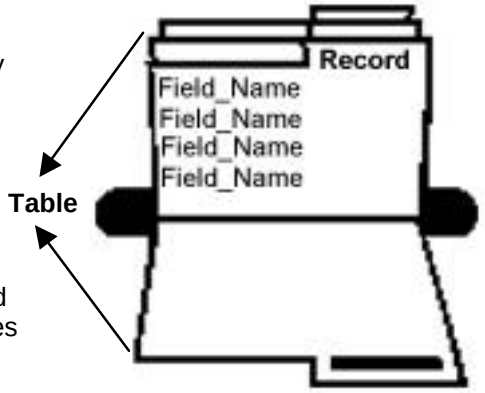
Database Design

WHAT IS A DATABASE?



A database is a store of information. The data is stored in **tables** and categorized by **fields**. Each group of information is a **record**.

Rolodex Example:

Think of a Rolodex as a table of information. Each card in the Rolodex is a record. The categories within each card are fields (name, address, phone number, etc.).

A Rolodex has tabs for each letter of the alphabet so that you can find the person you are looking for quickly and easily. A database can be searched or queried to find the data required. Because data is generally retrieved via a query, the data itself does not need to be entered in any particular order.

The type of database described above is called a **flat file** database. This is because it is like a single sheet of paper with all the information on it.

Microsoft Access is a **relational database**. This type of database has a much more complex design which, in turn, offers much more functionality and power. The downside is that an effective relational database needs to be designed properly.

A FLAT FILE VRS RELATIONAL DATABASE

Scenario:

You have decided to place all of your Rolodex information into a database in order to keep more detailed information on each of your contacts. You have noticed in your many successful years, that remembering the names and birth dates of your contacts' children produces a very positive image for yourself! So, your primary objective is to make sure the children's details are accurate and easily accessible. You decide you want to store the following information:

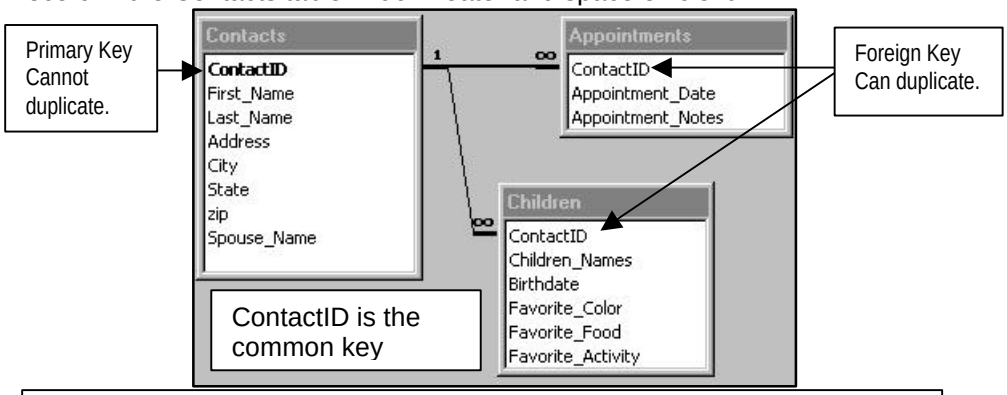
- Contacts – first name, last name, address, city, state, zip, spouse name
- Children – child's name, child's birth date, child's favorite color, child's favorite food, child's favorite activity
- Appointments – appointment date, appointment notes

Flat File

In a flat file database you would need to have a number of tables (3 to be exact), each of which holds one set of information. Unfortunately each table would store a large amount of duplicate information, each child would require the reentry of the contact's name. At some point a typing error is likely to occur, for example a contact's name is incorrectly typed. Now, if you try to look at all the children for a contact, you will miss some because of the typing error. Also, the huge amount of duplicate information takes up disk space, computer memory and will slow down the system.

Relational File

Now consider the situation where you have the same number of tables, but now assume you give each contact a reference number and against each child's name you enter that reference number. You can now **relate** a contact to a child and if a contact's details change (i.e. last name because of marriage or divorce) then you only have to change one record in the Contacts table. Much neater and space efficient!



A relational database will provide a link between these tables so that you can create a query, which will show you the children for each contact. A relational database saves typing, reduces the potential for errors, reduces the size of a database and makes for much more flexible data management.

In the relational database above, a contact is related to many children and many appointments using a common key. The common key that cannot be repeated is the Primary key in the Contacts table.

DATABASE DESIGN

Database design has nothing to do with using computers. It has everything to do with research and planning. The design process should be completely independent of software choices. When designing a relational database a number of terms and methods apply. The following pages detail each of these. However, before getting down to the specifics it is important to note one thing - there are not many better ways of designing a database than with a pen and paper. A good idea is to consider what outputs you need from the database (reports, data files, etc.) and then decide what these outputs need to be made up of (customer information, sales figures, etc.).

