A SIMPLE GUIDE TO FIVE NORMAL FORMS IN RELATIONAL DATABASE THEORY

WILLIAM KENT *International Business Machines Corporation*

SUMMARY: The concepts behind the five principal normal forms in relational database theory are presented in simple terms.

1. INTRODUCTION

The normal forms defined in relational database theory represent guidelines for record design. The guidelines corresponding to first through fifth normal forms are presented, in terms that do not require an understanding of relational theory. The design guidelines are meaningful even if a relational database system is not used. We present the guidelines without referring to the concepts of the relational model in order to emphasize their generality and to make them easier to understand. Our presentation conveys an intuitive sense of the intended constraints on record design, although in its informality it may be imprecise in some technical details. A comprehensive treatment of the subject is provided by Date [4].

The normalization rules are designed to prevent update anomalies and data inconsistencies. With respect to performance trade-offs, these guidelines are biased toward the assumption that all nonkey fields will be updated frequently. They tend to penalize retrieval, since data which may have been retrievable from one record in an unnormalized design may have to be retrieved from several records in the normalized form. There is no obligation to fully normalize all records when actual performance requirements are taken into account.

2. FIRST NORMAL FORM

First normal form [1] deals with the "shape" of a record type. Under first normal form, *all occurrences of a record type must contain the same number of fields.* First normal form excludes variable repeating fields and groups. This is not so much a design guideline as a matter of definition. Relational database theory does not deal with records having a variable number of fields.

3. SECOND AND THIRD NORMAL FORMS

Second and third normal forms [2, 3, 7] deal with the relationship between nonkey and key fields. Under second and third normal forms, *a nonkey field must provide a fact about the key, the whole key, and nothing but the*

Author's Present Address: William Kent, International Business Machines Corporation, General Products Division, Santa Teresa Laboratory, San Jose, CA

key. In addition, the record must satisfy first normal form.

We deal now only with "single-valued" facts. A single-valued fact could be a one-to-many relationship such as the department of an employee or a one-to-one relationship such as the spouse of an employee. Thus, the phrase "Y is a *fact* about X" signifies a one-to-one or one-to-many relationship between Y and X. In the general case, Y might consist of one or more fields and so might X. In the following example, QUANTITY is a fact about the combination of PART and WAREHOUSE.

3.1 Second Normal Form

Second normal form is violated when a nonkey field is a fact about a subset of a key. It is only relevant when the key is composite, i.e., consists of several fields. Consider the following inventory record.

PART	WAREHOUSE	QUANTITY	WAREHOUSE-ADDRESS
..... key			

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:

- 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
.... key		

WAREHOUSE	WAREHOUSE-ADDRESS
..... key	

When a data design is changed in this way, i.e., replacing unnormalized records with normalized records, the process is referred to as *normalization*. The term "normalization" is sometimes used relative to a particular normal form. Thus, a set of records may be normalized with respect to second normal form but not with respect to third.

The normalized design enhances the integrity of the data by minimizing redundancy and inconsistency, but at some possible performance cost for certain retrieval applications. Consider an application that wants the addresses of all warehouses stocking a certain part. In the unnormalized form, the application searches one record type. With the normalized design, the application has to search two record types and connect the appropriate pairs.

3.2 Third Normal Form

Third normal form is violated when a nonkey field is a fact about another nonkey field, as in

EMPLOYEE	DEPARTMENT	LOCATION
..... key		

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, e.g., 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	DEPARTMENT
.... key	

DEPARTMENT	LOCATION
..... key	

To summarize, a record is in second and third normal forms if every field is either part of the key or provides a (single-valued) fact about exactly the whole key and nothing else.

3.3 Functional Dependencies

In relational database theory, second and third normal forms are defined in terms of *functional dependencies*, which correspond approximately to our single-valued facts. A field Y is "functionally dependent" on a field (or fields) X if it is invalid to have two records with the same X value but different Y values. That is, a given X value must always occur with the same Y value. When X is a key, then all fields are by definition functionally dependent on X in a trivial way, since there cannot be two records having the same X value.

