More Views

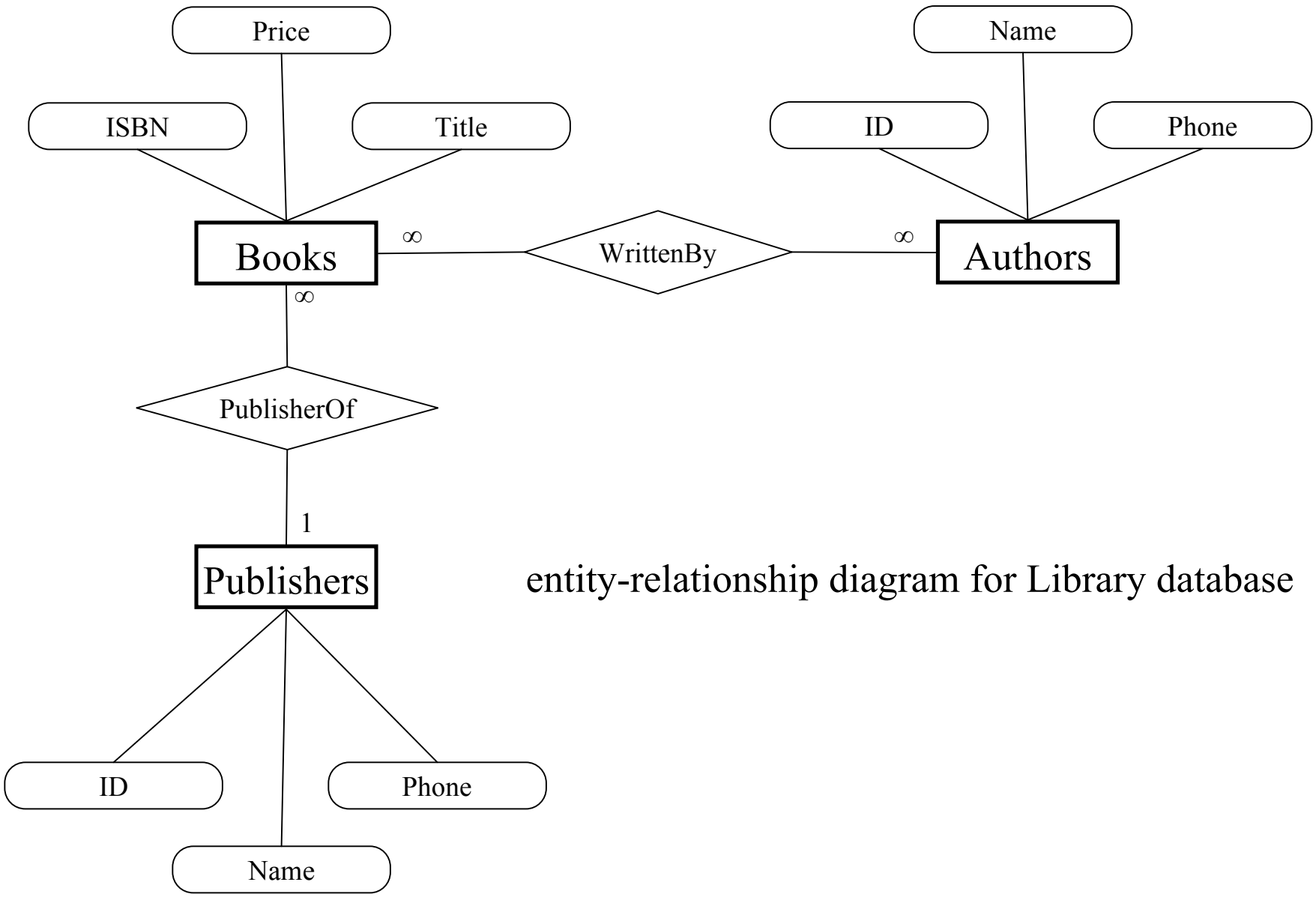
INFO/CSE 100, Spring 2005

Fluency in Information Technology

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

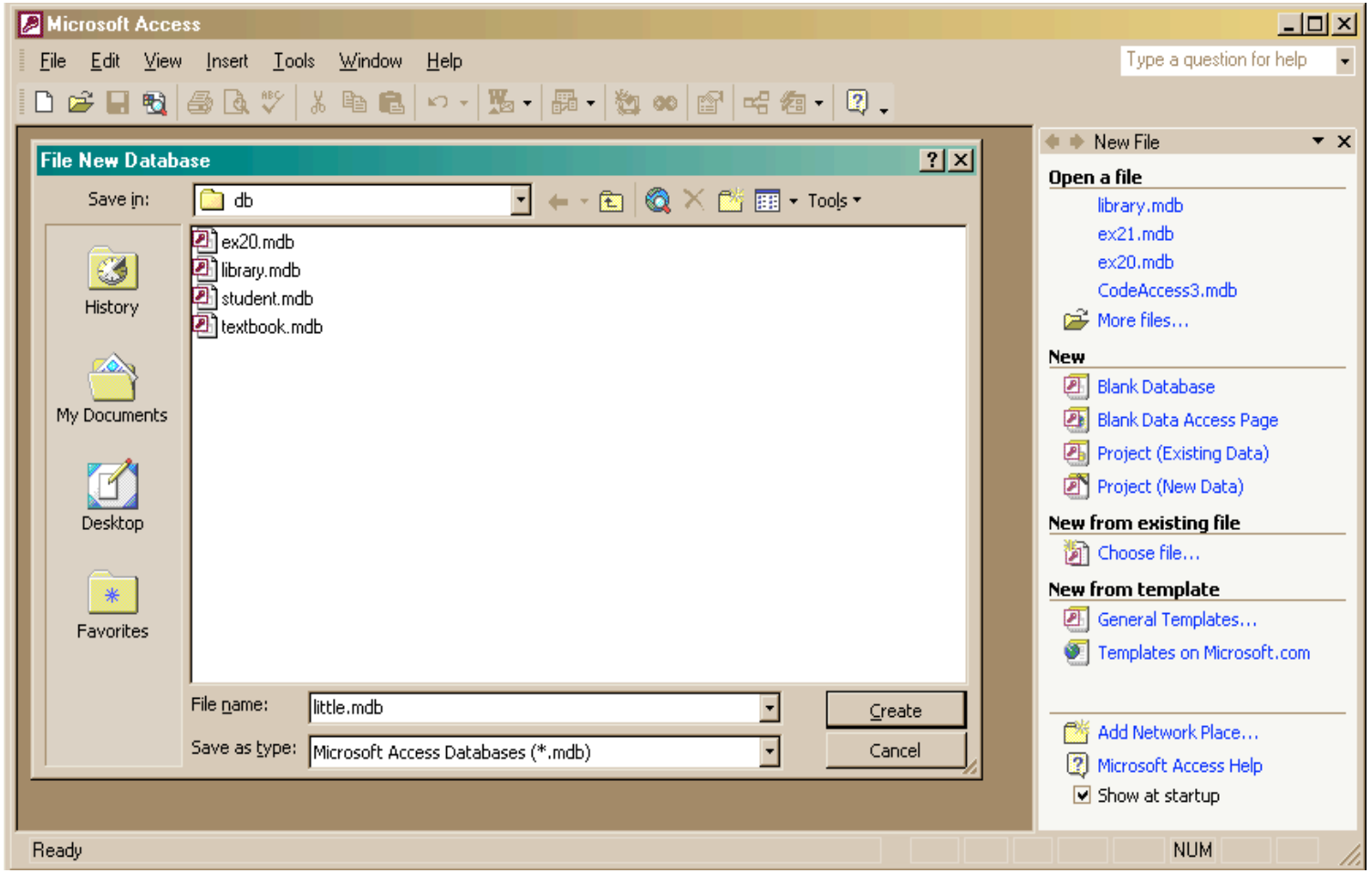
Readings and References

