Relational Databases Explained

In Chapter Two we briefly mentioned that relational databases can represent data in many-to-many relationships. Let’s look a bit more deeply into these relationships in databases.

Using Microsoft Excel for Complex Relationships

Imagine that you want to create a database of your personal contacts. In the analog days people would use address books for this task. A different system that allows for interfiling was the “Rolodex” (“google” the term, if you are unfamiliar with it). In the digital age, although database programs are available, they tend to be too complex for everyday tasks. Most people resort to Microsoft Excel, which was really designed as an electronic spreadsheet for accounting and bookkeeping tasks, but is easily adaptable to basic database functions. Let’s set up our project with fields as shown in Figure 1.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Last Name</td>
<td>First Name</td>
<td>Middle Initial</td>
<td>Address1</td>
<td>Address2</td>
<td>City</td>
<td>State</td>
<td>Zip</td>
<td>Phone (Land)</td>
<td>Phone (Cell)</td>
<td>Email</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 1. Address book setup in Microsoft Excel.*

In addition to standard name and address fields, we also have various contact fields. Most people have multiple ways to contact them, including several phone numbers and email contacts. After reflecting on this a bit more, we decide we need to add even more contact fields (Figure 2).

|   | A          | B       | C    | D    | E    | F    | G    | H    | I      | J         | K        | L          | M          | N          | O          | P          |
|---|------------|---------|-----|-----|-----|-----|-----|-----|-------|----------|----------|------------|------------|------------|------------|------------|------------|
| 1 | Last Name  | First Name | Middle Initial | Address1 | Address2 | City | State | Zip | Phone (Land) | Phone (Cell) | Email | Land1 | Phone (Cell) | Phone (Work) | Email1 | Land2 | Facebook | Twitter | Instagram |
| 2 |
| 3 |

*Figure 2. Address book with many contact fields.*

Now things are getting a lot more complicated. We have the option of three telephone numbers, two email addresses, and three flavors of social media. But what if a person only has one phone number? The other two fields simply remain empty. Or worse, what if they have four numbers? We have to either leave out one of the numbers, add it somehow to a miscellaneous notes field (that we haven’t yet created), or redesign the entire database to accommodate instances like this.

One workaround for this situation is a relational database. Examples of relational databases include Microsoft Access and FileMaker Pro on the basic client computer model side of things and Oracle and Panorama on the more sophisticated side.

In Figure 2 above, all of the cells in a given row (or record) exist for each record, whether they are needed or not. One of the contacts may have a land phone and a cell phone, but not a work phone or social media connections. For another contact, perhaps only a work phone number is known, and the other contact-related fields are empty. Nevertheless, the empty data cells persist in the database.
Using Microsoft Access for Complex Relationships

Let’s take a look at how we might design a database with a relational database. Microsoft Access is very commonly found on Windows computers, although it is not bundled in home editions of Microsoft Office. Access is not as easy to learn as Excel, but it is much easier than the higher-end relational databases. Figure 3 shows an example of how we might set up a database for contacts, but have the contacts in a separate table.

![Figure 3. Address book setup in Microsoft Access with relational table.](image)

The first thing to notice in Figure 3 is that there are two tables. The field labelled ID in the Names Table has a corresponding field in the Contacts Table. A line linking the two tables shows a “1” near the Names Table and an infinity symbol near the Contacts Table, denoting the one-to-many relationship. For each name record there can be one or more contact records. It may be that someone has only a work phone, and thus only a single record is required in the Contacts Table. For another name, however, all the contacts types may be needed. In fact, there may be three personal email addresses and two work email addresses, and the relational design is able to handle all of that.

To translate this to a realm very understandable to information professionals, let’s look at an example used to create a bibliography. Figure 4 shows a record from a project I worked on in the early 2000s, a bibliography for the United Nations Centre for Regional Development, based in Nagoya, Japan. The top part of the page is the central part of the database, the main table. This contains elements like the title, whether it is a book chapter, a journal article, etc., the pagination, and miscellaneous notes about figures, notes, and tables contained in the item being indexed. Below that you will notice a subform containing names of authors. Notice that it is possible to have many authors, since multiple authorship is common in academic publishing. You will also notice a subject subform with multiple subjects applied to this item. Finally, a relation subform
shows the relationship of this item to the whole item. This is a book chapter, and it is part of a book.

**Figure 4.** Microsoft Access form showing Title Table and related subforms in UNCRD database.

Now let’s see how this database has been designed within Microsoft Access (Figure 5).

**Figure 5.** Structure of relationships for UNCRD database.
The main table in this relational structure is the Title Table. Off to the right you will notice the Author Link Table and the Author Table. When designing this project the desired outcome was to have a many-to-many relationship between items in the Title Table and authors in the Author Table. We need this because one item (journal article, book chapter, etc.) can potentially be written by many authors, on the one hand, and a single author can (and often is) responsible for more than one work. It is not possible to have a direct many-to-many relationship between two tables. But by inserting an intervening table, the Author Link Table, we can establish a one-to-many relationship between the Title Table and the Author Link Table, and another one-to-many relationship between the Author Table and the Author Link Table. Another advantage to having the intervening Author Link Table is that we can assign an affiliation field—the affiliation of a particular author at the time they authored the item being indexed. It is often the case that authors change affiliations. So associating an affiliation with a field in the author table would not work.

By now you should be seeing some of the advantages of relational structures within a database. But let’s now look at the left-hand side, the subjects. For this project I decided to use the *Macrothesaurus for Information Processing in the Field of Economic and Social Development* (published by the Organisation for Economic Co-Operation and Development (OECD)) as the controlled vocabulary for the project. These entries are represented in the Thesaurus Table (Figure 5). Since there is a many-to-many relationship between items in the Title Table and those in the Thesaurus Table, an intermediate table needed to be employed (Subject Link Table).

By now you can see that relational databases can be usefully complex, but a down side is that they are difficult to migrate to other platforms. This is likely why we generally do not see them in the online databases available from vendors. Instead, we still rely on old-fashioned machine-readable cataloging (MARC).