

## More Views

INFO/CSE 100, Autumn 2004  
Fluency in Information Technology

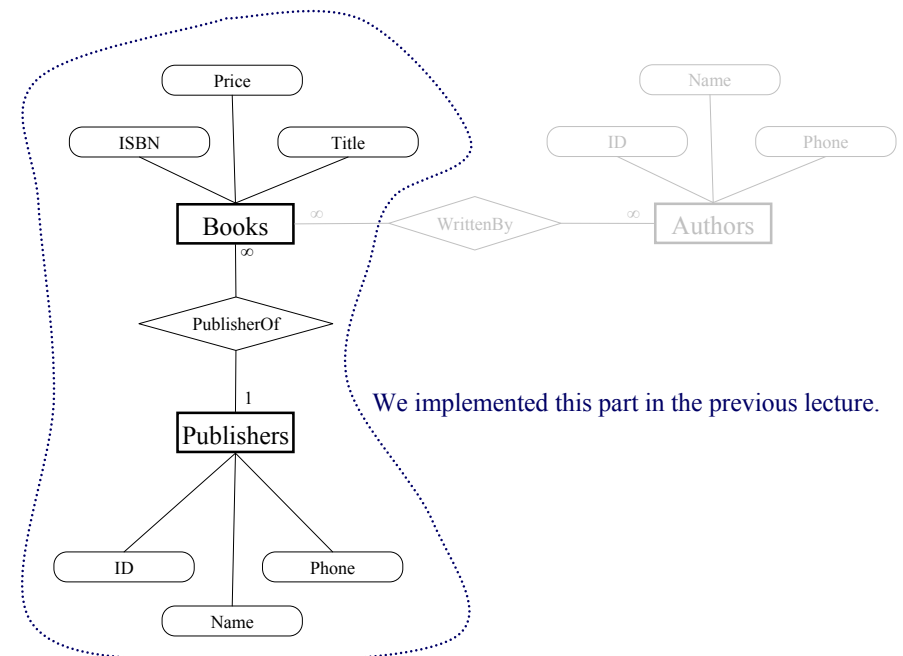
<http://www.cs.washington.edu/100>

## Readings and References

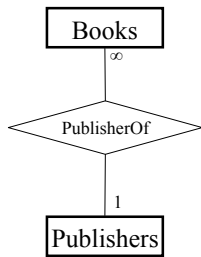
- Reading
  - » *Fluency with Information Technology*
    - Chapter 14, Database Queries
- References
  - » *Access Database: Design and Programming*
    - by Steve Roman, published by O'Reilly

## 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



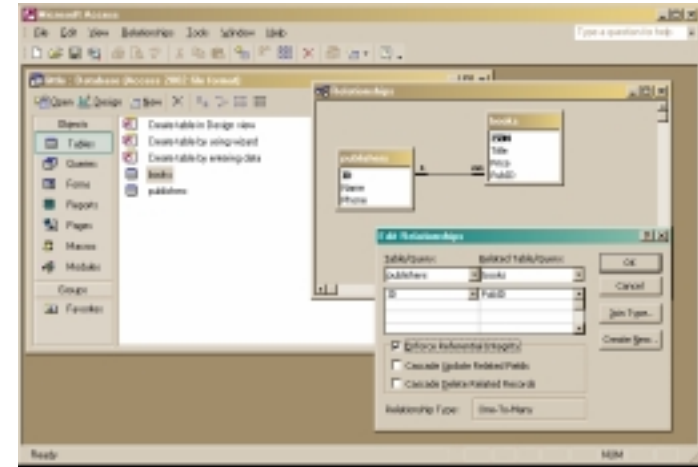
## What is the relationship?



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



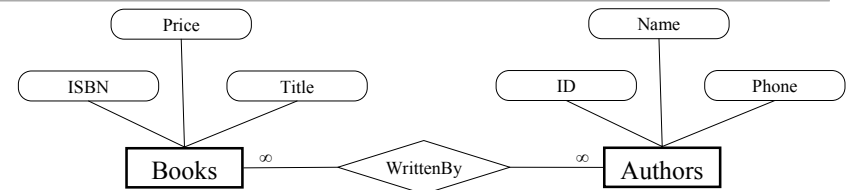
## PubID must reference an actual publisher

ISBN	Title	Price	PubID	ID	Name	Phone
	My Reader	\$10.00	1	1	A Press	555-1212
1-2	Your Reader	\$12.00	2	2	Another Press	555-3456
2-2	His Reader	\$25.00	2	2	Another Press	555-3456

```

All Books w/ all fields : Select Query
SELECT books.*, publishers.*
FROM publishers INNER JOIN books ON publishers.ID=books.PubID;
    
```

