Recall: Structure of the database

- A database contains one or more tables
  - Tables include entities with attributes
  - There are relationships defined between the entities in the various tables
  - Retrieve information from the tables using queries
- We designed and partially implemented a simple library database in the previous lecture

We implemented this part in the previous lecture.
What is the relationship?

This relationship is 1-to-many:
- One publisher is responsible for many books.
- Each book has only one publisher.

- The two tables are joined using the publisher ID number.
- The publisher ID is the primary key for each entry in the publishers table.
  - Therefore, each publisher must have a unique publisher ID.
- The publisher ID is a foreign key for each entry in the books table and we have requested referential integrity
  - Therefore, the given publisher ID must exist in the publishers table.

Referential Integrity

PubID must reference an actual publisher

This relationship is many-to-many:
- One book may have several authors.
- One author may have written several books.

- We need a unique identifier for each book.
  - We already selected the ISBN as the primary key and asked Access to make sure that there are no duplicates
- We need a unique identifier for each author
  - We will define an author table with a unique ID for each author
How do we link one book with many authors?

- We DO want:
  - to link each book to one or more authors
- We DON'T want
  - to specify extra fields (author1, author2, author3, …)
    - this is wasteful and limits the max number of authors
  - to specify each book entry several times, naming a different author in each row
    - this duplicates all the other information about the book

Add a cross-reference table!

- Refine the design so that it includes another table that is a book-author cross reference
  - Each entity in the table is a single cross reference
    - Attribute: ISBN
    - Attribute: Author ID
  - No primary key
- Now we can break the many-to-many relationship into two 1-to-many relationships that we already know how to implement
Define the new relationships

Get the new view of the data

- Notice that this view has redundant data
  - That's okay, because we are not storing it this way, just presenting it
  - The redundant items (Alex, Another Press) came from a single entry in a table – they are guaranteed to be identical
Now we've implemented this entire schema.

View: All Books from “Another Press”

View: All Books by Alex

View: All info about a given ISBN