is a candidate key for *R*.

An example of 5NF can be provided by the example below that deals with departments, subjects and students.



dept	subject	student
Comp. Sc.	CP1000	John Smith
Mathematics	MA1000	John Smith
Comp. Sc.	CP2000	Arun Kumar
Comp. Sc.	CP3000	Reena Rani
Physics	PH1000	Raymond Chew
Chemistry	CH2000	Albert Garcia

The above relation says that Comp. Sc. offers subjects CP1000, CP2000 and CP3000 which are taken by a variety of students. No student takes all the subjects and no subject has all students enrolled in it and therefore all three fields are needed to represent the information.

The above relation does not show MVDs since the attributes *subject* and *student* are not independent; they are related to each other and the pairings have significant information in them. The relation can therefore not be decomposed in two relations

(dept, subject), and
(dept, student)

without loosing some important information. The relation can however be decomposed in the following three relations

(dept, subject), and
(dept, student)
(subject, student)

and now it can be shown that this decomposition is lossless.

Domain-Key Normal Form (DKNF)

Def: A table is in DKNF if every constraint on the table is a logical consequence of the definition of keys and domains.

Review Questions

1. What do you mean by Normalization?
2. Explain BCNF?
3. Explain 4NF and 5NF?
4. Explain Domain Key Normal Form?

REFERENCES

1. Aho, A. V. and C. Beeri and J. D. Ullman, "The Theory of Joins in Relational Databases", ACM-TODS, Vol 4, No 3, Sept 1979, pp. 297-314.
2. Fagin, R. (1981), "A Normal Form for Relational Databases that is Based on Domains and Keys", ACM-TODS, Vol 6, No 3, Sept 1981, pp 387-415.
3. Beeri, C, and P. A. Bernstein (1979), "Computational Problems Related to the Design of Normal Form Relational Schemas", ACM-TODS, Vol 4, No 1, Sept 1979, pp 30-59.
4. Kent, W. (1983), "A Simple Guide to Five Normal Forms in Relational Database Theory", Comm ACM, Vol 26, No 2, Feb 1983, pp. 120-125.
5. Bernstein, P. A. (1976), "Synthesizing Third Normal Form Relations from Functional Dependencies", ACM-TODS, Vol. 1, No. 4, Oct. 76, pp. 277-298.