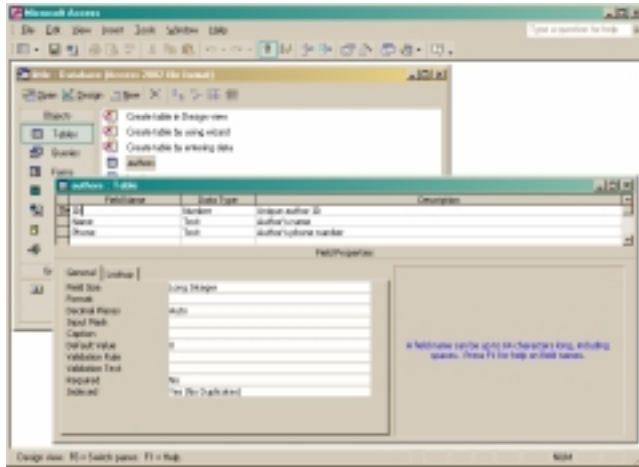
## What is the relationship?



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

## authors table



29-Nov-2004

cse100-22-views © 2004 University of Washington

9

## 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

29-Nov-2004

cse100-22-views © 2004 University of Washington

10

## Add a cross-reference table!

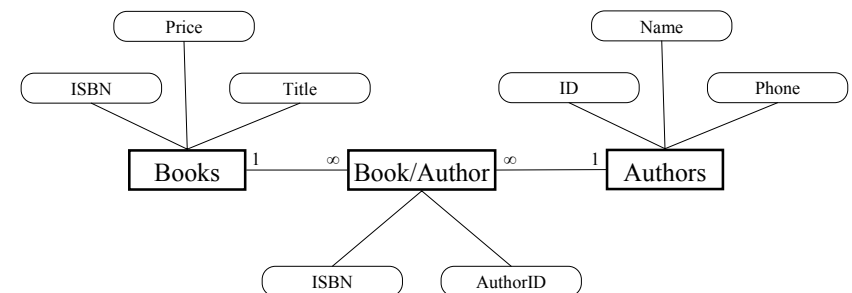
- Refine the design so that it includes another table that is a book-author cross reference
  - » Each entity in the table is a single cross reference
    - Attribute: ISBN
    - Attribute: Author ID
  - » No primary key
- Now we can break the many-to-many relationship into two 1-to-many relationships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

29-Nov-2004

cse100-22-views © 2004 University of Washington

11

## Define new cross-reference entities

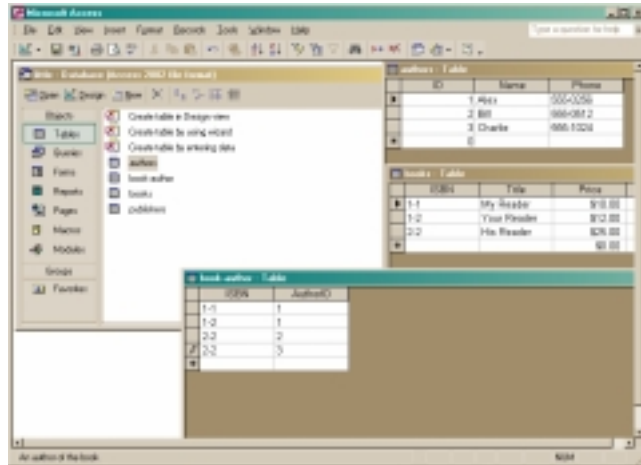


29-Nov-2004

cse100-22-views © 2004 University of Washington

12

## book-author table

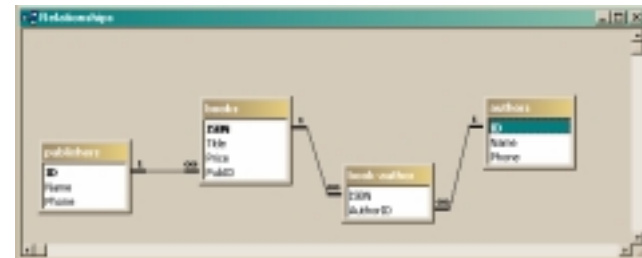


29-Nov-2004

cse100-22-views © 2004 University of Washington

13

## Define the new relationships



29-Nov-2004

cse100-22-views © 2004 University of Washington

14

## Define a query that uses the relationship

The screenshot shows the 'All Books w/Publishers, Authors : Select Query' design view. The design grid is as follows:

Field Name	Table	Field Name	Field Size	Field Properties
books	books	ISBN	Text	10
books	books	Title	Text	255
authors	authors	Name	Text	255
publishers	publishers	Name	Text	255

Below the design view is the 'Query By Example' window showing the actual SQL:

```

SELECT books.ISBN, books.Title, authors.Name, publishers.Name
FROM publishers INNER JOIN books INNER JOIN authors INNER JOIN [book-author] ON authors.ID = [book-author].AuthorID ON
books.ISBN = [book-author].ISBN ON publishers.ID = books.PubID;
    
```