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

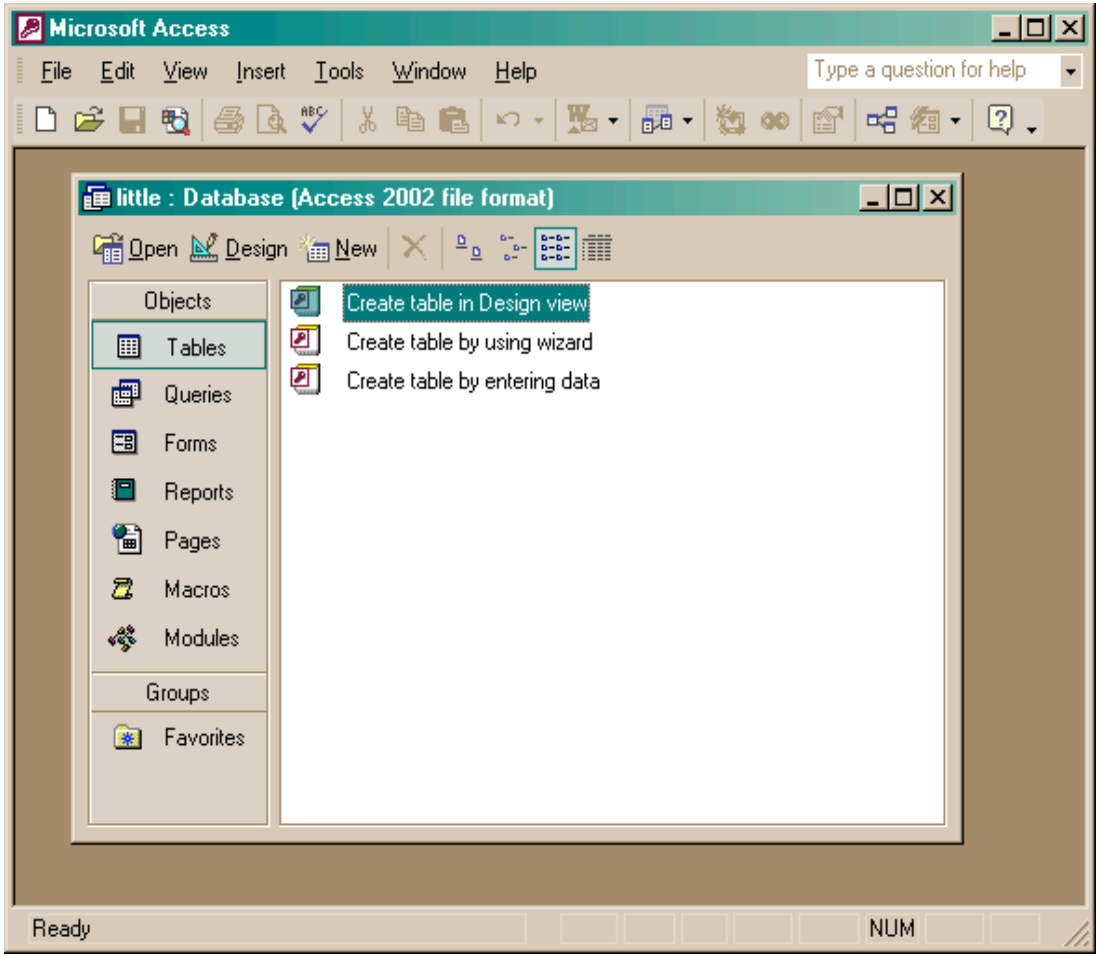


entity-relationship diagram for Library database

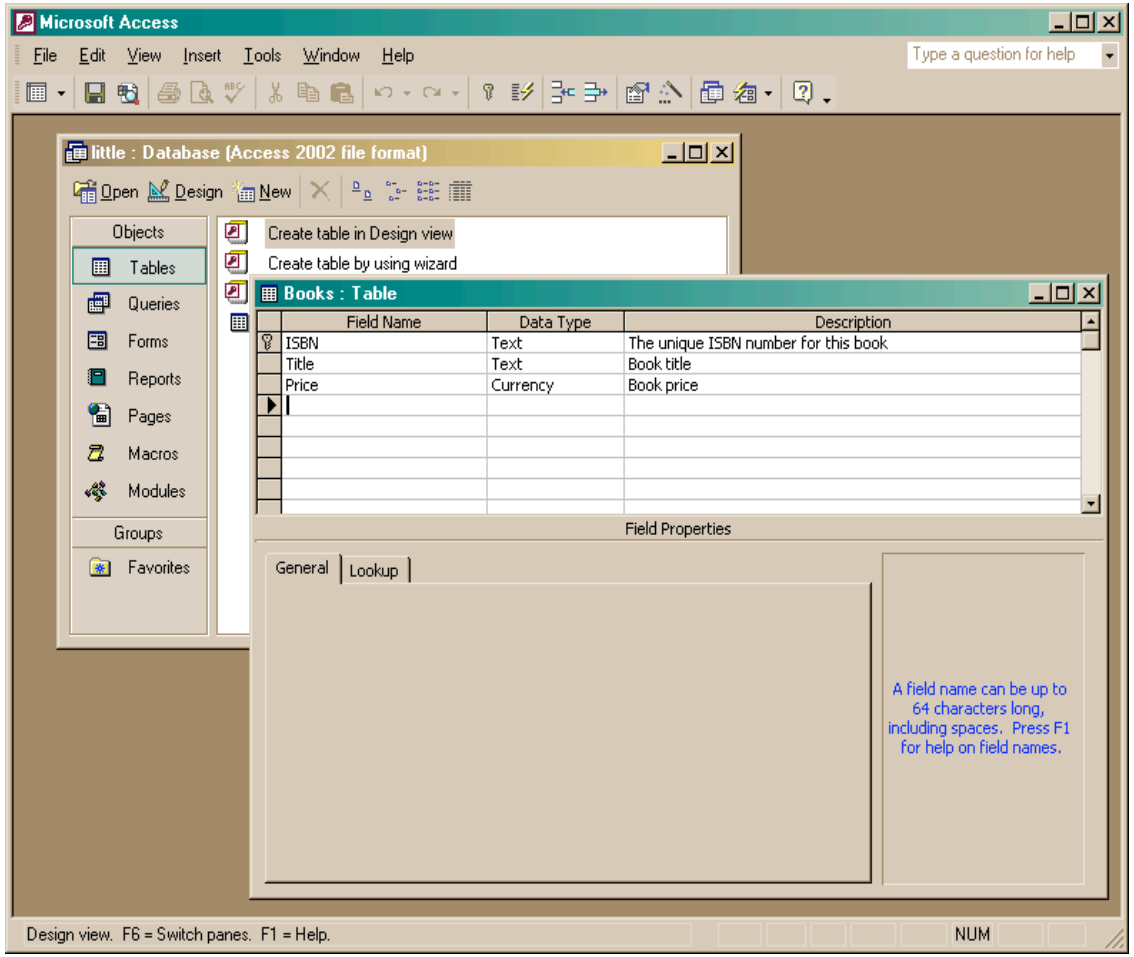
Create a new database



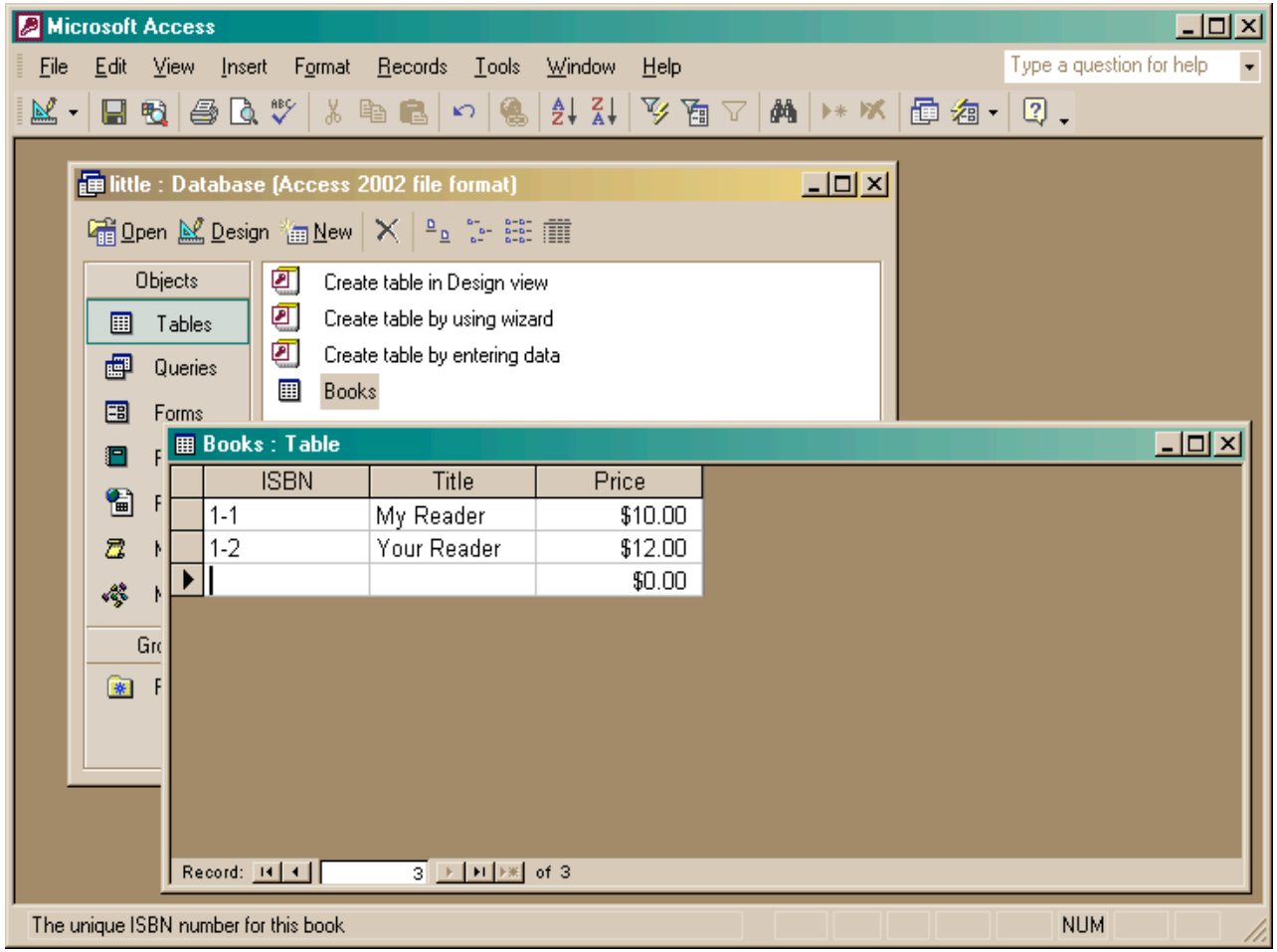
