3
The Searcher's Toolkit: Part 1

In my experience, there are a finite number of concepts, techniques, and strategies for searching databases that make all the difference between aimlessly groping around and efficiently and effectively retrieving useful material. If you spend your whole day searching, then you'll probably discover or develop many more, but for most researchers or reference librarians, the topics in this chapter and the next are most likely all that you will ever need. I've dubbed this set of concepts the *Searcher's Toolkit*. All of them have applications in searching commercial databases, and some can be used with Web search engines (and once you've grasped these concepts, you'll start to see what's missing in the Web search products and understand better why searching the Web is, well, the way it is: sometimes perfect, sometimes very frustrating).

The First Basic Tools

Let us plunge right in with the most fundamental concept of all.

Basic Tool No. 1: Boolean Logic

In fact, this concept is so fundamental that you've probably run into it before, possibly several times through grade school, high school, and college. But do you *really* know what Boolean logic is and how it works? Do you really understand how it will affect your searches? Bear with a discussion of it one more time—you may be surprised!

In the database context, Boolean logic (after a fellow named George Boole [1815–1864])[1] refers to the *logical* (rather than arithmetical) operations on sets. That is, rather than manipulating numbers using symbols for

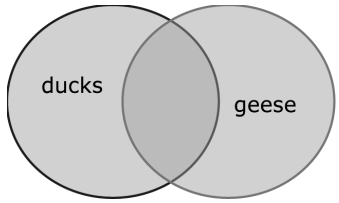

Figure 3.1. Boolean AND: Ducks AND Geese.

plus, minus, multiply, or divide, Boolean logic controls what happens to sets of things when acted on by *logical operators*. The Boolean operators used in database searching are:

AND

OR

AND NOT (frequently expressed simply as *NOT*; more on this later.)

Venn Diagrams

Arithmetical expressions (e.g., 2+2) result in some value (i.e., 4). Boolean expressions result in either a sub- or superset of the original sets; rather than producing a specific value, logical operators have an *effect* on the sets, producing a different output set. The effects of Boolean expressions are traditionally illustrated with drawings called *Venn diagrams*. Venn diagrams are always done with circles and shadings as in Figure 3.1, which keeps them nicely abstract. Feel free to start thinking of the terminology "sets," and these circle illustrations as "sets of database records," to make the concept more concrete. With just one operator in effect, Venn diagrams are quite simple to draw and to understand what the operator's effect is.

Boolean AND

In Figure 3.1, the circle on the left represents the set of all the database records that include the word *ducks*, a few of which also include the word *geese*. The circle on the right represents all records that include the word *geese*, a few of which also include the word *ducks*. A search for *ducks AND geese* retrieves only those records that mention *both* terms, represented by the smaller, overlapping section. It is easy to get confused here, because our regular use of the word *and* is additive, that is, it produces more (e.g., "two scoops, and sprinkles, and some whipped cream, please"), but a Boolean AND is very different: an AND operator will always, in practice, return a set that is *smaller* than either of the original sets. Theoretically, the largest set it could return would still be only of equal size to the smaller of the original

sets. For example, at one point in time, the ProQuest Research database contained 760 records that contained *Obama, Michelle* in a field called Person, and 28,790 records containing *Obesity* in a field called Subject. A search on

Obama, Michelle in the Person field,

AND

Obesity in the Subject field

produced 50 records.

Notice how even when the initial sets are quite large, the combined, "ANDed" set is considerably smaller. When one of the initial sets is much smaller than the other, as is the case here, the resulting set is affected even more.

AND says that *all* criteria must be met for an indicated action to happen (usually retrieval of a record). What the Boolean operators are really doing is evaluating true or false. The computer's thought process goes something like this:

If (ducks) is true (e.g., present in the record)

AND (at the same time, in the same record)

If (geese) is true

Retrieve the record.

So, to reiterate: the number of records meeting multiple conditions is, in practice, always smaller than the set of records meeting just one condition, and it is usually significantly smaller. The more conditions (criteria, terms) that you set, the smaller the number of records that will be retrieved: there will be fewer documents about ducks AND geese AND loons than there are about just ducks AND geese. If one of the initial sets is very small, ANDing it with some other term is likely to reduce the results to zero. But our next operator, OR, is here to help with that potential problem.

Boolean OR

In Figure 3.2, the circle on the left represents all the database records that include the word *banana*, and the circle on the right represents all the database records that include the word *orange*. In this case, we don't care

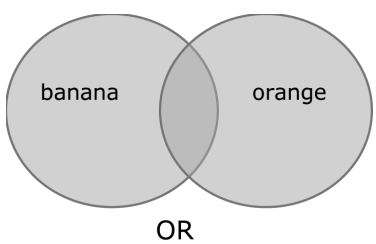

Figure 3.2. Boolean OR: Banana OR Orange.

what other words they contain, or how much they overlap. Thinking of this as a search, the OR retrieves *all* the bananas records, plus *all* the oranges records (including records mentioning both), for a total of a LOT of records! Again, our common parlance can make this confusing: usually we use the word *or* to mean "either one or the other"—"I'll have the banana *or* the orange." We wouldn't expect to be handed both fruits in response to that statement. But in Boolean logic, OR means "either the one, or the other, or both." Either of the criteria can be met for the computer to retrieve the record:

If (banana) is true (e.g., present in a record)

OR

If (orange) is true (present in a record)

Retrieve the record.

Think of it this way: in practice, Boolean OR is (practically) always more. By employing a judicious combination of Boolean AND and OR operators, you can "grow" small results sets in a controlled fashion, as we'll see later in this chapter.

