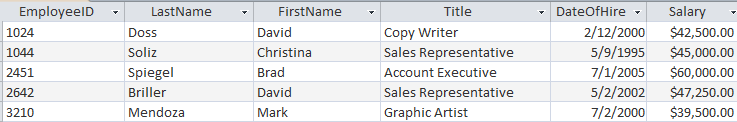
**Introduction to Database tables**

You use databases every day. When you go to the grocery store and the clerk scans one of your items at the register, that transaction is registered in a database. The grocery store’s database will record that they have one less of those items in inventory. If you booked a flight on an airplane, selected a seat on that plane, looked up a friend’s contact information on your phone, checked in at the doctor’s office, made an online purchase at Amazon… you used a database. Every day, even though you don’t hear the word ‘database’, you are working with databases.

A **database file**, by definition, is an organized collection of data, typically stored in electronic format. It allows you to input, manage, organize and retrieve data quickly. Traditional databases are made up of database tables. A **database table** is a collection of rows and columns that are used to organize information about a single topic. Each row within the table corresponds to a single **record** and contains several pieces of information pertaining to that record. Each piece of information in a record is referred to as a **field**. The below table contains employee records. Each record (row) contains information about one employee. Each field (column) contains a specific piece of information about that employee. In this table there are 5 employee records and each record contains the following fields: EmployeeID, LastName, FirstName, Title, DateOfHire and Salary.



**Database Management Software** (like Microsoft Access) is software that allows you to use a computer to create a database; add, change, delete data in the database; ask and answer questions concerning the data in the database; and create forms and reports using the data in the database.

**Database Models**

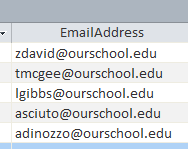
The three main types of database models are **flat**, **relational** and **hierarchical** models. In this course we will deal primarily with the first two; flat and relational. This document focusses on the tables that are used within each model, the components of each table and the advantages and disadvantages of each model. The process of designing, creating, populating and querying the database will be covered in a future lecture. For now, let’s focus on the concept.

**Flat Database Model**

The flat database model puts all database information in *one* large table, or list; much like a spreadsheet.

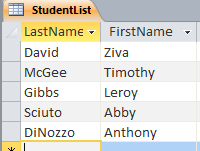
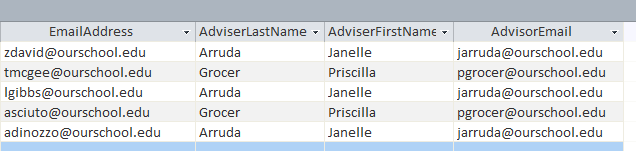
**Problems with Flat Databases**

The below image (*figure* 1) shows a simple table of student data stored in a flat database. The student table is a very simple list, and for such a list a flat database works quite well. Even if the list is long, you can sort it alphabetically by last name, first name, zip code, state, you can also search the table for specific student information. You can change the data for an existing student, add data for a new student, or delete student data.



Figure

Suppose, however, we change the student table by adding adviser data, as shown in the below image (*figure* *2*).

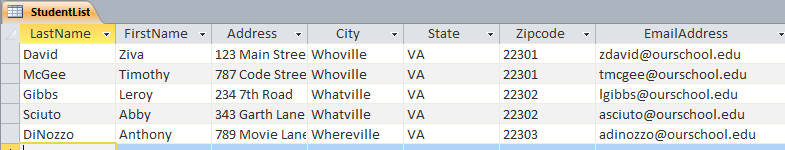


Figure

You can still sort, search, add, modify and delete from the table but since each adviser advises more than one student, the same adviser’s name and email address appears more than once in the table. This is a very small table - what if there were 500 students and 10 advisers? The same data (adviser’s name and email address) would be entered in numerous locations in the table. Entering the same data more than once takes time and increases the probability that typographical errors will occur. Erroneous entries will generate inconsistent data. For example, if the email address for one of the adviser’s is entered correctly for half of her advisees and incorrectly for the other half, the table would contain inconsistent data.

Now, what exactly happened in these two examples? We had a flat database with three fields, added two fields to it and thereby created a table which contains redundant data and is prone for corruption. The problem is not the number of columns; the problem is the content of those columns. Consider a different table that has 6 columns in *Figure 3*:

There’s no redundancy in this table. An essential difference does exist between the student list in *Figure 2* and the student list in *Figure 3*. Can you determine the difference? The essential difference is that the student list in *Figure 3* is about a *single thing*: All the data in that list is about the *student*. In contrast, the student list in *Figure 2* is about *two things:* Some of the data is about the *student* and some of the data is about the *adviser*. In general, if a table (or flat database) has data about two or more different things, data redundancy will likely occur and the table will be prone for corruption.



Figure

**Flat Database Disadvantages**

### Advantages

* Fast data retrieval for small files
* Easier for a human to read
* Easy to understand.
* Easy to implement.
* Less hardware and software requirements.
* Less skill sets are required to create and handle flat database systems.