Create a new table in the database



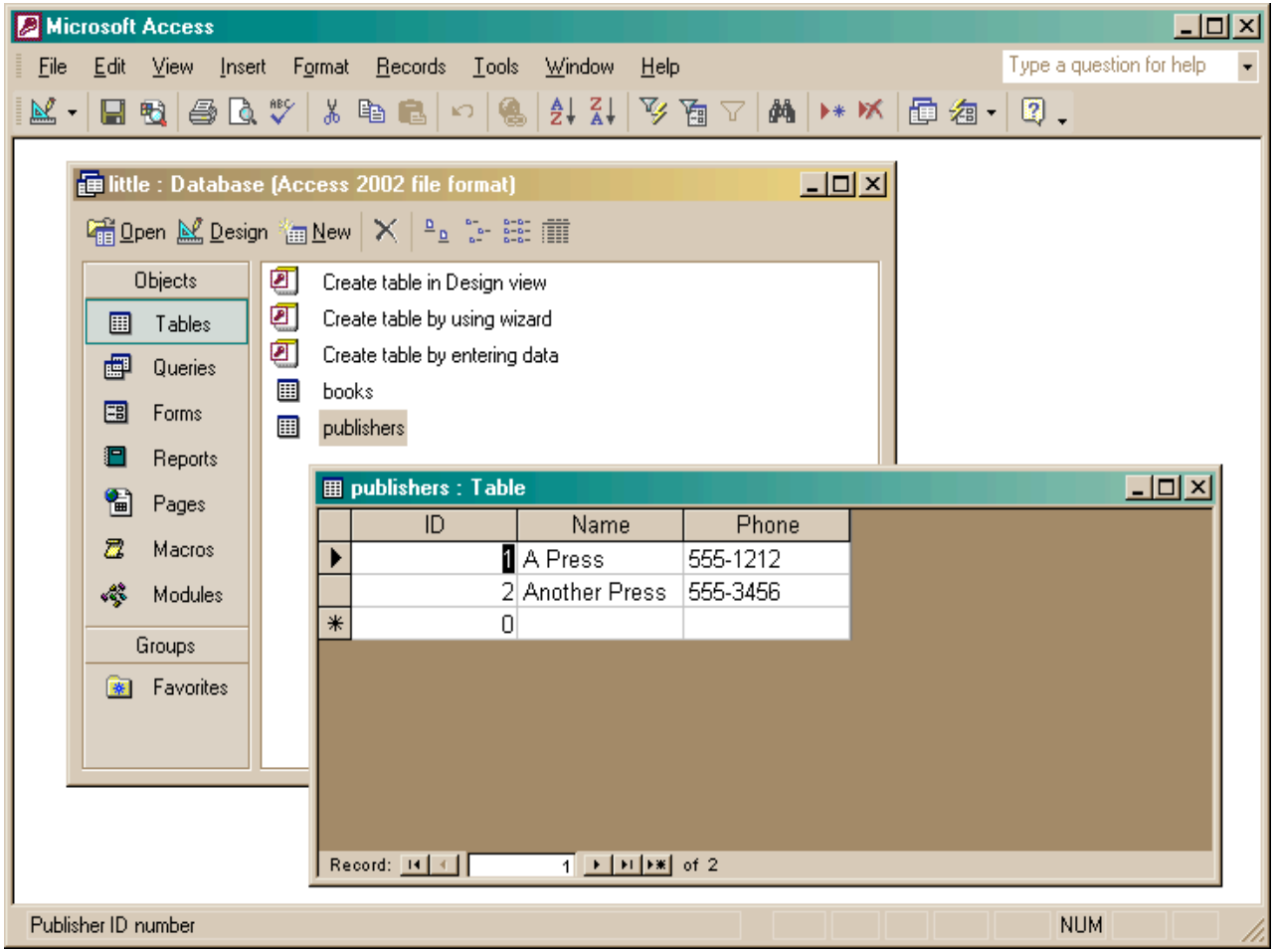
Creating a table in Design view



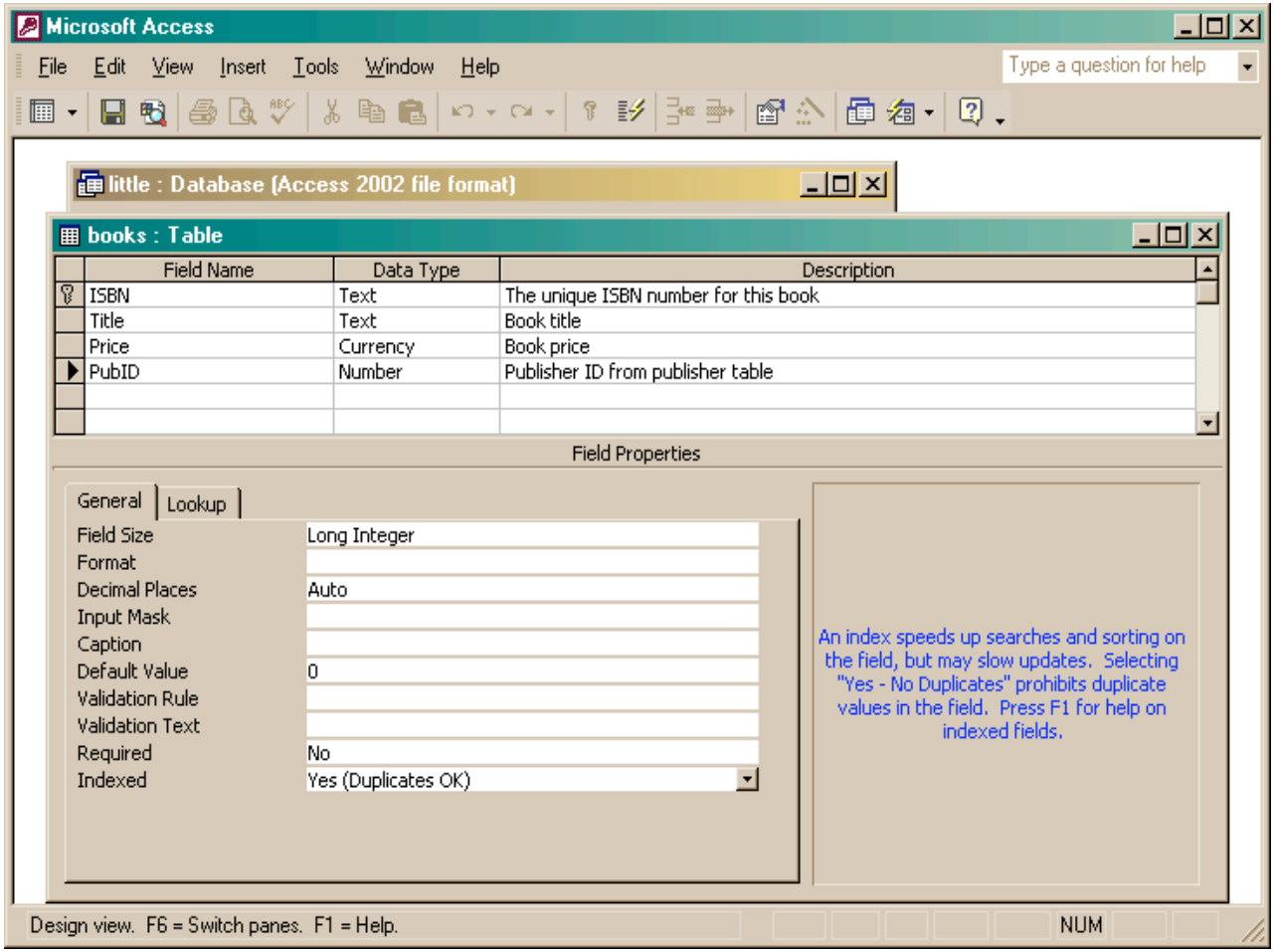
Entering Table Data



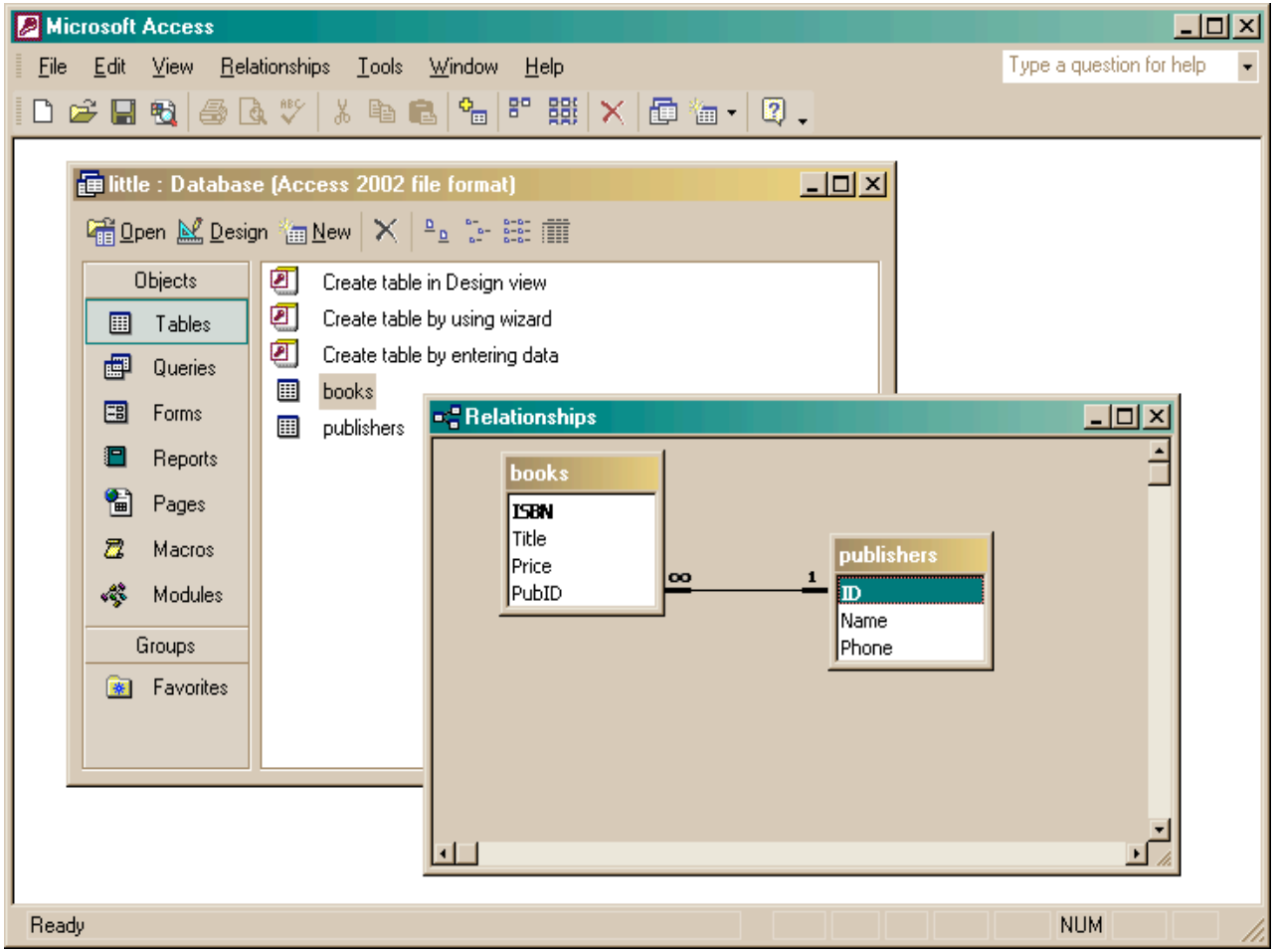
