

Database Programming in .Net

The Relational Model

Jim Fawcett
CSE686 – Internet Programming
Summer 2005

Topics

- Introduction to Relational Databases
 - Entity relationship model
 - Database design
 - Redundancy
 - Normal forms
 - Queries
 - Structured Query Language (SQL)
- Brief Preview of .Net Data Object Model
 - DataReader : connected cursor
 - DataSet : disconnected table set

Introduction to Relational Databases

- A relational database is composed of one or more (usually more) tables.
- Each **table** consists of rows and columns.
 - A row is referred to as a **record**
 - A record refers to a single entity – person, place, or thing
 - Each record in the database is unique, no duplicates are allowed
 - A column is referred to as a **field** or **attribute**
 - a column holds a specified data type – an attribute of the record
 - Most tables define a **primary key**
 - The primary key is an attribute that serves to uniquely identify each record in the database.
 - An exception is a table used to describe a many-to-many relationship which contain two foreign keys, that is, primary keys of other tables (more later...).
- Relationships are defined between tables.
 - A relationship is a unique association between two tables
 - Usually a column in one table is filled with (foreign) primary keys from the other table

Tables and Normalization

- A database is partitioned into tables to reduce redundancy.
 - A Books database that uses a single table would reproduce publisher data and author data many times even though a book entity is only recorded once, as an author many have created more than one book and publishers have many books.
 - That wastes space.
 - Affords the possibility that the data becomes inconsistent, e.g., publisher data is recorded differently in different places.
 - An Author table that has a Book column, will need additional Books columns if the author has more than one book. How many Books columns should we use?
 - The table schema is now ambiguous.
 - These problems are resolved by using more tables, a process called Normalization.
 - There are three commonly used Normalization Forms, each designed to avoid one type of problem, liked those cited above.

