



## Databases and I-O

**R. Jason Weiss**  
**Development Dimensions International**

**Jeffrey Worst**  
**Private I-O and IT Consultant**



This column focuses on budding technologies and their implications for I-O psychology. A review of the technologies discussed so far and some that are on the board for future columns reveals an interesting commonality: Underlying each technology is at least one database playing a key role. Closer to home, databases are at the heart of much of the software that I-O practitioners design and use, such as Human Resource Information Systems (HRIS), applicant screening applications, 360° software, and many others. Still closer to home, most versions of Microsoft Office install an advanced database application, Microsoft Access, by default.

As we thought about it, we found some good arguments for devoting space in **Leading Edge** to a discussion of databases. As suggested above, databases underlie many of the emerging technologies highlighted in this column; a review of database concepts should help those with less technical backgrounds better evaluate the promises of these technologies. Second, we feel that relational database management systems (RDBMS) have a great deal to offer I-O psychologists of all stripes but have been overlooked in favor of more familiar tools that can emulate some simple database functionality. We hope that a look at some of the advantages of RDBMSs might save someone some difficulty down the road when faced with a data management task that would, say, stretch a spreadsheet beyond its limits.

As we will see, databases range in complexity, capability, and in the types of software used to interact with, and manage them. We start our discussion with simpler databases, known as *flat* or *flat-file* databases. As we will note, many of us already create and use flat-file databases due to their simplicity and flexibility. We then discuss relational databases, which have significant advantages over flat-file databases in generating flexible, powerful solutions. We finish with a discussion of resources for those interested in learning more or diving into the world of databases.

## Keeping It Simple: Flat-File Databases

The first order of business is to define the term *database*. Many definitions exist, but for our purposes we will define a database as an organized collection of information related to a given topic. The information in a database is organized into files or records in a given file. A typical simple database is the *flat-file* database. A flat-file database is a single table of information. Each column, or field, in the table represents a variable, or category of information to be stored. Each row, or *record*, in the table describes a unique person, object, or concept that is described by the variables. The term “flat” is used to communicate the idea that the database is two-dimensional—all data are contained in the row-column structure of a single table.

Figure 1 is a very simple example of a very small flat-file database. This should look familiar to most readers who use spreadsheets or statistical analysis packages. Let’s assume that this is the very beginning of a simple applicant-tracking database for a rural company that primarily hires high school students from the surrounding area. Note how often the data related to school information is repeated (i.e., school name, school district, and address). Since most of this company’s applicants come from the surrounding area high schools, a significant amount of the data in this database will be the same.

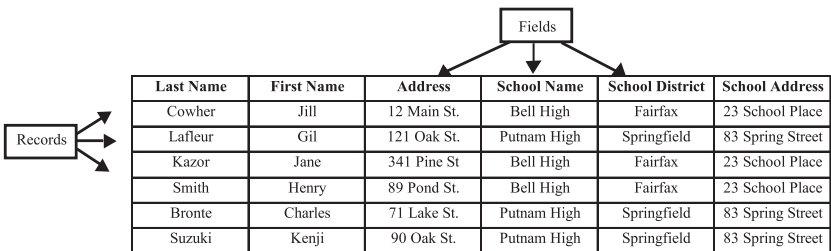


Figure 1. Sample flat-file database.

### *The Good News*

There is a lot to recommend flat-file databases. They are easy to conceptualize and do not require specialized background knowledge. They can be created and managed in almost any modern word processor or spreadsheet. Simple text editors such as the Windows Notepad may even be suitable for some flat-file databases. These are very powerful advantages—people often create and use flat-file databases without necessarily knowing anything about databases in the abstract. Consider the following representative examples that might be found on a given I-O psychologist’s computer:

- A spreadsheet file used to store and perform basic analyses on assessment data, where each record represents a different candidate, and each field a different competency;

- A collection of names and data stored in a word-processor table, used to generate reports as part of a mail-merge;
- A text file output by a “scan-sheet” reader, with names and data from a run of research participants, with responses to each question occupying a given column across all participants.

It seems as though flat-file databases are everywhere when you look for them. They are good, simple ways of working with data.

### *The Bad News*

The simplicity of using well-known productivity applications like word processors and spreadsheets is attractive and perfectly suitable for a wide range of applications. However, these solutions can also be extremely confining. For example, each software application makes assumptions about the data that it will accommodate—a spreadsheet cell can contain text and numerical data, but not binary files, so including graphic or sound files in the database is not possible. A second example of the confining nature of these types of databases is that management of each database is completely in the hands of the person who uses or maintains the file. Database management comprises a variety of functions including adding, updating, and deleting records, and controlling others’ access to the data, among others. This, too, is not a problem, provided that the database in question is small and easily managed. However, with frequent changes to records or a need to share different data with different people, it can grow very tedious.

