Part 1
Know what a database is.
Understand why they are useful and when you might want to use one.
Have a basic understanding of how the most common type of databases, "Relational Databases" are organized.

Part 2
Learning how to use Microsoft Access, a database management system
- Create a database
- Add information to the database
- Retrieve information from the database

Part 3
Lab exercises

What is a database?
A database is a collection of data that is organized in a systematic way.
The data stored in a database is generally about a single topic. For example:
- Patients’ files in a hospital
- The contents of an address book
- A catalog of movies in a video store
Computers make large databases possible

Digital computers have made electronic databases possible, which facilitate

- the **storage** of very large quantities of information
- the efficient **addition, modification** or **removal** of that information
- rapid **search and retrieval** of desired information

Modern society relies heavily on the electronic databases that digital computers make possible.

---

**Question:**

What websites have you visited that likely use a database?

These websites use databases

- A library catalogue
- Online stores or auction websites

Database management systems play a central role in the vast majority of modern businesses.

---

**What is NOT a database?**

The following are collections of data, and they can be "organized" but they are not considered databases.

- a file-system (e.g. the "C: drive" on your computer)
- a word-processing document
- a text-file that is written and edited by hand (e.g. using notepad, emacs)

The difference lies in **how databases are organized.**
Imagine you want to keep track of films that you have watched.

A non-database solution is to use a plain text file (or a word processing document)
But this could lead to many problems -- especially as the size of your "database" grew...

In this format, the data is not easily searched.
Example: What if you want to know which movies are directed by Ben Affleck (and not those in which he is the star)?

It would be very difficult to transform this data into another format, e.g. to display it on a website.

In this format, it is not easy to edit the data in the database, e.g. changing the rating from an "out of 5 stars" system to an "out of 10 stars" would involve a lot of manual labour.

In this format, there is nothing that prevents errors from creeping in, e.g. the same film being added twice?

Database Management Systems (DBMS) such as Microsoft Access, MySQL, and Oracle can help you avoid these problems (but they don't 100% prevent them).

...they also don't work on their own. To properly take advantage of DBMS requires knowledge of database design...

... and thus, knowledge of database design is central to a whole career path.
Search google for "database analyst jobs"!
Relational and Non-Relational Databases

There are two broad classes of databases, relational and non-relational.

We are going to focus on relational databases, which were introduced in the late 1970's and remain the most widely used approach today.

Tables, Records and Fields

In a relational database, data is organized into tables, where:

- Each row in a table represents a record -- a "thing"
- Each column in a table represents a field -- an "attribute"

A collection of tables form a database.

But this is all most easily seen with an example. Here we see a single table that contains the attributes of several students.

<table>
<thead>
<tr>
<th>StudentId</th>
<th>Name</th>
<th>Address</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>C. Brown</td>
<td>12 Apple St.</td>
<td>555-1234</td>
</tr>
<tr>
<td>67890</td>
<td>L. Van Pelt</td>
<td>34 Pear Ave.</td>
<td>555-5678</td>
</tr>
<tr>
<td>22222</td>
<td>P. Patty</td>
<td>56 Grape Blvd.</td>
<td>555-9999</td>
</tr>
</tbody>
</table>

Relationships Between Tables

The structure of databases comes from relationships, which are connections between records in different tables.

<table>
<thead>
<tr>
<th>Students</th>
<th>Enrollments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Course Code</td>
</tr>
<tr>
<td>5468975</td>
<td>COMPSCI101</td>
</tr>
<tr>
<td>1258956</td>
<td>COMPSCI101</td>
</tr>
<tr>
<td>1258956</td>
<td>COMPSCI107</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Courses</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPSCI11</td>
<td>Practical Computing</td>
</tr>
<tr>
<td>COMPSCI101</td>
<td>Programming</td>
</tr>
<tr>
<td>COMPSCI107</td>
<td>Advanced Computing</td>
</tr>
</tbody>
</table>

How do relationships work?

There are two parts to a relationship, a primary key and a foreign key.

The primary key is a unique way of identifying a record in the table.

Generally:
- every table will have a primary key field
- all records must have a value in the primary key field
- the primary key’s value must be unique
Question
Which field makes a good primary key in this table?

<table>
<thead>
<tr>
<th>Courses</th>
<th>Code</th>
<th>Title</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COMPSCI111</td>
<td>Practical Computing</td>
<td>SS 2016</td>
</tr>
<tr>
<td></td>
<td>COMPSCI101</td>
<td>Programming</td>
<td>S1 2016</td>
</tr>
<tr>
<td></td>
<td>COMPSCI107</td>
<td>Advanced Computing</td>
<td>S1 2016</td>
</tr>
</tbody>
</table>

Types of Relationships

There are three kinds of relationship that one can build using primary keys (PK) and foreign keys (FK).