There is a slight technical difference between functional dependencies and single-valued facts as we have presented them. Functional dependencies only exist when the things involved have unique and singular identifiers (representations). For example, suppose a person's address is a single-valued fact, i.e., a person has only one address. If we do not provide unique identifiers for people, then there will not be a functional dependency in the data.

PERSON	ADDRESS
John Smith	123 Main St., New York
John Smith	321 Center St., San Francisco

Although each *person* has a unique address, a given *name* can appear with several different addresses. Hence, we do not have a functional dependency corresponding to our single-valued fact. Similarly, the address has to be spelled identically in each occurrence in order to have a functional dependency. In the following case, the same person appears to be living at two different addresses, again precluding a functional dependency.

PERSON	ADDRESS
John Smith	123 Main St., New York
John Smith	123 Main Street, NYC

We are not defending the use of nonunique or nonsingular representations. Such practices often lead to data maintenance problems of their own. We do wish to point out, however, that functional dependencies and the various normal forms are really only defined for situations in which there are unique and singular identifiers. Thus, the design guidelines as we present them are a bit stronger than those implied by the formal definitions of the normal forms.

For instance, we as designers know that in the following example there is a single-valued fact about a nonkey field, and hence the design is susceptible to all the update anomalies mentioned earlier.

EMPLOYEE	FATHER	FATHER'S-ADDRESS
Art Smith	John Smith	123 Main St., New York
Bob Smith	John Smith	123 Main Street, NYC
Cal Smith	John Smith	321 Center St., San Francisco

However, in formal terms, there is no functional dependency here between FATHER'S-ADDRESS and FATHER, and hence, no violation of third normal form.

4. FOURTH AND FIFTH NORMAL FORMS

Fourth [5] and fifth [6] normal forms deal with multivalued facts. A multivalued fact may correspond to a many-to-many relationship, as with employees and skills, or to a many-to-one relationship, as with the children of an employee (assuming only one parent is an employee). By "many-to-many" we mean that an employee may have several skills and/or a skill may belong to several employees. Note that we look at the many-to-one relationship between children and fathers as a single-valued fact about a child but a multivalued fact about a father.

In a sense, fourth and fifth normal forms are also about composite keys. These normal forms attempt to minimize the number of fields involved in a composite key, as suggested by the examples that follow.

4.1 Fourth Normal Form

Under fourth normal form, *a record type should not contain two or more independent multivalued facts about an entity.* In addition, the record must satisfy third normal form. The term "independent" will be discussed after considering an example.

Consider employees, skills, and languages, where an employee may have several skills and several languages. We have here two many-to-many relationships, one between employees and skills, and one between employees and languages. Under fourth normal form, these two relationships should not be represented in a single record such as

EMPLOYEE	SKILL	LANGUAGE
..... key		

Instead, they should be represented in the two records

EMPLOYEE	SKILL
..... key	

EMPLOYEE	LANGUAGE
..... key	

Note that other fields, not involving multivalued facts, are permitted to occur in the record, as in the case of the QUANTITY field in the earlier PART/WAREHOUSE example.

The main problem with violating fourth normal form is that it leads to uncertainties in the maintenance policies. Several policies are possible for maintaining two independent multivalued facts in one record.

(1) A disjoint format, in which a record contains either a skill or a language, but not both.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	
Smith	type	
Smith		French
Smith		German
Smith		Greek

This is not much different from maintaining two separate record types. We note in passing that such a format also leads to ambiguities regarding the meanings of blank fields. A blank SKILL could mean the person has no skill, that the field is not applicable to this employee, that the data is unknown, or, as in this case, that the data may be found in another record.