Book Database with four Tables

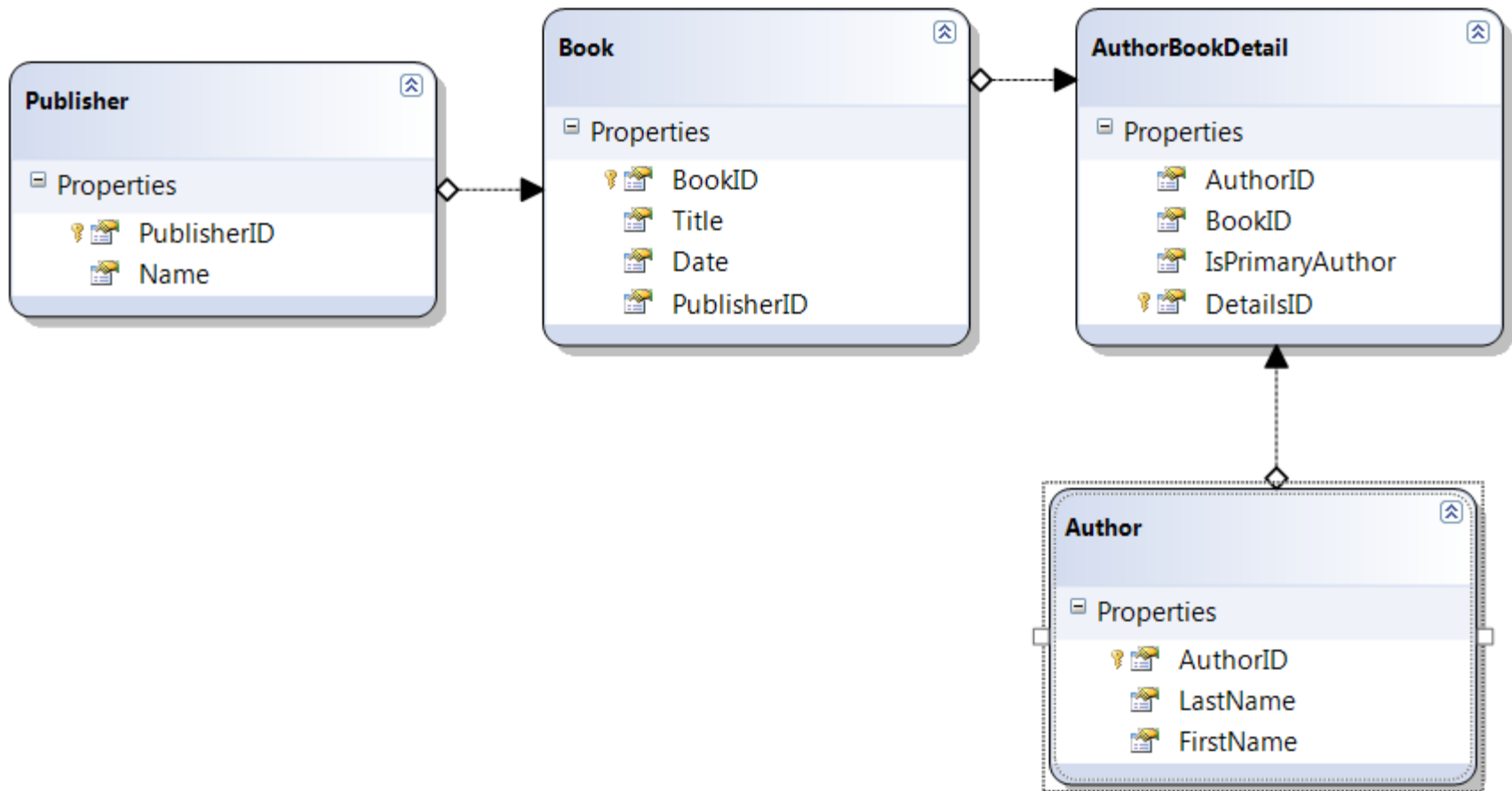


Table Details

dbo.Books: Table...ATA\BOOKSDB.MDF x Books.dbml* Books.designer.cs Book.cs

Column Name	Data Type	Allow Nulls
BookID	int	<input type="checkbox"/>
Title	nvarchar(MAX)	<input type="checkbox"/>
Date	nchar(10)	<input checked="" type="checkbox"/>
PublisherID	int	<input type="checkbox"/>

Column Properties

Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1
Indexable	Yes
Is Columnset	No

Identity Specification

Books: Query(ap...ATA\BOOKSDB.MDF) x dbo.Books: Table...ATA\BO

	BookID	Title	Date	PublisherID
▶	8	somebook	2010	1
	9	anotherbook	2011	1
	14	third book	2020	1
	15	fourth book	2030	2
	16	fifth	2040	1
	17	still another...	2090	2
*	NULL	NULL	NULL	NULL

Table Structure

- A table represents a class of a certain kind of entity, e.g., a book, an author, or a publisher.
- The first row represents the scheme of the table, e.g., its set of attributes. These attributes provide information about an entity
 - Book(Title, ISBN, Date, PubID)
 - Author(LastName, FirstName, Phone)
 - Publisher(Name, Phone)
- Each row of the table, after the first, represents a unique entity, one specific book, author, or publisher.

Keys

- A super key is a set of one or more attributes that serve to uniquely identify each entity in a table's domain.
 - The domain of the table is the set of all possible entities that can be entered into the table. It is identified by the attributes of the table.
- A key is a minimal super key. That is, if any attribute from the key set is removed the remaining set of attributes no longer uniquely identify each record in the table's domain.
 - Usually a table's key consists of a single attribute, e.g., BookID.