Query By Example

actual SQL

## Get the new view of the data

The screenshot shows the results of the query in a table view:

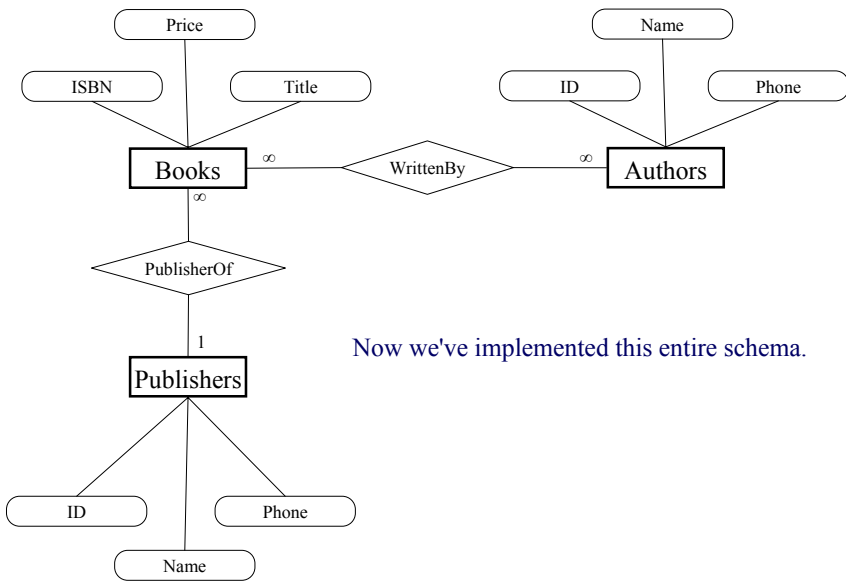
ISBN	Title	authors.Name	publishers.Name
1-1	My Reader	Alex	A Press
1-2	Your Reader	Alex	Another Press
2-2	His Reader	Bill	Another Press
2-2	His Reader	Charlie	Another Press

- 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

29-Nov-2004

cse100-22-views © 2004 University of Washington

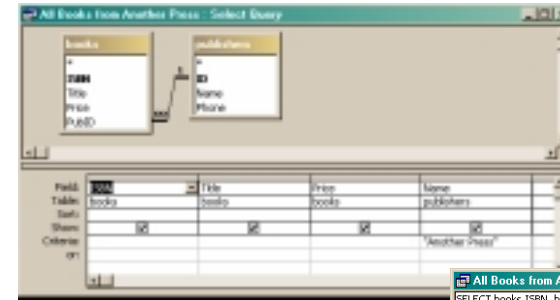
16



Now we've implemented this entire schema.

## View: All Books from "Another Press"

ISBN	Title	Price	Name
1-2	Your Reader	\$12.00	Another Press
2-2	His Reader	\$25.00	Another Press



```

SELECT books.ISBN, books.Title, books.Price, publishers.Name
FROM publishers INNER JOIN books ON publishers.ID=books.PubID
WHERE (((publishers.Name)="Another Press"));
  
```

## View: All Books by Alex



Name	ISBN	Title
Alex	1-1	My Reader
Alex	1-2	Your Reader

```

SELECT authors.Name, [book-author].ISBN, books.Title
FROM books INNER JOIN [authors] INNER JOIN [book-author] ON authors.ID=[book-author].AuthorID ON books.ISBN=[book-author].ISBN
WHERE (((authors.Name)="Alex"));
  
```

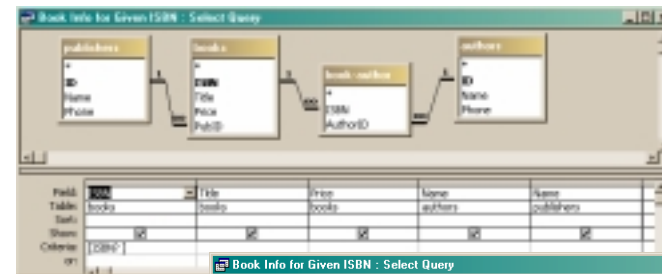
## View: All info about a given ISBN

Enter Parameter Value

ISBN? [1-1]

OK Cancel

ISBN	Title	Price	authors.Name	publishers.Name
1-1	My Reader	\$10.00	Alex	A Press



```

SELECT books.ISBN, books.Title, books.Price, authors.Name, publishers.Name
FROM publishers INNER JOIN (books INNER JOIN (authors INNER JOIN [book-author] ON authors.ID = [book-author].AuthorID) ON books.ISBN = [book-author].ISBN) ON publishers.ID = books.PubID
WHERE (((books.ISBN)=[ISBN]));
  
```