(2) A random mix, with three variations

(a) Minimal number of records with repetitions.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	German
Smith	type	Greek

(b) Minimal number of records, with null values.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	German
Smith		Greek

(c) Unrestricted.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	
Smith		German
Smith	type	Greek

(3) A "cross-product" form where, for each employee, there must be a record for every possible pairing of one of his skills with one of his languages.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	cook	German
Smith	cook	Greek
Smith	type	French
Smith	type	German
Smith	type	Greek

Other problems caused by violating fourth normal form are similar in spirit to those mentioned earlier for violations of second or third normal form. They take different variations depending on the chosen maintenance policy.

- If there are repetitions, then updates have to be done in multiple records, and the records could become inconsistent.
- Insertion of a new skill may involve looking for a record with a blank skill, inserting a new record with a possibly blank language, or inserting multiple records pairing the new skill with some or all of the languages.
- Deletion of a skill may involve blanking out the skill field in one or more records (perhaps with a check that this does not leave two records with the same language and a blank skill) or deleting one or more records, coupled with a check that the last mention of some language has not been deleted also.

Fourth normal form minimizes such update problems.

4.1.1 *Independence* We mentioned *independent* multivalued facts earlier, and we now illustrate what we mean by that term. The two many-to-many relationships, employee:skill and employee:language, are independent in that there is no direct connection between skills and languages. There is only an indirect connection because they belong to some common employee. That is, it does not matter which skill is paired with which language in a record; the pairing does not convey any information. That is precisely why all the maintenance policies mentioned earlier can be allowed.

In contrast, suppose that an employee can only exercise certain skills in certain languages. Perhaps Smith can cook French cuisine only, but can type French, German, and Greek. Then the pairing of skills and languages becomes meaningful, and there is no longer an ambiguity of maintenance policies. In the present case, only the following form is correct.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	French
Smith	type	German
Smith	type	Greek

Thus, the employee:skill and employee:language relationships are no longer independent. These records do not violate fourth normal form. When there is an interdependence among the relationships, it is acceptable to represent them in a single record.

4.1.2. *Multivalued Dependencies* For readers interested in pursuing the technical background of fourth normal form a bit further, we mention that fourth normal form is defined in terms of *multivalued dependencies* that correspond to our independent multivalued facts. Multivalued dependencies, in turn, are defined essentially as relationships that accept the "cross-product" maintenance policy mentioned above. For our example, every one of an employee's skills must appear paired with every one of his languages. It may or may not be obvious to the reader that this is equivalent to our notion of independence; since every possible pairing must be present, there is no "information" in the pairings. Such pairings convey information only if some of them can be absent, i.e., only if it is possible that some employee cannot perform some skill in some language. If all pairings are always present, then the relationships are really independent.

We should also point out that multivalued dependencies and fourth normal form also apply to relationships involving more than two fields. For example, suppose we extend the earlier example to include projects, in the following sense:

- An employee uses certain skills on certain projects.
- An employee uses certain languages on certain projects.

If there is no direct connection between the skills and languages that an employee uses on a project, then we could treat this as two independent many-to-many relationships of the form EP:S and EP:L, where EP represents a combination of an employee with a project. A record including employee, project, skill, and language would violate fourth normal form. Two records, containing fields E, P, S and E, P, L, respectively, would satisfy fourth normal form.

4.2. Fifth Normal Form

Fifth normal form deals with cases where information can be reconstructed from smaller pieces of information which can be maintained with less redundancy. Second, third, and fourth normal forms also serve this purpose, but fifth normal form generalizes to cases not covered by the others.

We will not attempt a comprehensive exposition of fifth normal form, but will illustrate the central concept with a commonly used example, namely, one involving agents, companies, and products. If agents represent companies, companies make products, and agents sell products, then we might want to keep a record of which agent sells which product for which company. This information could be kept in one record type with three fields:

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	GM	truck

This form is necessary in the general case. For example, although agent Smith sells cars made by Ford and trucks made by GM, he does not sell Ford trucks or GM cars. Thus, we need the combination of all three fields to know which combinations are valid and which are not.

But suppose that a certain rule is in effect: if an agent sells a certain product and he represents the company making that product, then he sells that product for that company.