Build another table

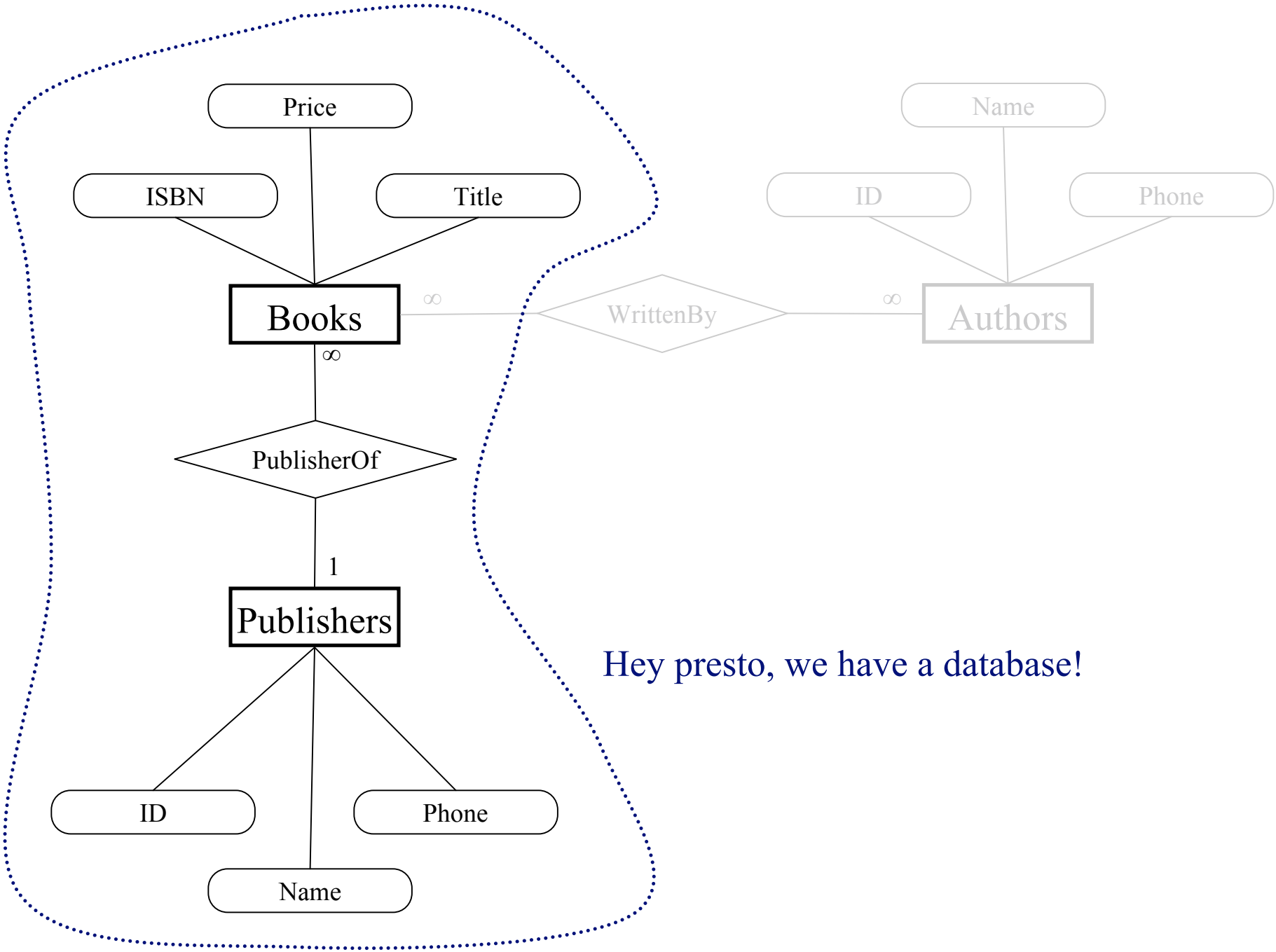


Add publisher ID to books



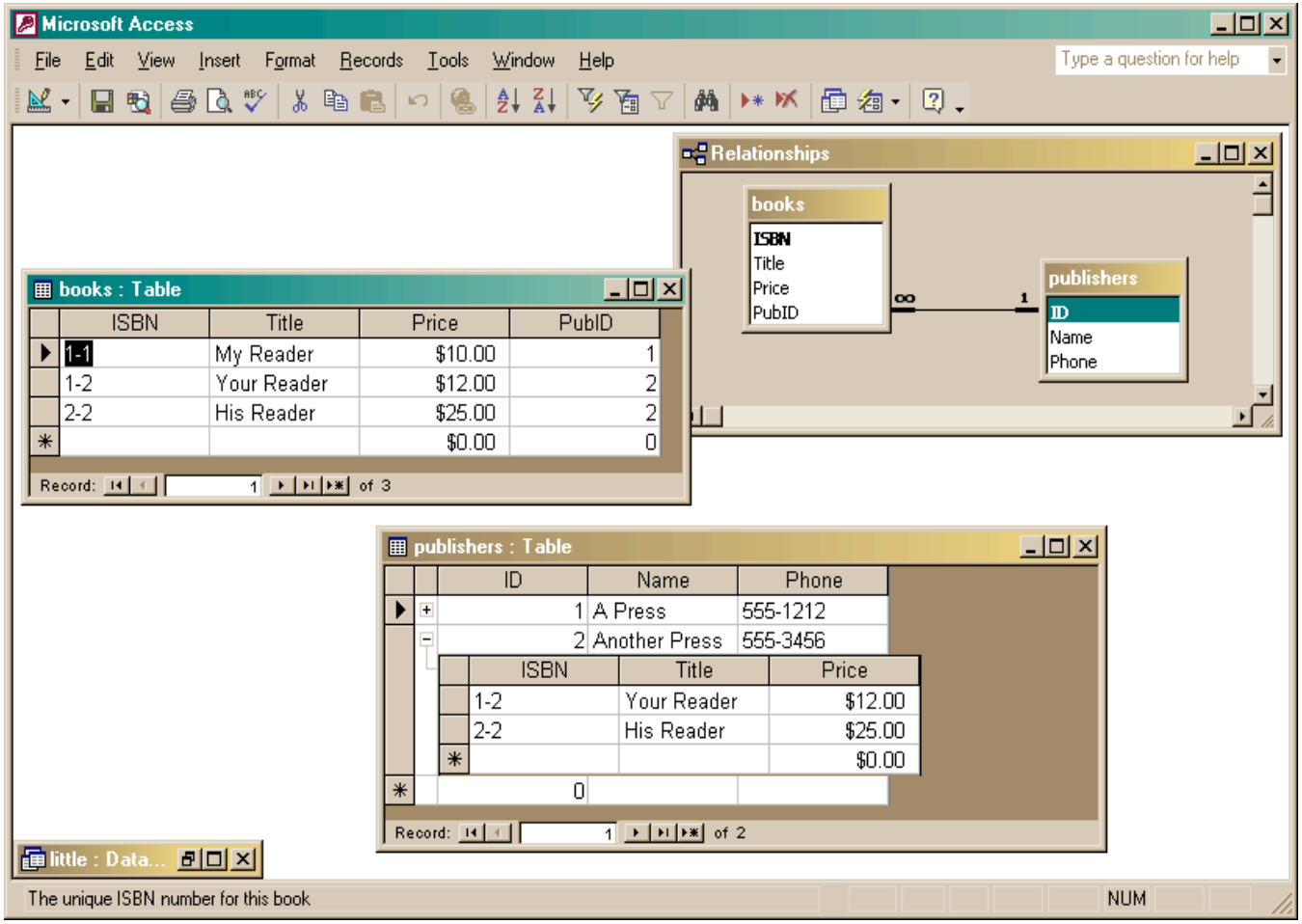
Create the link between the tables



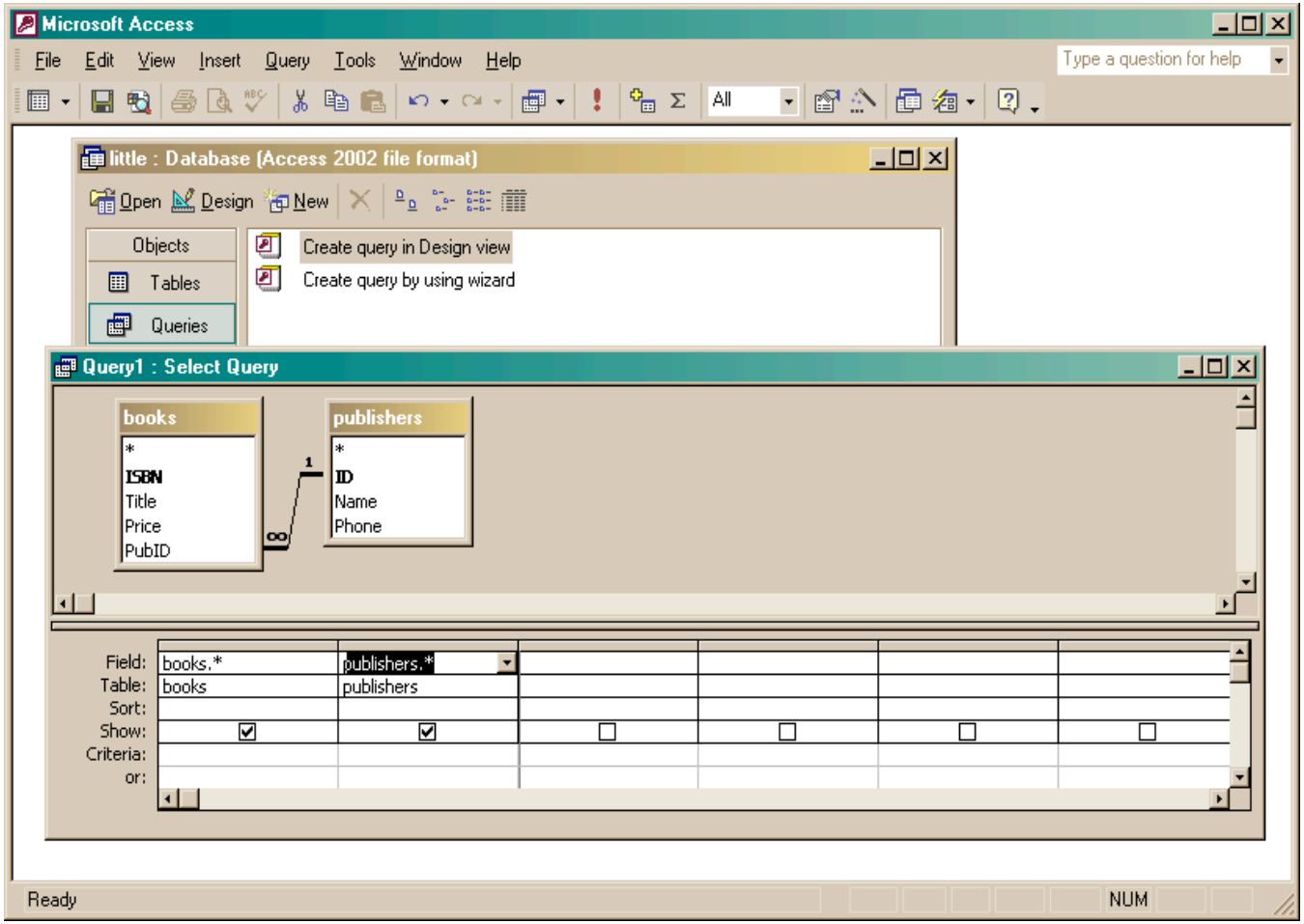


Hey presto, we have a database!