Boolean NOT

Finally, Figure 3.3 represents the Boolean exclusion operator, which you'll see expressed as AND NOT and simply NOT with almost equal frequency. For our purposes, the terms are interchangeable, and this text will generally use simply NOT to make the operators as distinguishable as possible.[2] If this feels confusing, take comfort that at least the effects of this operator are more in line with our common usage of *not*: rivers NOT lakes retrieves records that include the term *rivers*, as long as they do NOT also contain the word lakes. Both criteria must be met, in the sense that the first term must be present, and the second term must *not* be present, for a record to be retrieved. The set of records retrieved can be thought of as just the lighter area of the *rivers* circle; anything from the darker *lakes* circle would not be in the results set. Note how the syntax is subtly different (which is further proof that even when a database vendor just uses NOT, what is really going on is AND NOT):

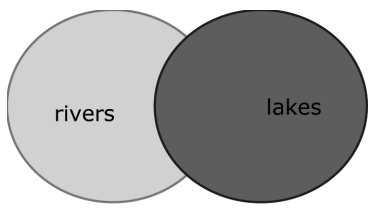

Figure 3.3. Boolean NOT: Rivers NOT Lakes.

If (rivers) is true (present in a record)

And

If (lakes) is NOT true (not present in the record)

Retrieve the record.

As you would expect, like AND, NOT almost always reduces the number of records retrieved. In general usage, you'll find that you don't often use the NOT operator in commercial databases: the possibility of missing useful records just because they happened to include the "NOTed" out term is too risky. If too many results are coming back, the better strategy is almost always to AND in another term, rather than to NOT out a term.

Order of Boolean Operations

A statement using just one Boolean operator—ducks OR geese—is straightforward. But just as you can write arithmetical expressions with several operators (2+2–3*9), you can write Boolean expressions with multiple operators. You will encounter plenty of searches that require more complexity than simply (word) AND (word). Again, just as in the arithmetical statements, the Boolean operators have very specific effects, and the order in which they are processed has a powerful effect on what ends up in the results set. When there is more than one operator in a search statement, they are generally evaluated in this order:

- NOT operations are performed first.
- Then AND operations are evaluated.
- Finally, OR operations are performed.

This is called the *order of operation*. This is the standard order for processing Boolean operators, but some systems simply evaluate statements left to right, the way you read. The results could be very different, depending on which order is used. Although it's important to be aware of the idea of order of operation, luckily you don't have to figure out what it is on each system you use. There is a simple way to take control and bend the order of operation to your will.

The Power of Parentheses

To control the order of operation, many systems allow you to group your ANDs, ORs, and NOTs with parentheses: ( ). Just as in arithmetic statements, the use of parentheses is helpful either to make the order of operation explicit, or to override it. What happens is that the expression in parentheses will be evaluated first, and then the order of operation (standard or left to right) will take over.[3] Throwing parentheses into the mix can dramatically change the way the system interprets the search.

For example, the statement

ducks NOT migration OR geese

produces the same result set as

(ducks NOT migration) OR geese

because in this case, putting the parentheses around the NOT statement (causing it to be executed first) is exactly the same as what happens in the standard order of operation (NOT first, OR last). The set of documents retrieved would be fairly large, and it would contain records for ducks (as long as those records didn't mention migration) and any records mentioning geese—even records that discuss geese and migration. OR really opens the door to let things back in, sometimes in surprising ways.

Represented as a Venn diagram, the statement (ducks NOT migration) OR geese looks like Figure 3.4.

The results of this search would include all the records except those represented by the visible dark area of the "migration" circle. Note that records about geese that mention migration *would* be included in the results; the NOT only affects the duck records. However, writing the search statement as

ducks NOT (migration OR geese)

will produce quite a different result set, much smaller and more focused. These would be records that mention ducks, and only those duck records that don't mention either migration or geese—the lightest area in Figure 3.5. The Venn diagram for this statement is shown in Figure 3.5.

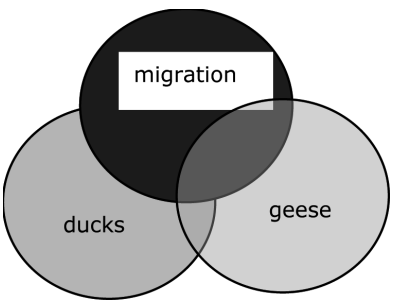

Figure 3.4. (Ducks NOT Migration) OR Geese.

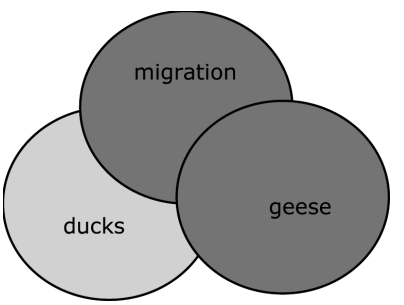

Figure 3.5. Ducks NOT (Migration OR Geese).

Quick Recap

This section introduced the first and most fundamental tool in the Searcher's Toolkit, Boolean logic. The Boolean logical operators are AND, OR, NOT (also expressed AND NOT). Adding terms to a search with AND will reduce the number of results, as will "NOTing" out a term. "ORing" more terms into a search will increase the number of results. Boolean expressions are often represented graphically with Venn diagrams. The standard order of operations for Boolean statements is to evaluate any NOT statements first, followed by AND statements, and finally by OR statements. Putting part of the statement in parentheses changes the order of operations, because the parenthesized section will be evaluated first.