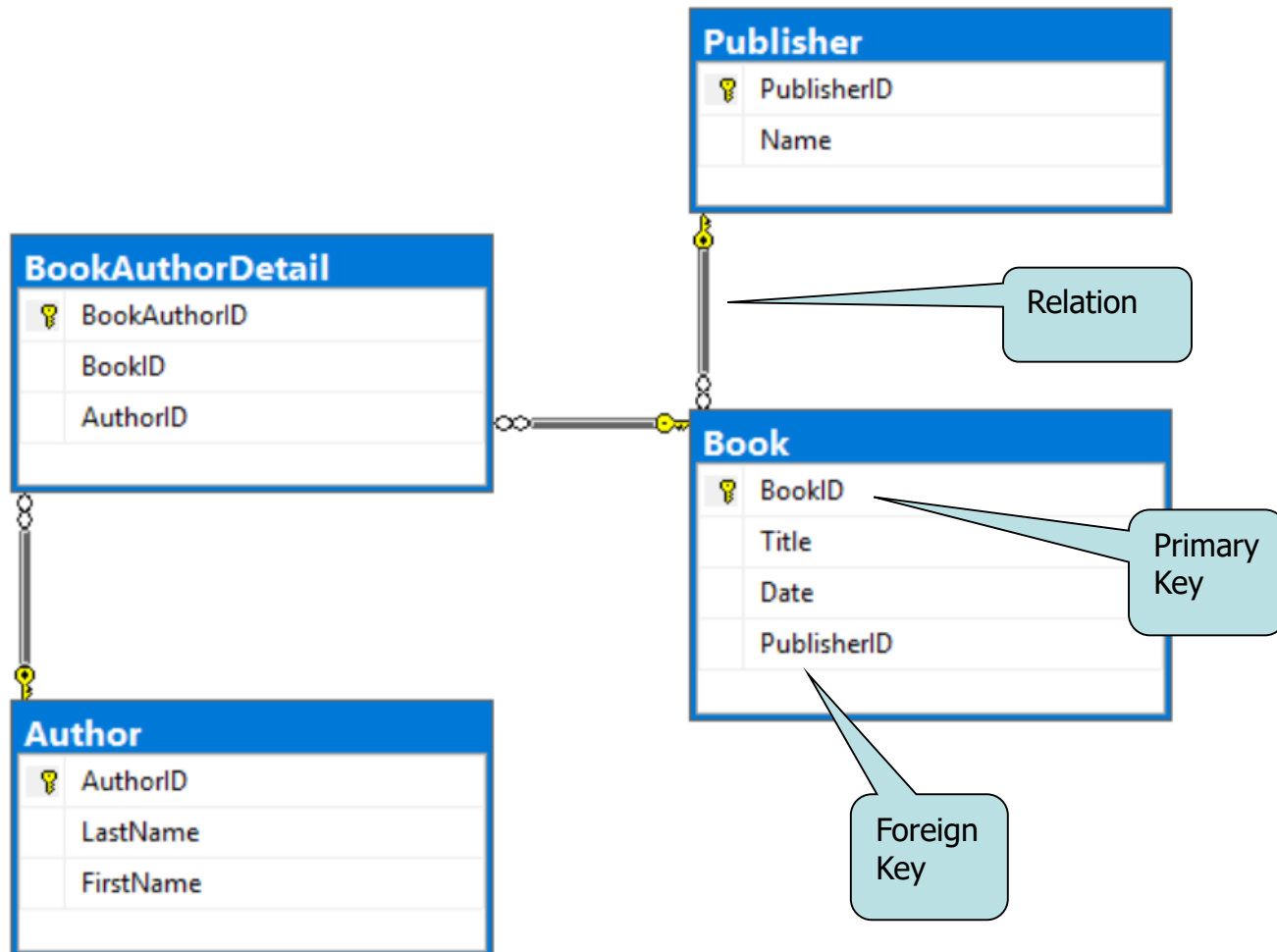
Relationships

- A relationship exists between a book and its author(s). The book and author have a writtenBy relationship.
- Each of the four tables in our Books database has one or more relationships with other tables in the database:

Authors \leftrightarrow AuthorsBooks \leftrightarrow Books \leftrightarrow Publishers

- The AuthorsBooks table establishes a many to many relationship between authors and books. An author may write many books and a book may have more than one author.
- Publishers have a one to many relationship with books. Each book has one publisher, but a publisher publishes many books.

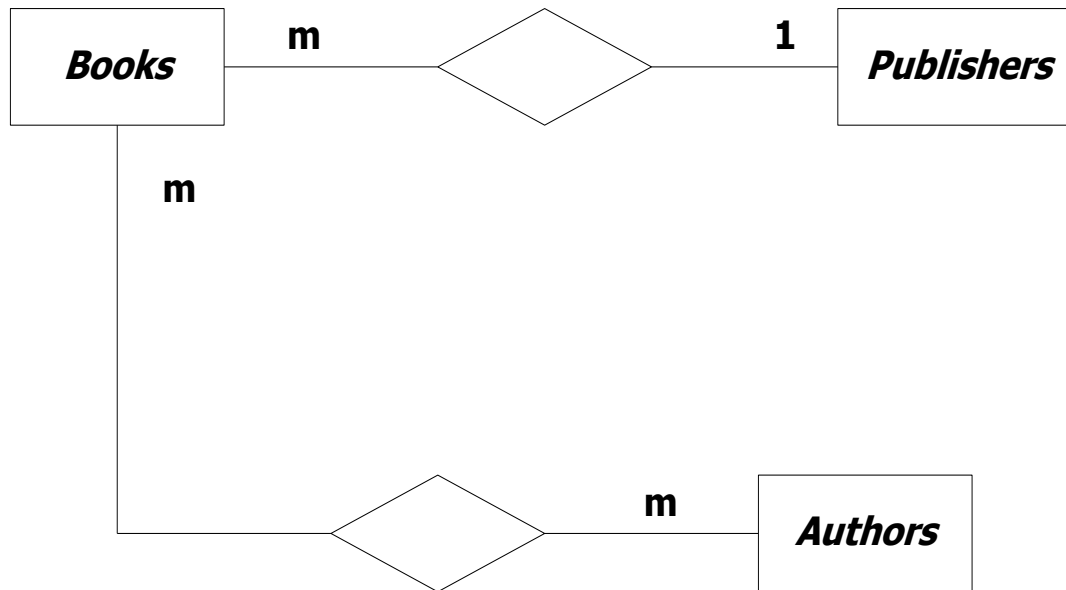
Relationships



Representing Relationships

- A relationship is represented between tables S and T in the database by making the primary key of T an attribute of the S table. We say that the T key is a foreign key in the S table.
- The relationships between tables in the Books database is shown in an entity-relationship diagram on the next page.
 - Compare this diagram with the tables reproduced on the following page.
 - The AuthorsBooks table's only purpose is to gracefully represent the many to many relationship between books and authors.

