The normalization rules are designed to prevent update anomalies and data inconsistencies. With respect to performance tradeoffs, these guidelines are biased toward the assumption that all non-key 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 doesn't 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 non-key and key fields.

Under second and third normal forms, a non-key field must provide a fact about the key, us the whole key, and nothing but the key. In addition, the record must satisfy first normal form.

We deal now only with "single-valued" facts. The 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 non-key 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 |
====================-----------------------------
```

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 |  | WAREHOUSE | WAREHOUSE-ADDRESS |
======================-------  =============--------------------
```

When a data design is changed in this way, 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 non-key field is a fact about another non-key field, as in

```
-----------------------------------
| EMPLOYEE | DEPARTMENT | LOCATION |
============-=======================
```

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 | DEPARTMENT |  | DEPARTMENT | LOCATION |
============-=============  ==============-==========
```

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 can't 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 don't 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 non-unique or non-singular 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 non-key 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 multi-valued facts. The multi-valued 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 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 multi-valued 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 to follow.

4.1 Fourth Normal Form

Under fourth normal form, a record type should not contain two or more independent multi-valued facts about an entity. In addition, the record must satisfy third normal form.

The term "independent" will be discussed after considering an example.