### Disadvantages

* Data redundancy
* Extra time it takes to enter (re-enter) redundant data
* Data redundancy can also lead to data inconsistency
* When redundant data exists, disk space is wasted
* Access mode is sequential therefore it gets slower as the database grows
* Not conducive to complicated search queries
* Impossible to assign different levels of access to different users. Even if the flat-file database is stored in a DBMS (database management system) it is not possible to implement the table-level access that relational databases allow.
* A final, and rather complex disadvantage of flat-file databases is that the user interface is dependent on the file structure. That means that if you want to create a different view of the data, just some of the fields say, then you have to create a new file. This is known as program-data dependence.

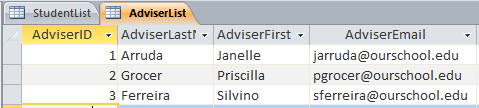
**Relational Database Model**

A number of different techniques were developed to solve the problems of using flat database files. A methodology called the **relational model** emerged as the leading solution, and today almost every commercial database is based on the relational model. The relational database model is much more complicated than the flat model will be examined in more detail in a separate lecture. Here, however, we will introduce the basic concept of the relational model by showing how it solves the redundancy problems experienced with flat databases.

A **relational database** contains a collection of separate tables. A **table** holds data about one and only one *theme.* If a table has two or more themes, we break it up into two or more tables.

**A Relational Design for the Student List**

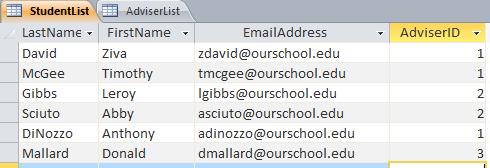
Refer back to *Figure 2* in this document. The student list in that Figure has two themes: *students and advisers.* To solve the redundancy problems we would need to place this data in a relational database model. This can be done by creating two tables as depicted in Figure 4 below. In addition to moving the data into two tables, a unique ID number would be assigned to each adviser. This will allow you to use the adviser’s ID number in the student list in place of the adviser’s name and email address. Now, if a change is made to an advisor’s name or email address, it need only be done in one place; in the AdviserList table.



Adviser Data was deleted from the original Student List and added to the newly created Adviser List.

An AdviserID field was included in the Adviser List. This will be used to assign a unique ID number to each adviser.

The Adviser’s ID number has been added to the Student List.



Figure

**Relational Database Disadvantages**

### Advantages

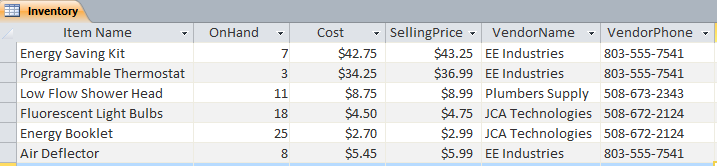
* Avoids data duplication
* Avoids Inconsistent records
* More efficient use of disk space
* Complex queries can be carried out.
* Better Security - you are able to assign different levels of access to data. This is possible by splitting data into separate tables and assigning different levels of authorization to each table.
* The user interface is not dependent on the file structure. That means that if you want to create a different view of the data, just some of the fields say, you can.

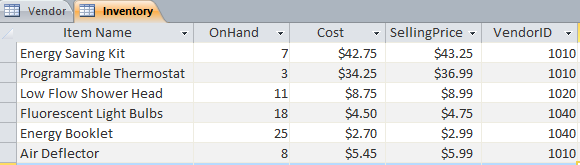
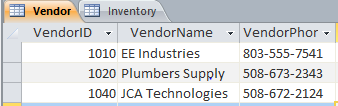
### Disadvantages

* The main problem when using a relational database is the complexity that arises when it is first created. It is absolutely vital that the defined relationships between the tables are correct and that each set of information is linked to its pair. Although less information has to be entered in total than with other databases, making sure every point is set up correctly is a slow process. Furthermore, the relationships can become extremely intricate when a relational database contains more than just two tables.
* Performance: A major constraint and therefore disadvantage in the use of relational database system is machine performance. If the number of tables between which relationships to be established are large and the tables themselves effect the performance in responding to queries.
* Physical Storage Consumption: With an interactive system, for example an operation like join (combining rows from two or more tables) would depend upon the physical storage.

**Inventory Example**

Let’s look at another example of examining a flat database file for redundancy. Consider the inventory table displayed in *Figure 5*. How many themes exist in this table? Does the table contain redundant data? The data in this table contains information about inventory items; the item name, how many are on hand, the items cost and the selling price. It also contains information pertaining to the vendor from whom the item was purchased. The vendor’s name and phone number appear repeatedly in the file causing redundancy. Since there are two themes in the table, it would be safe to say that we can split the data into two tables; one will contain the information that pertains to the item and the other will contain the information pertaining to the vendor. The two relational tables created to resolve the redundancy of this table are displayed in *Figure* 6

Figure 



Vendor Data was deleted from the original Inventory table and added to the newly created Vendor table.

An VendorID field was included in the Vendor Table. This will be used to assign a unique ID number to each Vendor.

The Vendor’s ID number has been added to the Inventory table in order to associate the item with the appropriate vendor.

Figure