Books Database



Queries

- A query expression represents a process for building a new (transient) table from data residing in other (permanent) tables.
 - The tables used to represent a databases entities are designed to minimize redundancy.
 - They do not necessarily present data the way a user may wish to view it.
 - Generating user friendly views are one thing queries do.
 - They also may be used to enter new data or modify existing data in the database.

Query Build and Test

The screenshot shows the SQL Server Enterprise Designer interface. At the top, there are tabs for the query, the database, and the design files. The main area displays a diagram of four tables: Publishers, Books, AuthorBookDetails, and Authors. The Publishers table is connected to the Books table, which is connected to the AuthorBookDetails table, which is connected to the Authors table. The query below the diagram is:

```
SELECT Books.Title AS Expr1, Books.Date AS Expr2, Publishers.Name AS Expr3, Authors.LastName AS Expr4
FROM Books INNER JOIN
    AuthorBookDetails ON Books.BookID = AuthorBookDetails.BookID INNER JOIN
    Authors ON AuthorBookDetails.AuthorID = Authors.AuthorID INNER JOIN
    Publishers ON Books.PublisherID = Publishers.PublisherID
```

The result grid shows the following data:

	Expr1	Expr2	Expr3	Expr4
▶	somebook	2010	Wrox	Sells
	anotherbook	2011	Wrox	Fawcett
	anotherbook	2011	Wrox	OfTheApes

At the bottom, there are navigation controls and a status bar indicating "Cell is Read Only."

Structured Query Language (SQL) Basics

- Uses English words keyed to their tasks:
 - **SELECT**: pick records for processing,
SELECT * FROM Employees.LastName
 - **WHERE**: qualify the selection with specific attribute values
SELECT * FROM Employees WHERE Pay > 50000
 - **AND, OR**: combining conjunctions
WHERE Pay > 40000 AND Pay < 60000
 - **IN**: set inclusion, e.g.,
WHERE Employees.FirstName IN ('Joe', 'Sam')
 - **ORDER BY**: sort selected data,
ORDER BY Employees.LastName
- See SQL References on Lecture #7 webpage

SubQueries

- You can create precise selections using subqueries:
 - `SELECT * FROM Orders
WHERE PartNum in
(SELECT PartNum FROM Part WHERE Description LIKE "Road%")`
 - Here, Orders and Part are tables, PartNum and Description are attributes, and we are selecting Descriptions that have substring "Road"

SQL – Manipulating Data

- You modify the data held in a database using:
 - **INSERT**: insert a new record with specified values,
INSERT INTO Log (EmID, Date, Title, Message)
VALUES (1, 6/30/2004, "More Stuff", "blah blah blah")
 - And insert multiple records,
INSERT INTO TempTable
SELECT * FROM Log
 - **UPDATE**: modify existing records
UPDATE Employee
SET Address = 'Some New Street'
WHERE EmID = 3
 - **DELETE**: remove records from database
DELETE FROM Employee
WHERE City = 'Hoboken'

SQL – Built in Functions

- Some of the most useful SQL built in functions are:
 - **COUNT**: count the number of rows satisfying some condition,
`SELECT COUNT(*) FROM Logs WHERE EmID = 1`
 - **SUM**: returns sum of all values in a column,
`SELECT SUM(Cost) FROM Order`
 - **AVG**: returns average value of a column,
`SELECT AVG(Price) From Order`
 - **MAX, MIN**: returns maximum or minimum value in a column,
`SELECT MAX(Price) From Order`
 - **GETDATE**: returns current system date
`INSERT INTO Log (EmID, Date, Title, Message)
VALUES (1, GetDate(), "More Stuff", "blah blah blah")`

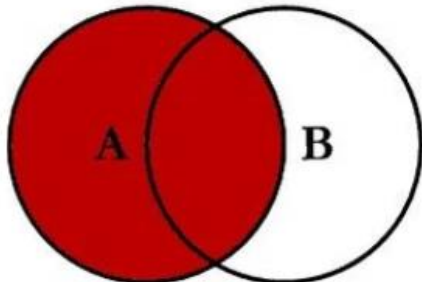
SQL – Altering Database Structure

- Create database, table, query, and alter column type
 - **CREATE DATABASE** CSE686
 - **CREATE TABLE** Log (
 LogID timestamp,
 EmID int,
 Date datetime,
 Title text,
 Message text);
 - **CREATE VIEW** LogAfter_Fawcett
 SELECT Date, Title, Message
 FROM Log WHERE Date < '6/28/2004'
 - **ALTER TABLE** Employees
 -> CHANGE Country Country char(12)