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car

In this case, it turns out that we can reconstruct all the true facts from a normalized form consisting of three separate record types, each containing two fields.

AGENT	COMPANY
Smith	Ford
Smith	GM
Jones	Ford

AGENT	PRODUCT
Smith	car
Smith	truck
Jones	car

COMPANY	PRODUCT
Ford	car
Ford	truck
GM	car
GM	truck

These three record types are in fifth normal form, whereas the corresponding three-field record shown previously is not.

Roughly speaking, we may say that a record type is in fifth normal form when its information content cannot be reconstructed from several smaller record types, i.e., from record types each having fewer fields than the original record. The case where all the smaller records have the same key is excluded. If a record type can only be decomposed into smaller records which all have the same key, then the record type is considered to be in fifth normal form without decomposition. A record type in fifth normal form is also in fourth, third, second, and first normal forms.

Fifth normal form does not differ from fourth normal form unless there exists a symmetric constraint such as the rule about agents, companies, and products. In the absence of such a constraint, a record type in fourth normal form is always in fifth normal form.

One advantage of fifth normal form is that certain redundancies can be eliminated. In the normalized form, the fact that Smith sells cars is recorded only once; in the unnormalized form, it may be repeated many times.

It should be observed that although the normalized form involves more record types, there may be fewer total record occurrences. This is not apparent when there are only a few facts to record, as in the example shown above. The advantage is realized as more facts are recorded, since the size of the normalized files increases in an additive fashion, while the size of the unnormalized file increases in a multiplicative fashion. For example, if we add a new agent who sells x products for y companies, where each of these companies makes each of these products, we have to add $x + y$ new records to the normalized form, but $x \cdot y$ new records to the unnormalized form.

It should also be noted that all three record types are required in the normalized form in order to reconstruct the same information. From the first two record types shown above we learn that Jones represents Ford and that Ford makes trucks. But we cannot determine whether Jones sells Ford trucks until we look at the third record type to determine whether Jones sells trucks at all.

The following example illustrates a case in which the rule about agents, companies, and products is satisfied, and which clearly requires all three record types in the normalized form. Any two of the record types taken alone will imply something untrue.

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car
Jones	Ford	truck
Brown	Ford	car
Brown	GM	car
Brown	Toyota	car
Brown	Toyota	bus

AGENT	COMPANY
Smith	Ford
Smith	GM
Jones	Ford
Brown	Ford
Brown	GM
Brown	Toyota

Fifth Normal Form

COMPANY	PRODUCT
Ford	car
Ford	truck
GM	car
GM	truck
Toyota	car
Toyota	bus

Fifth Normal Form

AGENT	PRODUCT
Smith	car
Smith	truck
Jones	car
Jones	truck
Brown	car
Brown	bus

Fifth Normal Form

Observe that:

- Jones sells cars and GM makes cars, but Jones does not represent GM.
- Brown represents Ford and Ford makes trucks, but Brown does not sell trucks.
- Brown represents Ford and Brown sells buses, but Ford does not make buses.

Fourth and fifth normal forms both deal with combinations of multivalued facts. One difference is that the facts dealt with under fifth normal form are not independent, in the sense discussed earlier. Another difference is that, although fourth normal form can deal with more than two multivalued facts, it only recognizes them in pairwise groups. We can best explain this in terms of the normalization process implied by fourth normal form. If a record violates fourth normal form, the associated normalization process decomposes it into two records, each

containing fewer fields than the original record. Any of these violating fourth normal form is again decomposed into two records, and so on until the resulting records are all in fourth normal form. At each stage, the set of records after decomposition contains exactly the same information as the set of records before decomposition.

In the present example, no pairwise decomposition is possible. There is no combination of two smaller records which contains the same total information as the original record. All three of the smaller records are needed. Hence, an information-preserving pairwise decomposition is not possible, and the original record is not in violation of fourth normal form. Fifth normal form is needed in order to deal with the redundancies in this case.