Bigger problems arise when requirements increase and a simple solution gets stretched in directions it cannot easily support. Using the spreadsheet-based assessment data spreadsheet described above, let’s say that assessments are being run concurrently in many different geographical locations, and the data need to be maintained in a central location. Since a given spreadsheet file can only be edited by one user at a time, updating the file would require either that representatives from each location take turns entering their data or that everyone forward their data to a single person who compiles it all. Either solution is messy and fraught with the danger of data loss or corruption, though potentially manageable. With a little more complexity, the whole situation becomes completely untenable. For example, what if there is a second spreadsheet with resume data...and a third with test score data? That’s when it’s time to look for a better solution.

Software issues aside, the flat-file database format is not ideal. Consider, for example, the issue of redundancy. The sample database above contains a significant amount of redundant information as data related to high schools are repeated across applicants. Naturally, this amounts to a great deal of wasted space across a large database, bloating the size of the database file. Worse, errors may slip into the odd entry such that “Bell High” occasionally comes out as “Belle High” or “Bel High.” If the recruiter is interested in

looking at summary analyses by high school, these types of anomalies will prove to be a source of considerable frustration. Many more issues have been documented but would be beyond the scope of this article. Readers interested in a more complete discussion are directed to Roman (1999).

As we prepare to turn our attention to more sophisticated relational databases, it is necessary that we emphasize that the use of flat-file databases with spreadsheet or word processing software is not a straw man solution. There are many situations in which such approaches make good sense, such as when the database is small and manageable, or if its use will be relatively temporary. It is when database management requirements grow more complex that these tools are found wanting. In such cases, significant power can be harnessed with dedicated database-management software. With this in mind, let's take a look at RDBMSs and the functionality they offer.

### Feel the Power: Relational Database Management Systems

Our discussion of RDBMSs is divided into two parts. We begin with a review of the basic concepts underlying relational database design and the advantages offered by the relational database approach. Next, we describe basic functionality included in common off-the-shelf RDBMS systems.

#### *The Relational Database Design*

Similar to a flat-file database, a relational database uses tables as containers for data. However, a relational database includes multiple tables of information, rather than the single table of a flat-file database. Typically, each table contains a set of fields focused on a specific subject. Let's consider this by taking our flat-file database example above and turning it into two tables. In Figure 2, the first table is the "applicant" table, showing information about the applicants. The second table is the "school" table.

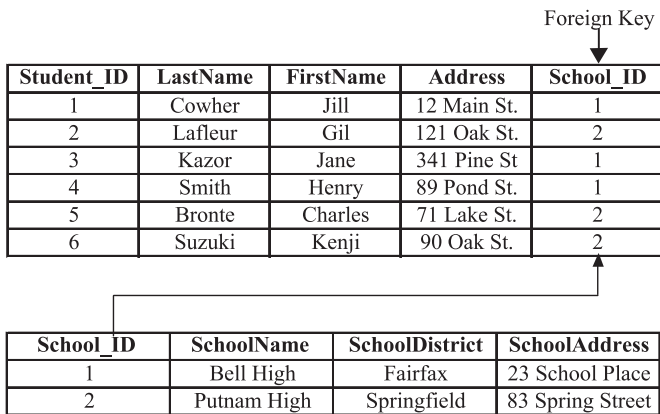


Figure 2. Simple Relational Database.

Note that we have added some new fields to help work with the tables. We added a Student ID field to the student table. This ID becomes what is known as the primary key for that table since it uniquely identifies each record in the table. For the school table, the School ID field is the primary key. A School ID field has also been added to the student table so we can determine which school a student has attended. It is called a foreign key in this context because it refers to a primary key in a different table.

### ***Advantages of Relational Database Design***

Relational databases achieve a number of powerful advantages by using additional tables to hold information that would otherwise be repeated across records. Let's just explore a few of these advantages to illustrate the utility of the relational database. First, having a single record for each high school means that all references to that high school refer to the same information. As a result, we would avoid the problem described above where data-entry errors result in minor variations in the high school name and frustrate efforts to generate meaningful summaries across high schools. A related advantage is that if we have to make changes to the high school information—say Bell High changes school districts—we only have to make the change in one place and can rest assured that the change has been implemented universally. A final advantage we will mention—though there are many more—is that the universe of information in the database is not limited to what is covered in the individual records. For example, with a relational database, we can represent all high schools in the region, rather than just those which existing applicants have attended. As a result, we can consider not only which schools are producing applicants but which schools aren't.

Relational database design is only one side of RDBMS software. The other side is the functionality that supports working with the data. In the following section, we describe typical functions universal to RDBMS packages along with some advanced functionality unique to high-end packages.

### ***RDBMS Functionality***

#### ***Queries***

If you've had a conversation with someone who works with databases, it's likely that you have heard the word "query" at some point. One way to think about a query is as a business question that can be answered through an analysis of the data in the RDBMS. You ask the database for a subset of its data that relate to the question. Queries are composed using a standard language called Structured Query Language (SQL). The acronym "SQL" is pronounced as if you were saying the word "sequel." Currently, all major RDBMSs (e.g., Oracle, MS SQLServer, MS Access, etc.) use SQL as the language for writing queries and processing data. It is actually an easy language to learn and is very intuitive.