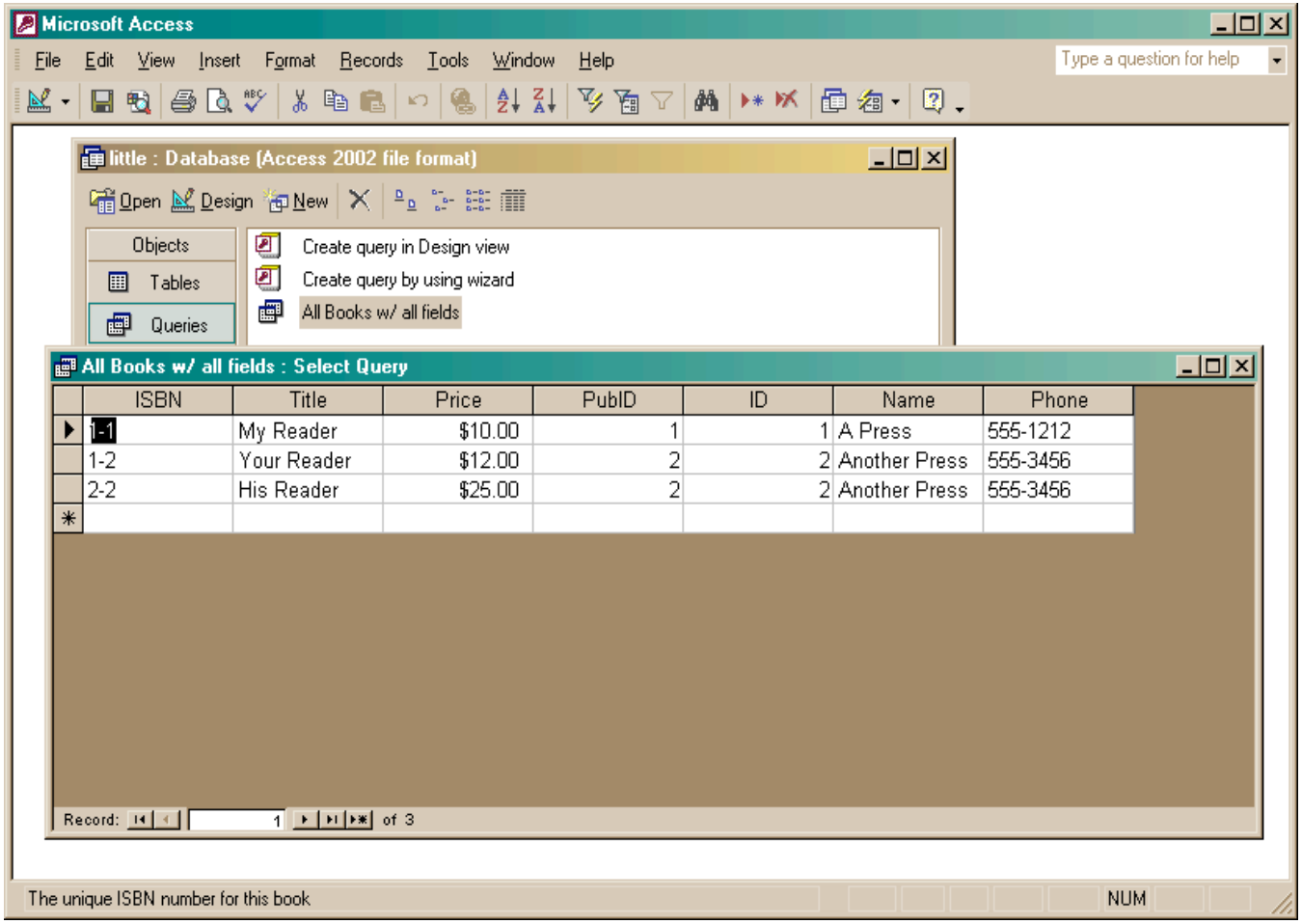
Two tables with a relationship



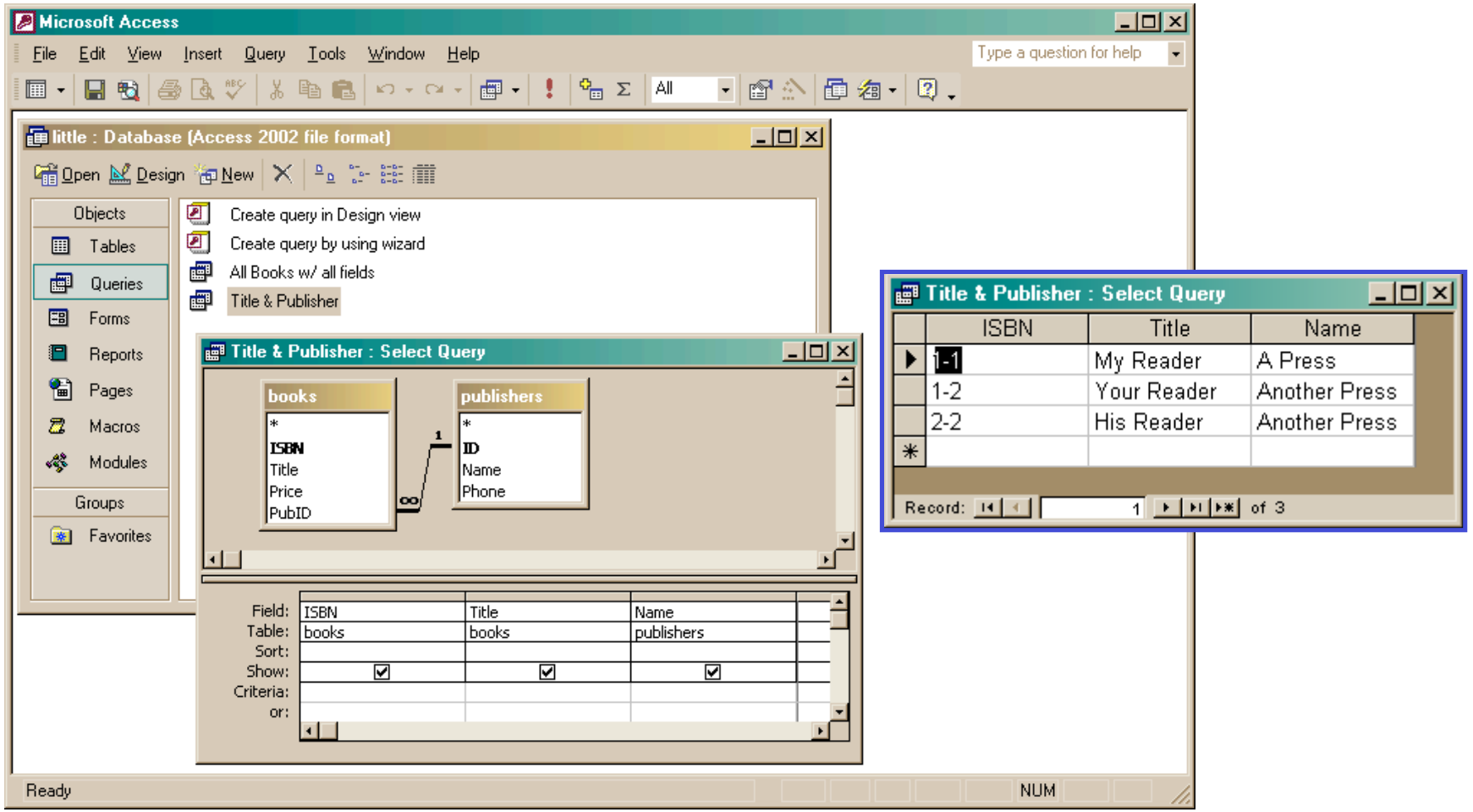
Create a query



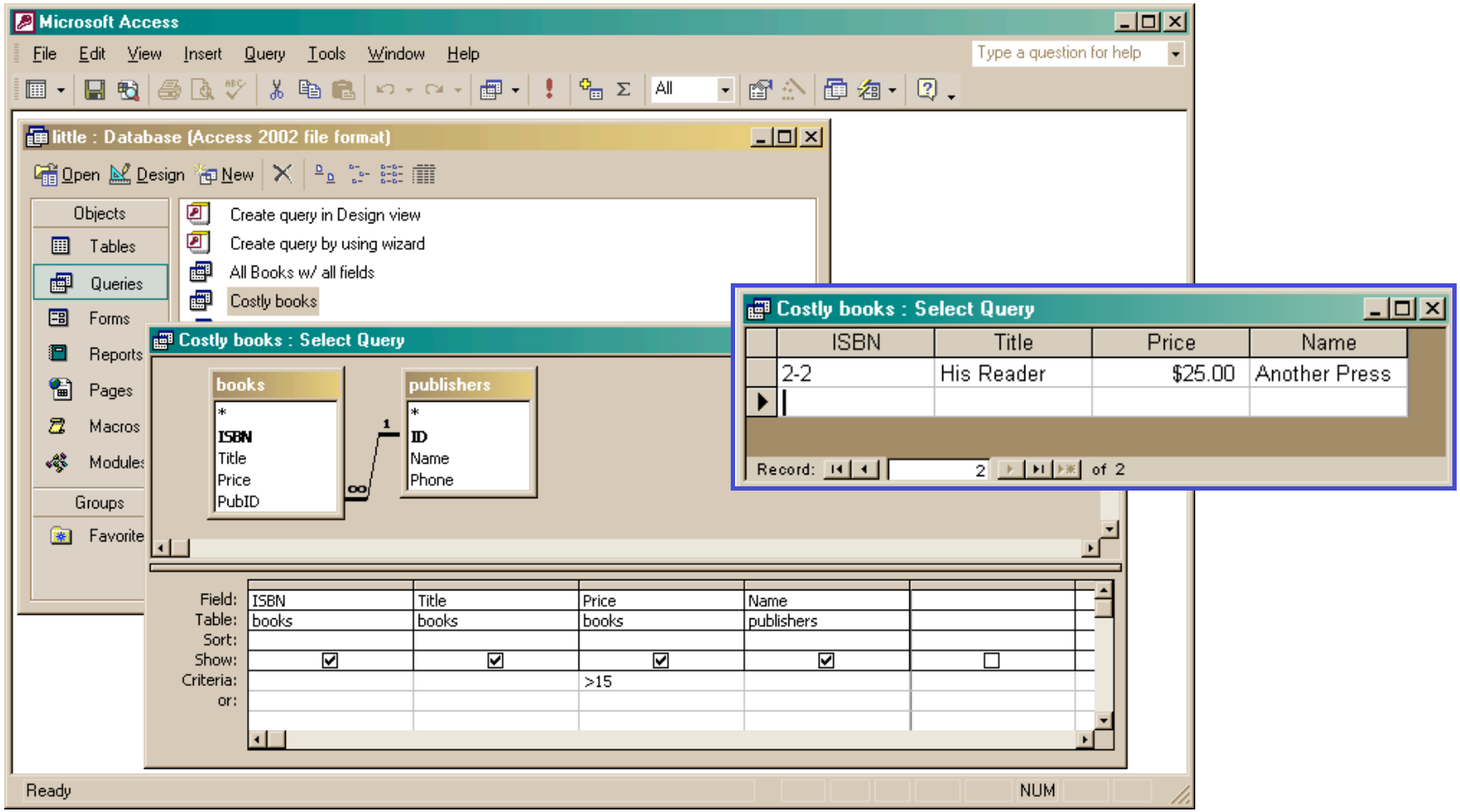
The query produces a new (virtual) table



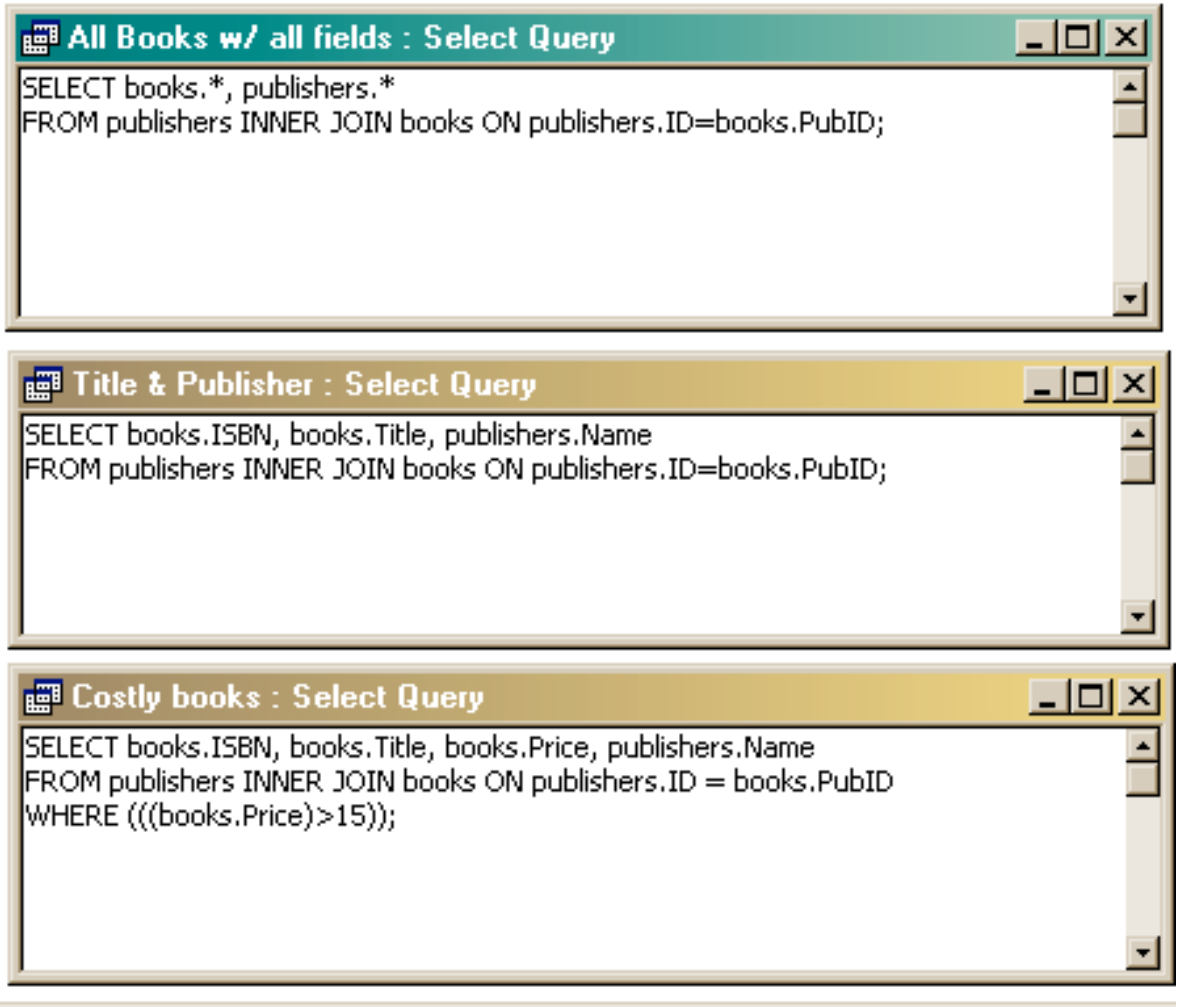
Project (select particular columns)