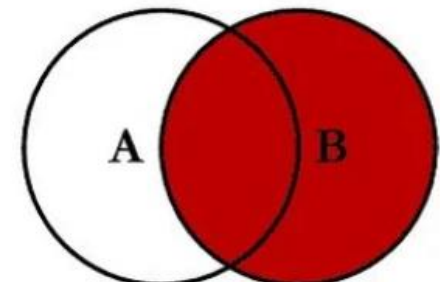
SQL Joins

- Joins are selections from multiple tables. There are several kinds:
 - EQUI JOIN: combine matching records
SELECT * FROM Employees, Log
WHERE Employees.EmID = Log.EmID
 - INNER JOIN:
Employees INNER JOIN Log
ON Employees.EmID = Log.EmID
 - Also:
Left Inner Join, Right Inner Join,
outer Join, Left outer Join, Right outer Join, Full Outer Join

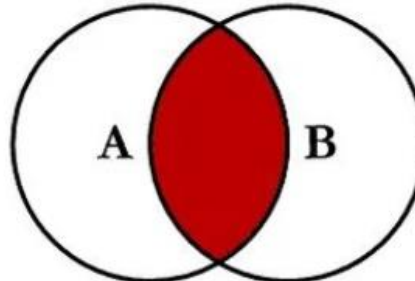
SQL JOINS



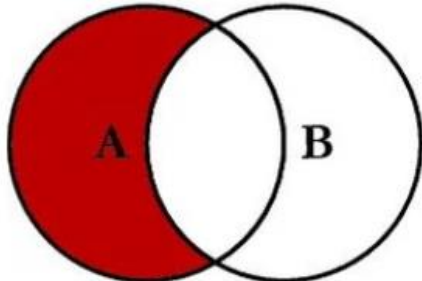
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



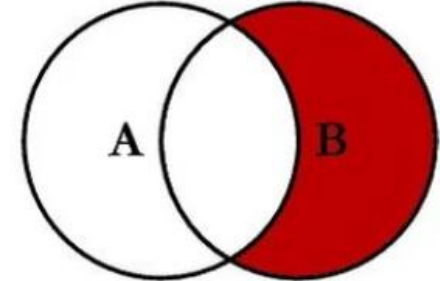
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



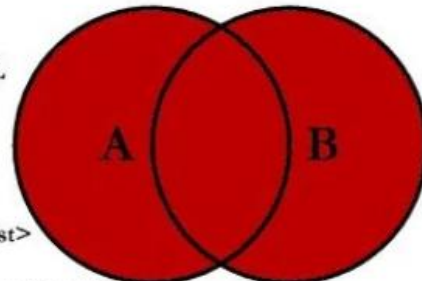
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



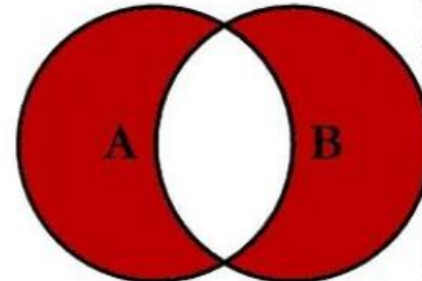
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