One to many
- One record (i.e., one row) in Table A can be related to multiple records in Table B.

One to one
- One record in Table A can be related to exactly one record in Table B.

Many to many
- Multiple records in Table A can be related to multiple records in Table B.

Foreign keys

A foreign key refers to a primary key in another table. This creates a connection or relationship between the two records.

<table>
<thead>
<tr>
<th>Enrolments</th>
<th>StudentID</th>
<th>Code</th>
<th>Date enrolled</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5468975</td>
<td>COMPSCI101</td>
<td>01/03/2016</td>
</tr>
<tr>
<td></td>
<td>1258956</td>
<td>COMPSCI101</td>
<td>15/12/2015</td>
</tr>
<tr>
<td></td>
<td>1258956</td>
<td>COMPSCI107</td>
<td>15/12/2015</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Courses</th>
<th>Code</th>
<th>Title</th>
<th>Semester</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COMPSCI111</td>
<td>Practical Computing</td>
<td>SS 2016</td>
</tr>
<tr>
<td></td>
<td>COMPSCI101</td>
<td>Programming</td>
<td>S1 2016</td>
</tr>
<tr>
<td></td>
<td>COMPSCI107</td>
<td>Advanced Computing</td>
<td>S1 2016</td>
</tr>
</tbody>
</table>

One to many

One record in Table A can be related to multiple records in Table B.

Example: Each enrollment refers to a single student. But each student can have several enrollments.

One record in the Students Table can be related to multiple records in the Enrolments Table (but not vice versa).
One to one

One record in Table A is related to exactly one record in Table B.

This relationship is relatively rare, as often it makes more sense for Table A and Table B to be merged into a single table. Sometimes this relationship is used when some data is costly to access and less frequently accessed.

### Relationship diagrams

**ACTORS**
- ID (primary key)
- Full Name
- Date of Birth

**PHOTOS**
- ID (primary key)
- ImageData

**ROLES**
- ID (primary key)
- Role
- ActorID (foreign key)
- FilmID (foreign key)

**FILMS**
- ID (primary key)
- Title
- Film Release Date

Many-to-many relationship

Here is an example of the many-to-many relationship.

The roles table can be thought of as saying:

- "Sarah Polley played the role of Ana in Dawn of the Dead"
- "Sarah Polley played the role of Elsa in Splice"
- "Ben Affleck played the role of Bruce Wayne in Batman vs. Superman"
- "Ving Rhames played the role of Kenneth Hall in Dawn of the Dead"

Each actor can be associated with multiple films. Each film can be associated with multiple actors!

### Q: Why is this database structure not as good as that on the previous slide?

**ACTORS**
- ID (primary key)
- Full Name
- Date of Birth

**PHOTOS**
- ID (primary key)
- ImageData

**ROLES**
- ID (primary key)
- Role
- ActorID (foreign key)
- FilmID (foreign key)

**FILMS**
- ID (primary key)
- Title
- Film Release Date
A: Data gets repeated..

The same data about films ends up getting repeated in the database...

1. makes it hard to identify and correct errors in the data
2. the introduction of errors more likely
3. increases the amount of storage used by the database

In this improved structure, data about each film is stored only once, and it is referred to in other tables.

Referential Integrity

One of the advantages of using a DBMS is it can force you to keep relationships valid.

A database with “referential integrity” is one in which “the references in the database all make sense”

More formally, referential integrity requires all values of a foreign key field to be:

- present in the related primary key field, or
- Null (ie. blank)

Referential Integrity

ID in the Students table is a primary key.

StudentID in the Enrolments table is a foreign key referring to the primary key ID in the Students table.

EXERCISE

Which of the following changes to the database breaks the referential integrity?

A. Insert 0101010, Ella Fitz, 11/11/1944 into Students
B. Insert 9998881, COMPSCI101, 22/12/2015 into Enrolments
C. Insert 6697826, COMPSCI101, 16/12/2015 into Enrolments
D. Insert , COMPSCI101, 01/12/2015 into Enrolments
E. Delete row with ID 6697826 from Students
F. Delete row with ID 1258956 from Students
Recapitulation

A database is a collection of data that is **systematically organized**, so as to allow efficient **addition, modification, removal** and **retrieval**.

A relational database is a collection of **tables**, where each row of the table is a **record** and each column is a **field**.

Databases use **foreign keys** and **primary keys** to establish **relationships** between records on different tables.

A database has **referential integrity** when all of the values in all foreign key fields point to the primary key of an existing record in the appropriate table (or are null).