Select particular rows

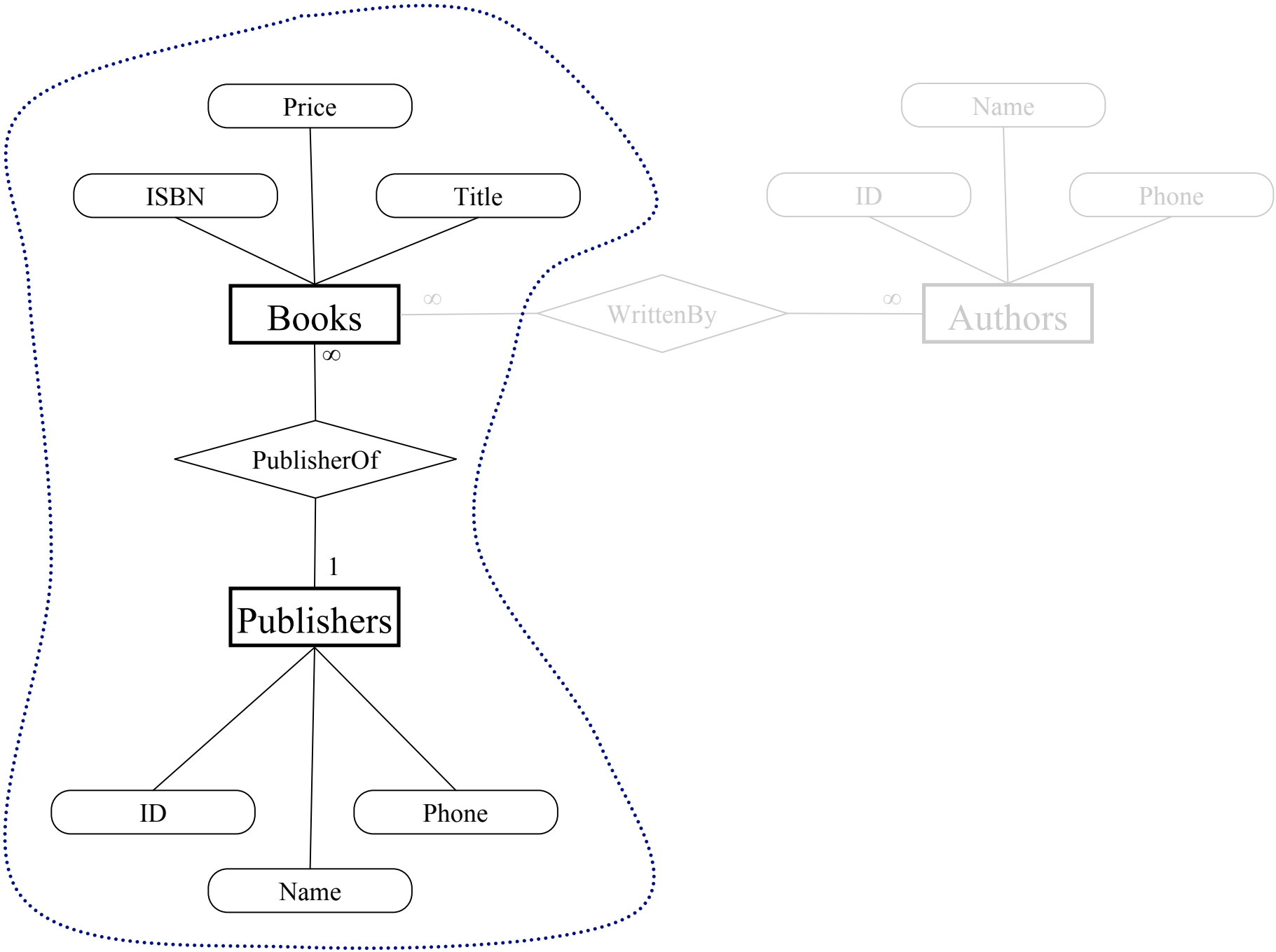


SQL behind the scenes

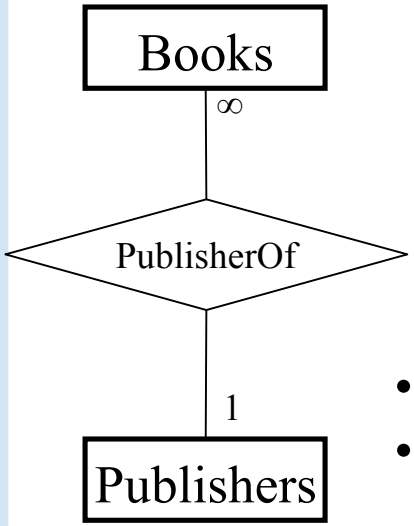


Recall: Structure of the database

- 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


Price
ISBN
Title
Books
∞
PublisherOf
1
Publishers
ID
Phone
Name
∞
WrittenBy
∞
Authors
Name
ID
Phone

What is the relationship?

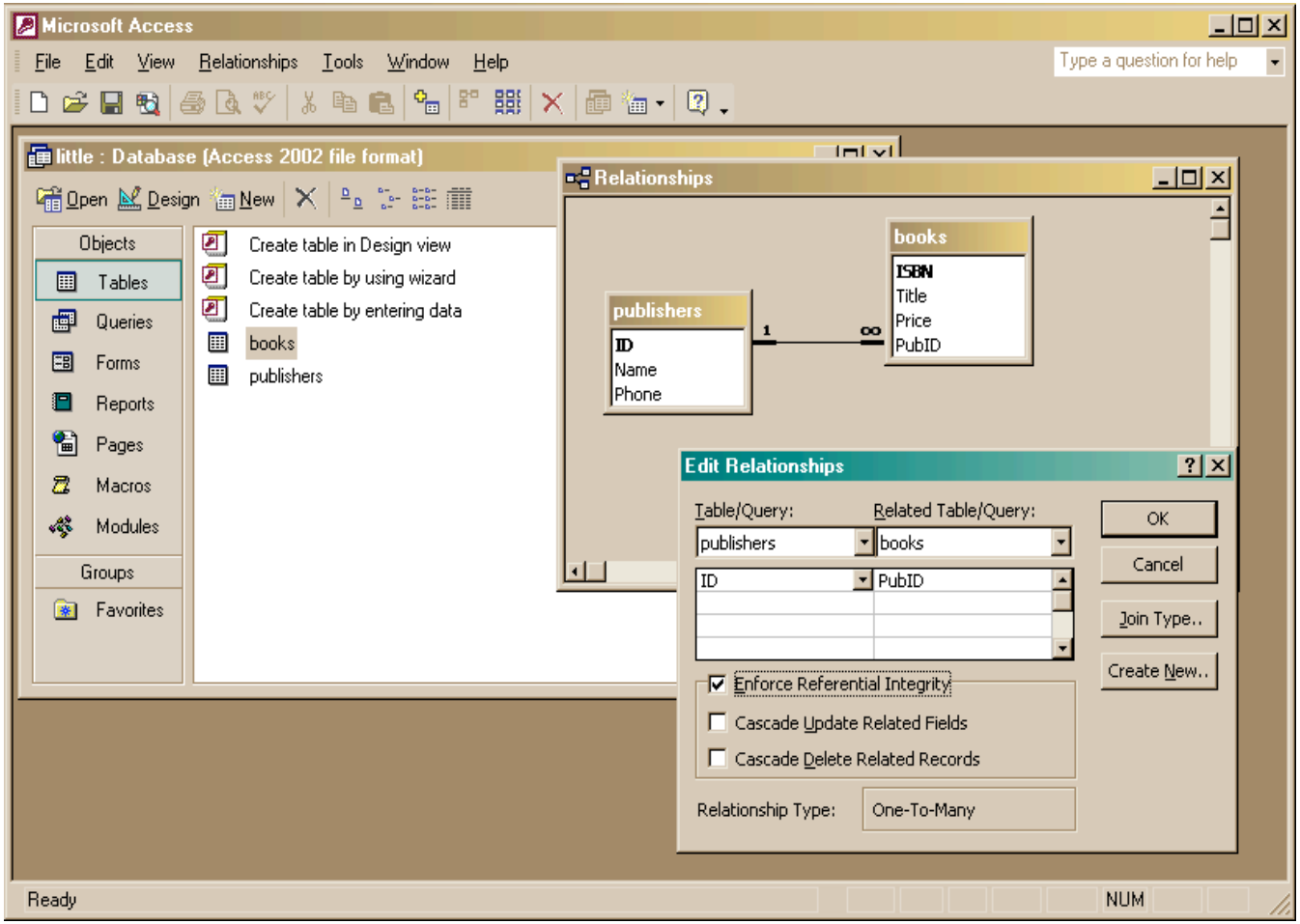


This relationship is 1-to-many:

- 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

All Books w/ all fields : Select Query

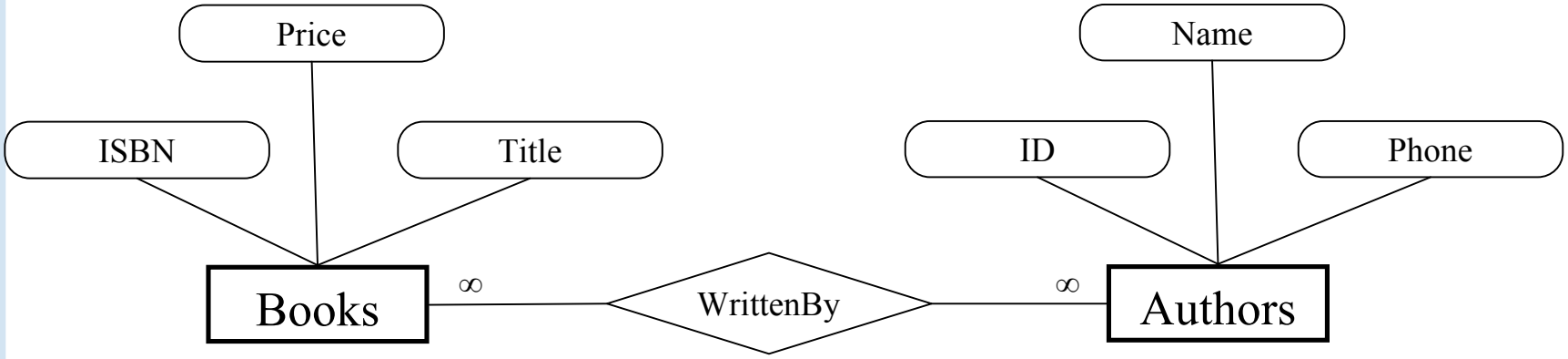
ISBN	Title	Price	PubID	ID	Name	Phone
1-1	My Reader	$10.00	1	1	A Press	555-1212
1-2	Your Reader	$12.00	2	2	Another Press	555-3456
2-2	His Reader	$25.00	2	2	Another Press	555-3456
*						

Record: 1 of 3

All Books w/ all fields : Select Query

```
SELECT books.*, publishers.*
FROM publishers INNER JOIN books ON publishers.ID=books.PubID;
```

What is the relationship?

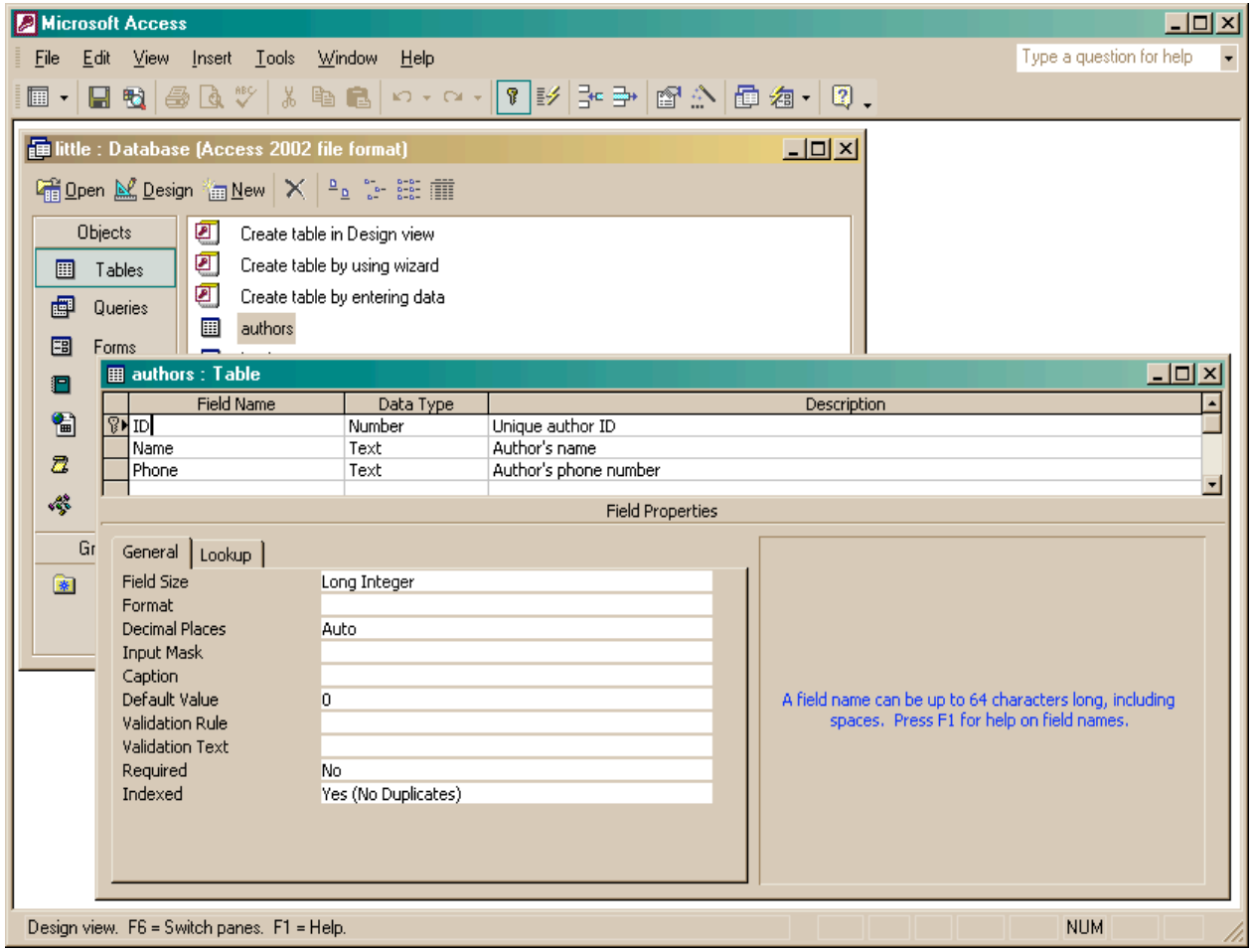


This relationship is many-to-many:

- 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



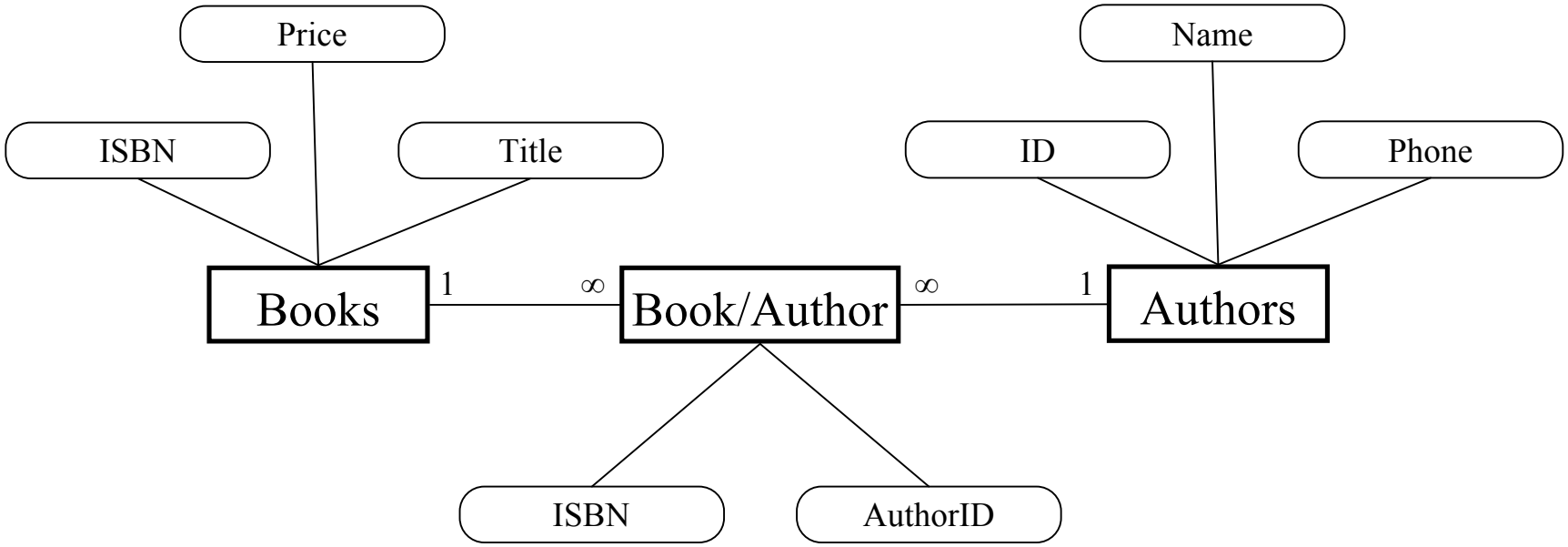
Link one book with many authors?

- 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

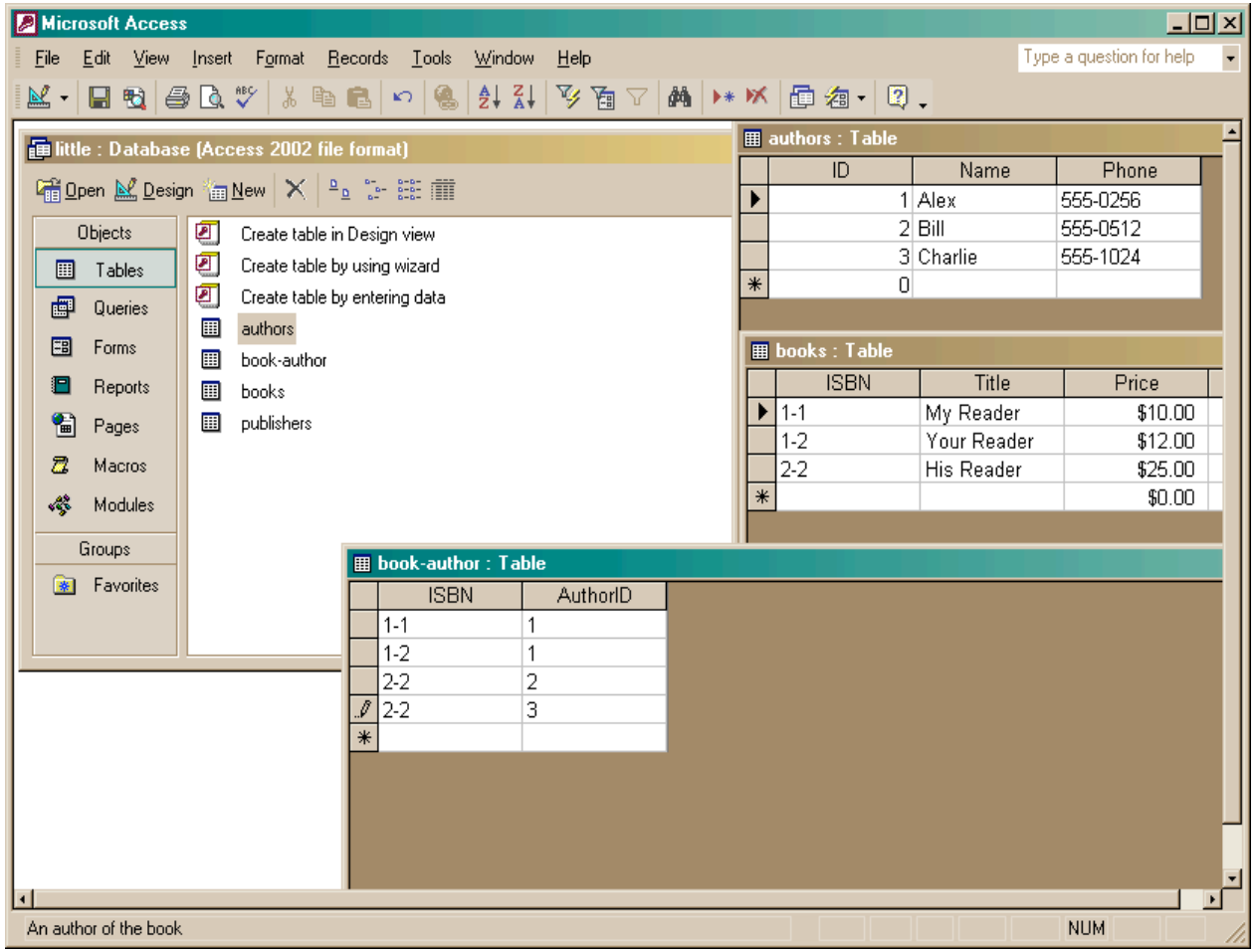
Add a cross-reference table!