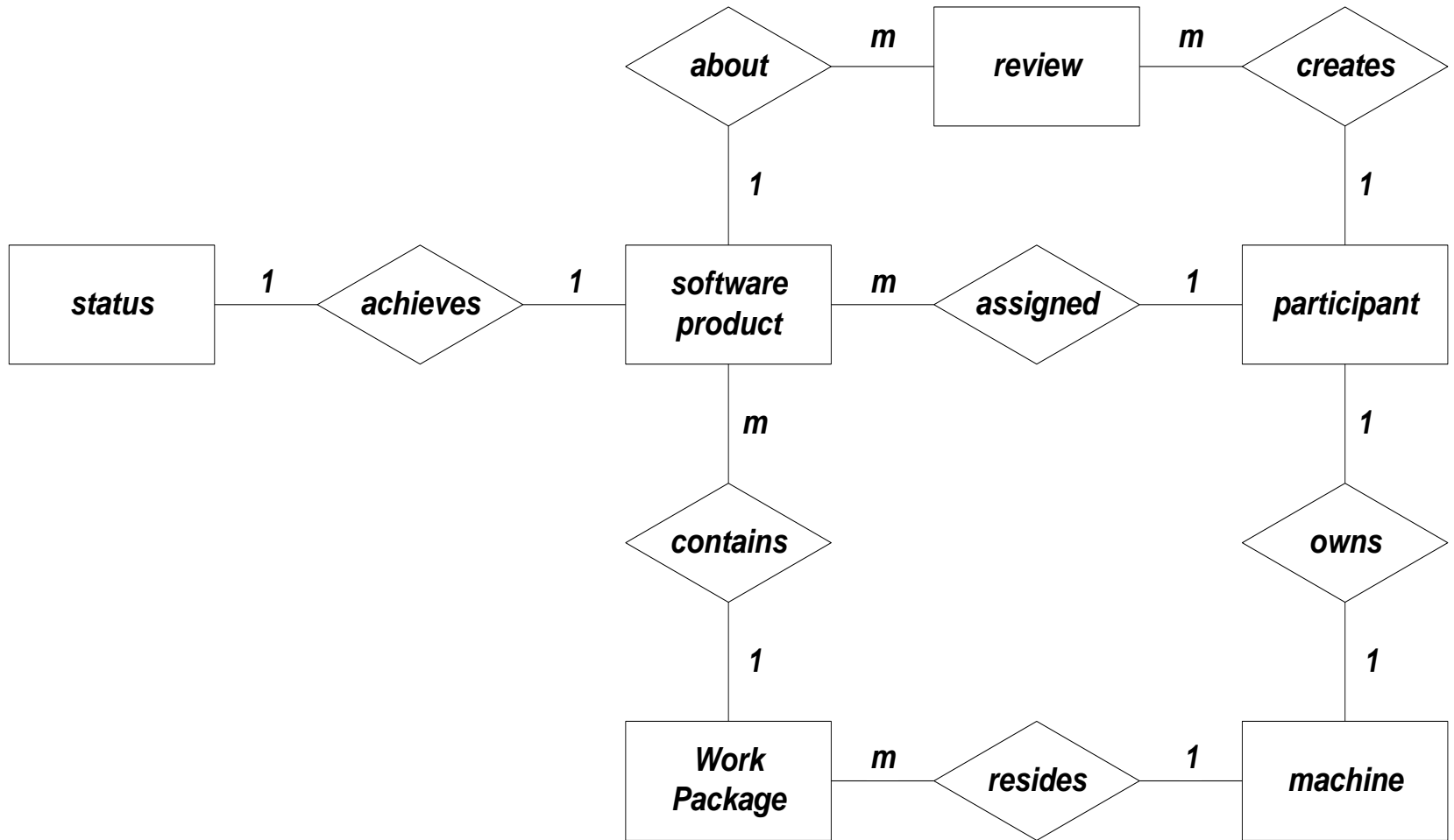


```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Entity Relationship Diagram (ERD)

Final Project – Internet Programming, Summer 2002



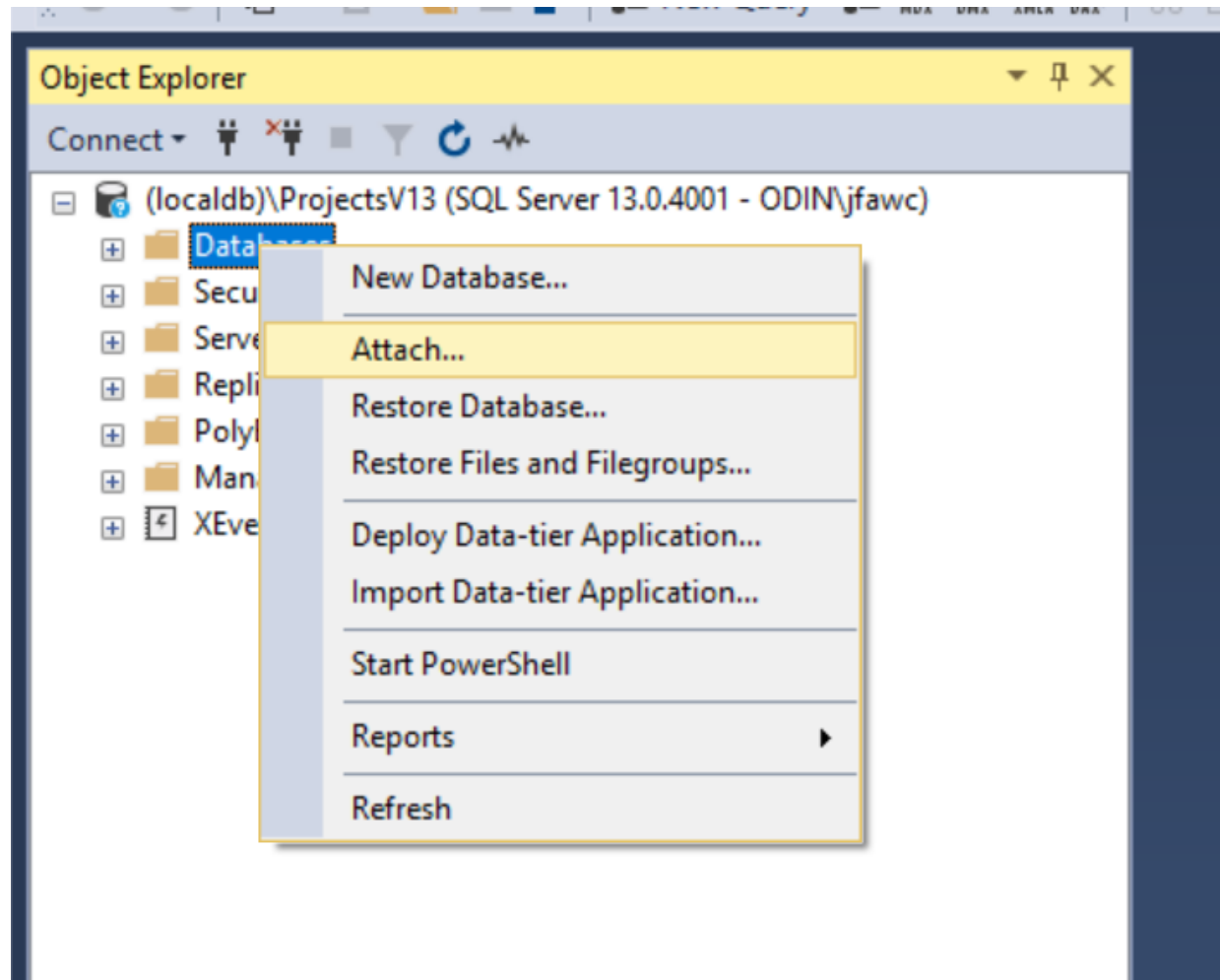
Creating an SQL Server Database

- An SQL Server Database can contain:
 - Multiple tables with relationships between them