TERMINOLOGY

Normalization

Normalization is a process designed to achieve three ends ... eliminate redundant information, increase data integrity and make systems more efficient. There are four basic rules of normalization:

1. Each table must have a unique key.
   Although not mandatory, a primary key is a field that uniquely identifies each record in a table. This helps ensure that the system does not hold duplicate data.
2. Eliminate repeating groups.
   Split your data where you have repeating groups. For the Rolodex example, we split the data into tables of unique children and unique appointments.
3. Examine multi-field key relationships.
   When you have a Primary key based on two or more fields you may be able to split that table.
4. Examine any remaining groups of information.
   If descriptive information does not directly relate to the key in that table then move the descriptive data to a new table.

ID as a Primary Key

Primary keys are used to uniquely identify a record in a table. It is quite usual for a table's primary key to be some sort of ID such as a customer ID (or Social Security Number). The ID itself is normally a number and in Access (and many other systems) this number can automatically be assigned by the system (data type called Auto Number). The benefits of using an automatically created number are that the user does not even have to consider it, it takes up very little storage space and it is faster to add new data with one than with a text field (tables are primarily sorted by the primary key so adding a lot of data, by importing for example, means that the data is added to the end of the table rather than the system having to slot it into the middle of the table).

Relationships

Relationships 'connect' tables. In other words they link the data in one table to the data in another. Relationships are established using a common field that is present in both the tables to be related. Data can be linked via three types of relationship - one-to-one, one-to-many and many-to-many.

In a one-to-one relationship each record in one table has at most one related record in another table. This type of relationship is rare.

A one-to-many relationship is by far the most common. Here one record in one table can be related to many records in another table. In the rolodex example each contact can have many appointments and many children.

A many-to-many relationship means that for each record in one table there can be many records in another table and for each record in the second table there can be many in the first. Many-to-many relationships can not be directly represented in relational database programs and have to be built by using two or more one-to-many relationships.

Integrity Rules

There are two general integrity rules and these apply to all databases.

1. The **entity** integrity rule says that primary keys cannot contain null (empty) values.
2. The **referential** integrity rule says that the database must not contain any unmatched foreign key values. What this is saying is that a record cannot be added to a table with a foreign key unless the referenced value exists in the primary table.

Design Steps

Complete these steps using a piece of paper and a pencil or a whiteboard - anything erasable. Experiment with sample data and models of your forms and reports. Make sure the database design stores the data in the manner you need, retrieves the data correctly, and gives you the output (printed document) that you require. It is much more difficult to make changes to the tables, forms, and reports **after** real data is entered. Double-check your design to make sure it contains all the data you will require BEFORE entering the real data.

<table>
<tr><th>STEP</th><th>EXAMPLE</th></tr>
<tr><td><u>1. Determine the purpose of your database</u>
This will help you decide what data you want your Access database to store.</td><td>Create a database to track clients, especially those currently in my rolodex. Keep detailed information about a clients <u>children</u> and <u>appointments</u>.</td></tr>
<tr><td><u>2. Determine the fields that you will need</u>
Decide what specific pieces of information you want to store.
Naming Convention
Incorrect ⇒ LASTNAME, Last Name
Correct ⇒ Last_Name, lastname or last_name</td><td>First name
Last name
Address
City
State
Zip
Appointment date
Spouse’s name
Appointment time
Children’s names
Kids birth date Kid’s
favorite color
Kid’s favorite food
What the kid likes to do?</td></tr>
<tr><td><u>3. Group related fields together</u>
Once you have a clear purpose for your database and know what pieces of information you want to track, begin to divide your information into separate subjects, such as "Employee" or "Orders". Each subject will be a table in your database.</td><td><u>Contacts</u>
First_Name
Last_Name
Address
City
State
Zip
Spouse_Name

<u>Appointments</u>
appointment_date
appointment_time

<u>Children</u>
Child_Name
Birth_Date
Favorite_Color
Favorite_Food
Favorite_Activity</td></tr>
<tr><td><u>4. Determine Keys and Relationships</u>
Look at each table and decide how the data in one table is related to the data in other tables.
o Common Key
field that will connect related tables (ContactID)
o Primary Key (*)
common field that cannot be repeated (a contact and his/her information)
o Foreign Key (**)
common key that can be repeated (a contact can have many children)</td><td><u>Contacts</u>
ContactID (*)
First_Name
Last_Name
Address
City
State
Zip
Spouse_Name

One Contact to Many Children

One Contact to Many Appointments

<u>Children</u>
ContactID (**)
Child_Name
Birth_Date
Favorite_Color
Favorite_Food
Favorite_Activity

<u>Appointments</u>
ContactID (**)
appointment_date
appointment_time</td></tr>
<tr><td><u>5. Determine Properties for each field</u>
o Data Type
text, number, yes/no, memo, date/time, currency
o Field Characteristics
size limit, required, case specifications, entry specifications</td><td>Contacts Table
ContactID — auto number, primary key
First_Name — text, 25 length
Last_Name — text, 50 length
Address — text, 50 length
City — text, 50 length
State — text, 2 length, upper case
Zip — text, 10 length
Spouse_Name — text, 25 length</td></tr>
<tr><td><u>6. Construct a test database</u></td><td>Enter a contact and children’s information.</td></tr>
</table>