- 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

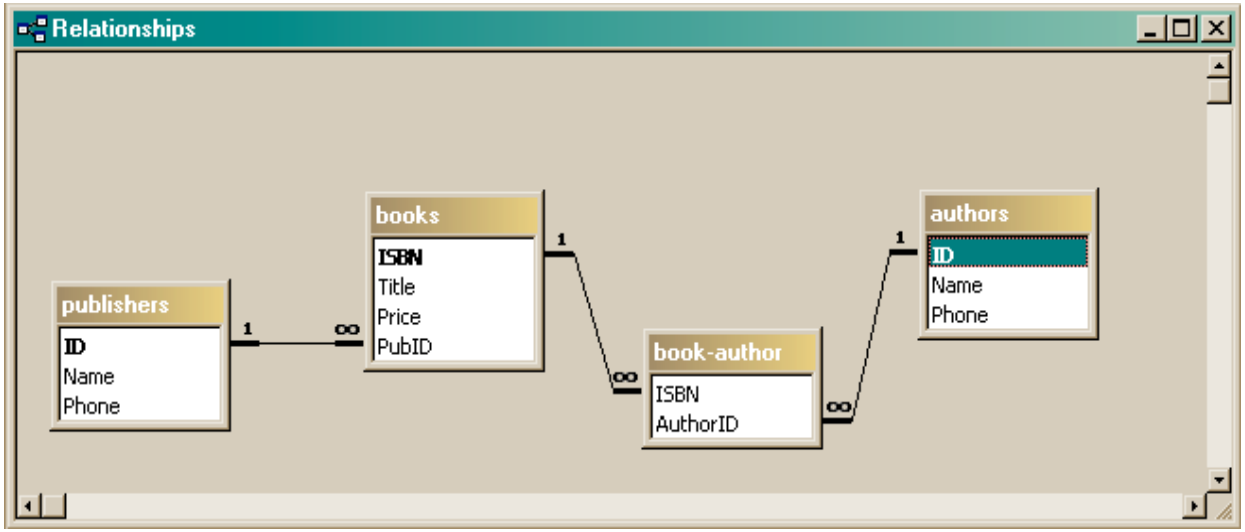
Define new cross-reference entities



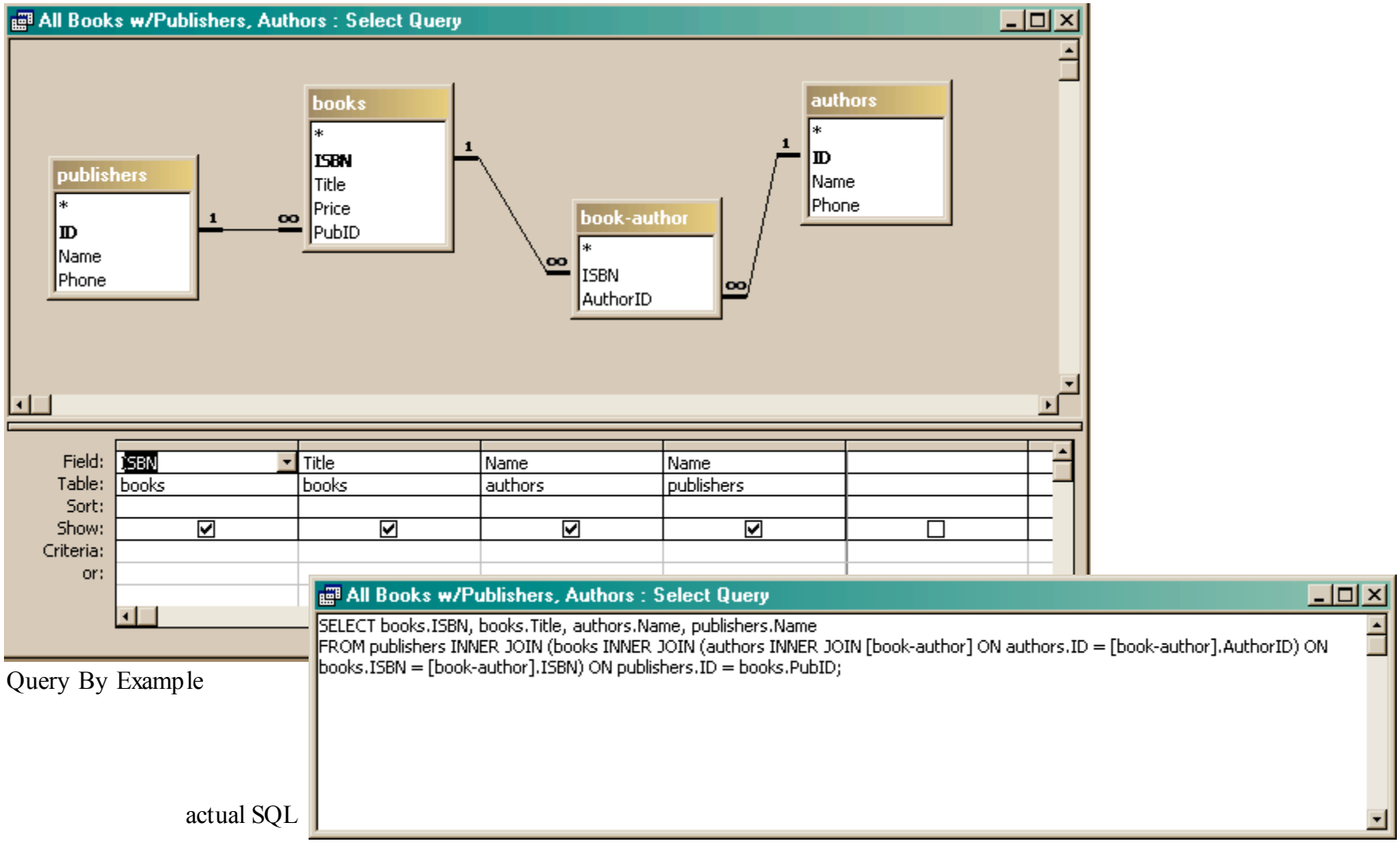
book-author table



Define the new relationships



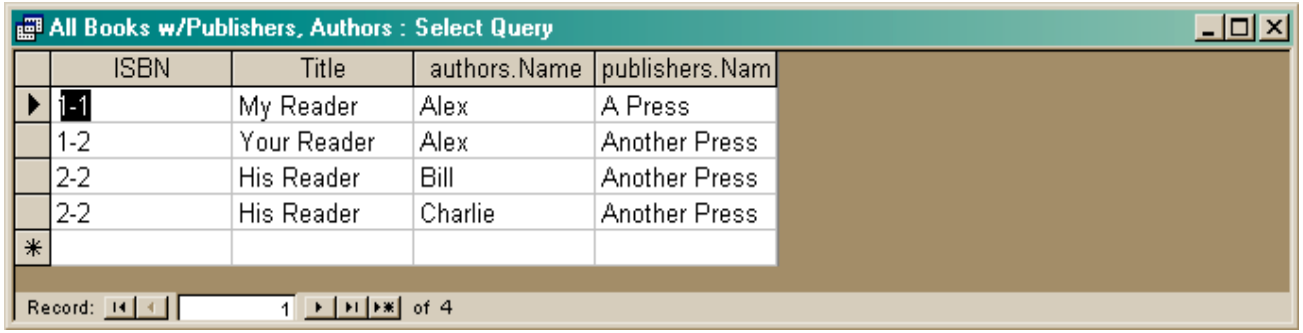
Define a query that uses the relationship



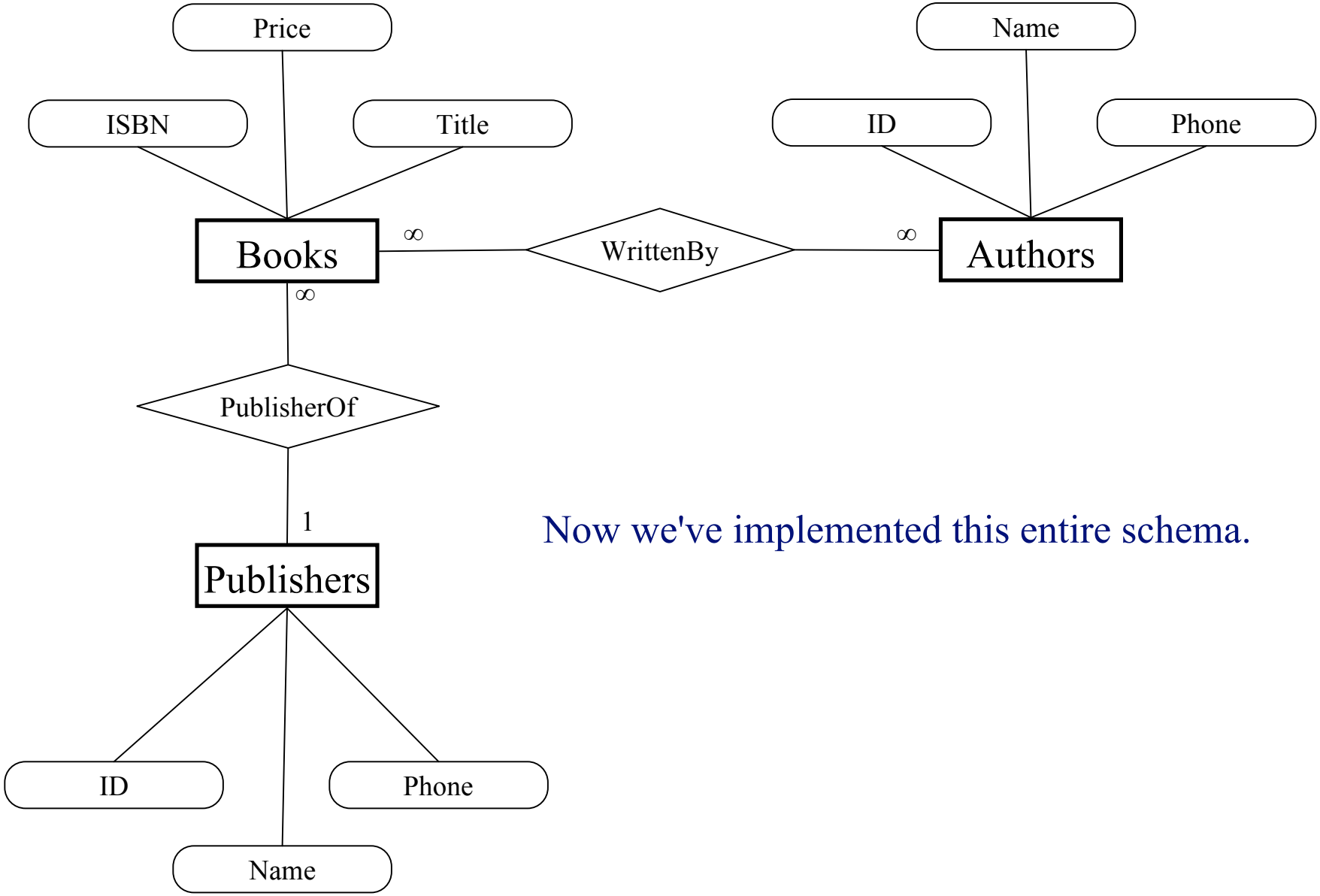
Query By Example

actual SQL

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”

All Books from Another Press : Select Query

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

Record: 3 of 3

All Books from Another Press : Select Query

books
*
ISBN
Title
Price
PubID

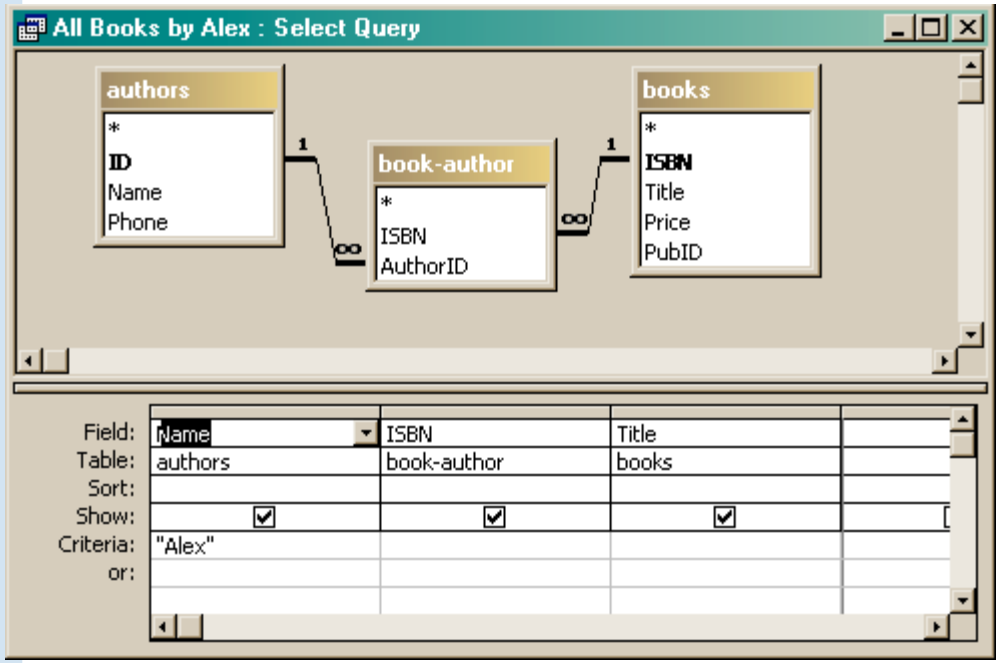
publishers
*
ID
Name
Phone

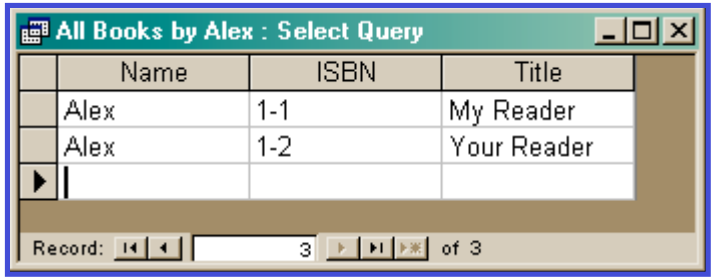
Field:	ISBN	Title	Price	Name
Table:	books	books	books	publishers
Sort:				
Show:	☑	☑	☑	☑
Criteria:				"Another Press"
or:				

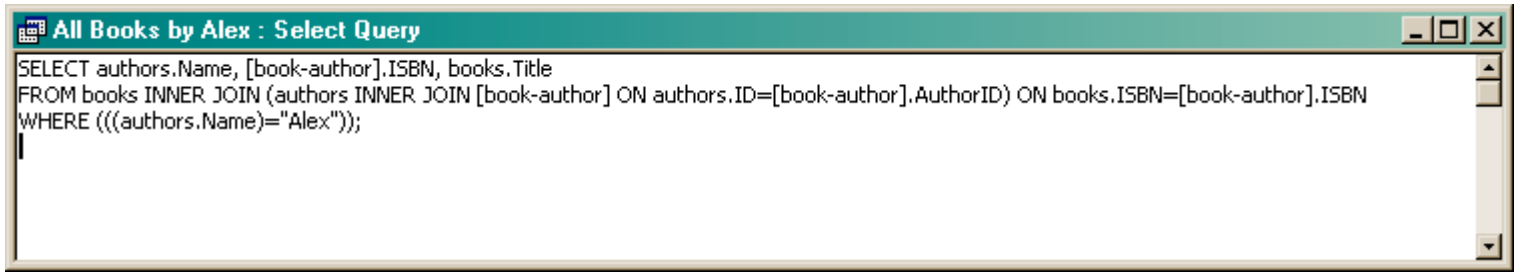
All Books from Another Press : Select Query

```
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







View: All info about a given ISBN