For example, suppose we are asked, “Who did we interview from Putnam High?” To find the answer, we would write a query that joins together the two tables in our database. Our query might look like the following SQL statement:

```
SELECT applicant.lastname, applicant.firstname,
school.schoolname
FROM applicant INNER JOIN school
ON applicant.school_id = school.school_id
WHERE ((school.schoolname)="Putnam High");
```

In plain English, it says that we are interested in seeing the applicant’s last name, first name, and school name (line 1). These are to be found in the applicant and school tables (line 2). We will join these two tables to get the appropriate information by matching records according to the school ID code found in the records of both tables (line 3). Remember, each applicant in the applicant table has a School\_ID code representing the high school he or she attended, and this points back to the record associated with the matching School\_ID in the school table. Finally, we are interested only in the records where the name of the school is “Putnam High” (line 4). The results of this query would return in a form similar to Table 1.

Table 1.

*Sample Query Results*

<b>First Name</b>	<b>Last Name</b>	<b>School Name</b>
Lafleur	Gil	Putnam High
Bronte	Charles	Putnam High
Suzuki	Kenji	Putnam High

Of course, we don’t really need to show the school name, since we wrote the query to select only interviewees from Putnam High. We just included it in this example as a way of illustrating the query joining the two tables.

You don’t have to be a programmer to write a query. With a minimal amount of instruction, one can begin writing his or her own queries very easily. For those who would like some assistance in the process, RDBMSs such as MS Access have a “wizard” that guides the user in query construction.

Once a query is written, it can be saved and run again—you can use the query to answer that question whenever necessary. We have found that once people learn the power of queries they start generating all sorts of questions

that otherwise might have seemed impossible to consider when the data were confined to a filing cabinet. This is often called “mining the data” or extracting as much value as possible from the database.

### ***Reports***

Though queries are an exciting idea on their own, the results of a query are typically not much to look at. To help make them presentable, RDBMSs also have the ability to create formatted reports that have the query results embedded within them. Hence, once you have set up your report and linked the applicable queries to it, you can run the report minutes before a meeting and present “hot off the press” results in a professionally formatted document. You can develop as many reports and queries as necessary to meet your business information reporting needs.

### ***Database Applications***

Finally, a database can serve as what is sometimes called the “back end” of a software application. For example, when you are browsing through Amazon.com, what you see on your computer screen (i.e., the various Web pages) forms the front end of the Web site. All of the data on books, customers, and orders is contained in a database at the back end. For example, you use the front end of Amazon’s Web site to enter the name of a book you might like to buy and press a button to execute a search for the book. The front end takes this request, uses it to query the back-end database where the book titles are stored and then returns the result back to you via the front end (i.e., a new Web page). Purchasing a book and entering delivery information might take a number of such front-end/back-end interactions. A similar front-end/back-end approach is taken by nearly all database applications.

## **Moving On**

We hope that our discussion of databases has accomplished two goals. The first was to communicate how databases have a lot to offer as a way of storing and working with data—as it is, we have barely scratched the surface. The second goal was to generate excitement and encourage those readers who have not done so to consider creating databases to address their own data management needs. We would like to suggest the following resources for those readers interested in creating their own databases or learning more about database technology:

### ***Database Software***

#### ***Microsoft Access***

As noted above, most versions of Microsoft Office include Microsoft Access, which is a reasonably powerful desktop RDBMS. If you installed Office with all of the defaults, look for an icon featuring a maroon key. That’s

Access! The online help offers good information on getting started. Additionally, we have other “getting started” resources below.

### ***Freeware Databases***

MySQL and PostgreSQL are available on the Web as free downloads. You can find MySQL at <http://www.mysql.com>, and PostgreSQL at <http://www.postgresql.org>. Both are available for Windows and Linux, as well as a number of other operating systems.

## ***Resources on the Web***

### ***User Communities***

*Google Groups.* There are several good sources of user communities on the Web. One is the Usenet newsgroups accessible through Google Groups (formerly [deja.com](http://deja.com)), located at <http://groups.google.com>. The search terms “mailing.database” yielded a sizeable number of discussion boards.

*Microsoft Communities.* Microsoft has a set of boards discussing its database offerings (Access and SQL Server). These can be found by starting at <http://communities2.microsoft.com/home/default.aspx>.

*Yahoo Groups.* Yahoo has a number of groups associated with a variety of database topics. To locate them, start at <http://groups.yahoo.com> and search for “SQL,” “Microsoft Access,” or other database-related terms.

### ***Web Sites***

A vast number of Web sites are devoted to database-related topics. Here are a few good starting points:

- <http://www.sqlcourse.com> A free online tutorial on writing SQL queries. Best of all, it includes a “live” SQL interpreter, so you can practice what you learn. Very impressive.
- <http://www.mvps.org/access/> The home of the Microsoft Access MVPs, many of whom answer user questions on the Microsoft discussion boards described above. Lots to learn on this site, spanning everything from how-to articles to a compendium of bugs in the software.
- <http://www.microsoft.com/office/using/column06.asp> A nice frequently-asked questions guide to databases and Access.

## **Reference**

Roman, S. (1999). *Access database design and programming*. Sebastopol, CA: O’Reilly.