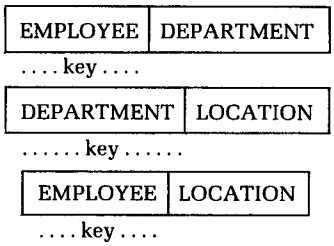
5. UNAVOIDABLE REDUNDANCIES

Normalization certainly does not remove all redundancies. Certain redundancies seem to be unavoidable, particularly when several multivalued facts are dependent rather than independent. In the example shown in Sec. 4.1.1, it seems unavoidable that we record the fact that "Smith can type" several times. Also, when the rule about agents, companies, and products is not in effect, it seems unavoidable that we record the fact that "Smith sells cars" several times.

6. INTERRECORD REDUNDANCY

The normal forms discussed here deal only with redundancies occurring within a single record type. Fifth normal form is considered to be the ultimate normal form with respect to such redundancies [6].

Other redundancies can occur across multiple record types. For the example concerning employees, departments, and locations, the following records are in third normal form in spite of the obvious redundancy.



In fact, two copies of the same record type would constitute the ultimate in this kind of undetected redundancy. Interrecord redundancy has been recognized for some time [1], and has recently been addressed in terms of normal forms and normalization [8].

7. CONCLUSION

While we have tried to present the normal forms in a simple and understandable way, we are by no means suggesting that the data design process is correspondingly simple. The design process involves many complexities that are quite beyond the scope of this paper. In the first place, an initial set of data elements and records has to be developed as candidates for normalization. Then the factors affecting normalization have to be assessed:

- Single-valued vs. multivalued facts.
- Dependency on the entire key.
- Independent vs. dependent facts.
- The presence of mutual constraints.
- The presence of nonunique or nonsingular representations.

Finally, the desirability of normalization has to be assessed in terms of its performance impact on retrieval applications.

Acknowledgments. I am very grateful to Ted Codd and Ron Fagin for reading earlier drafts and making valuable comments, and especially to Chris Date for helping to clarify some key points.

References

1. Codd, E.F. A relational model of data for large shared data banks. *Comm. ACM* 13 6, (June 1970) 377-387.
2. Codd, E.F. Normalized data base structure: A brief tutorial. *ACM SIGFIDET Workshop on Data Description, Access, and Control.* Nov. 11-12, 1971, San Diego, California. E.F. Codd and A.L. Dean (Eds.)
3. Codd, E.F. Further normalization of the data base relational model. R. Rustin (Ed.), *Data Base Systems* (Courant Computer Science Symposia 6). Prentice-Hall, Englewood Cliffs, NJ, 1972. Also IBM Research Report RJ909.
4. Date, C.J. *An Introduction to Database Systems.* 3rd Ed. Addison-Wesley, Reading, MA, 1981.
5. Fagin, R. Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems* 2 3, (Sept. 1977). Also IBM Research Report RJ1812.
6. Fagin, R. Normal forms and relational database operators. *ACM SIGMOD International Conference on Management of Data.* May 31-June 1, 1979, Boston, Mass. Also IBM Research Report RJ2471, Feb. 1979.
7. Kent, W. A primer of normal forms. *IBM Technical Report TR02.600,* Dec. 1973.
8. Ling, T.-W., Tompa, F.W., and Kameda, T. An improved third normal form for relational databases. *ACM Transactions on Database Systems* 6 2, (June 1981) 329-346.

CR Categories and Subjects: H.2.1 **[Database Management]**: Logical Design—*normal forms*
General Term: Design

Received 3/82; revised 8/82; accepted 9/82

Computing Practices readers are directed to this month's Forum letter, "On Animals Around the 'Wolf Fence'" by Ted Tenny, which refers to the Pracnique by E.J. Gauss in the November 1982 issue, and to the Technical Correspondence "On File Comparison Techniques" by Arthur I. Schwarz, referring to the Computing Practices article by Michael Pechura, which appeared in September 1982.