 - Structured Query Language (SQL) based queries
 - Can form joins of data from multiple tables
 - Forms which provide views of the database data
 - Extracts data from tables but can hide the table structure from clients
- SQL Server supports a complete relational model with a fairly high performance engine, suitable for concurrent users.
- If you only need to support a few, mostly non-concurrent users you can use Access.

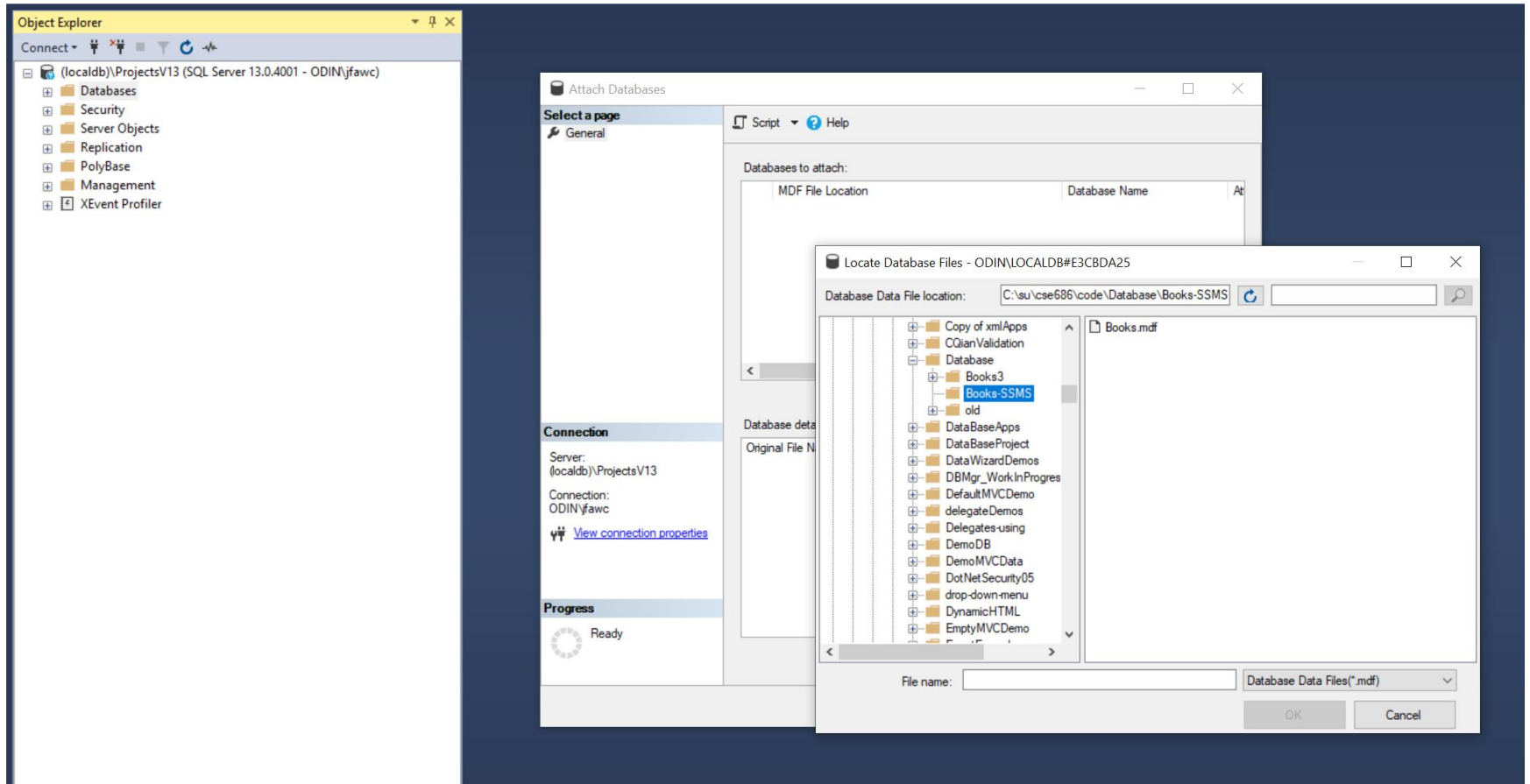
SQL Server Management Studio

- This section provides screen shots for building a multi-table database with relationships.
- Download here:
<https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017>

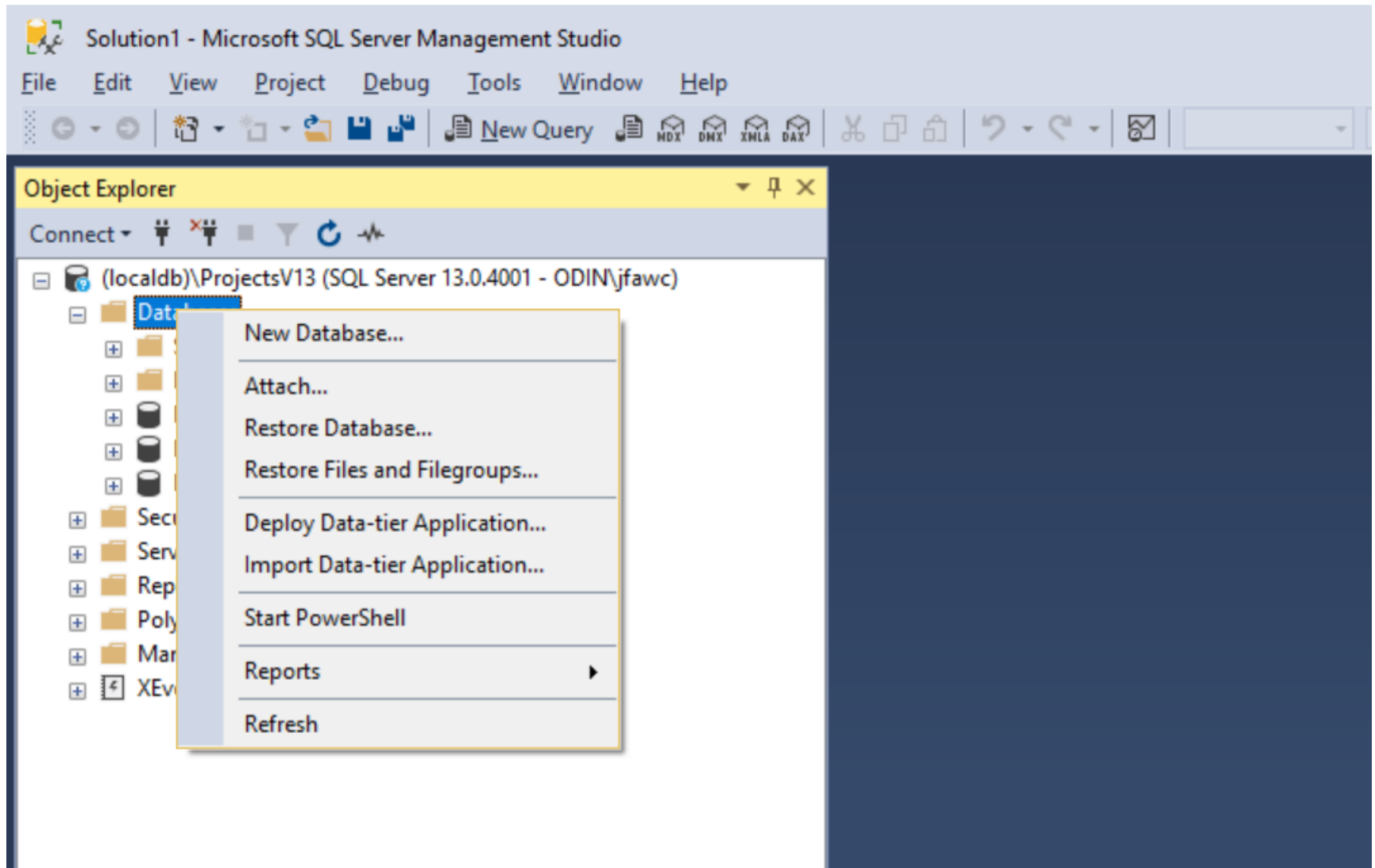
Attach to Existing Database



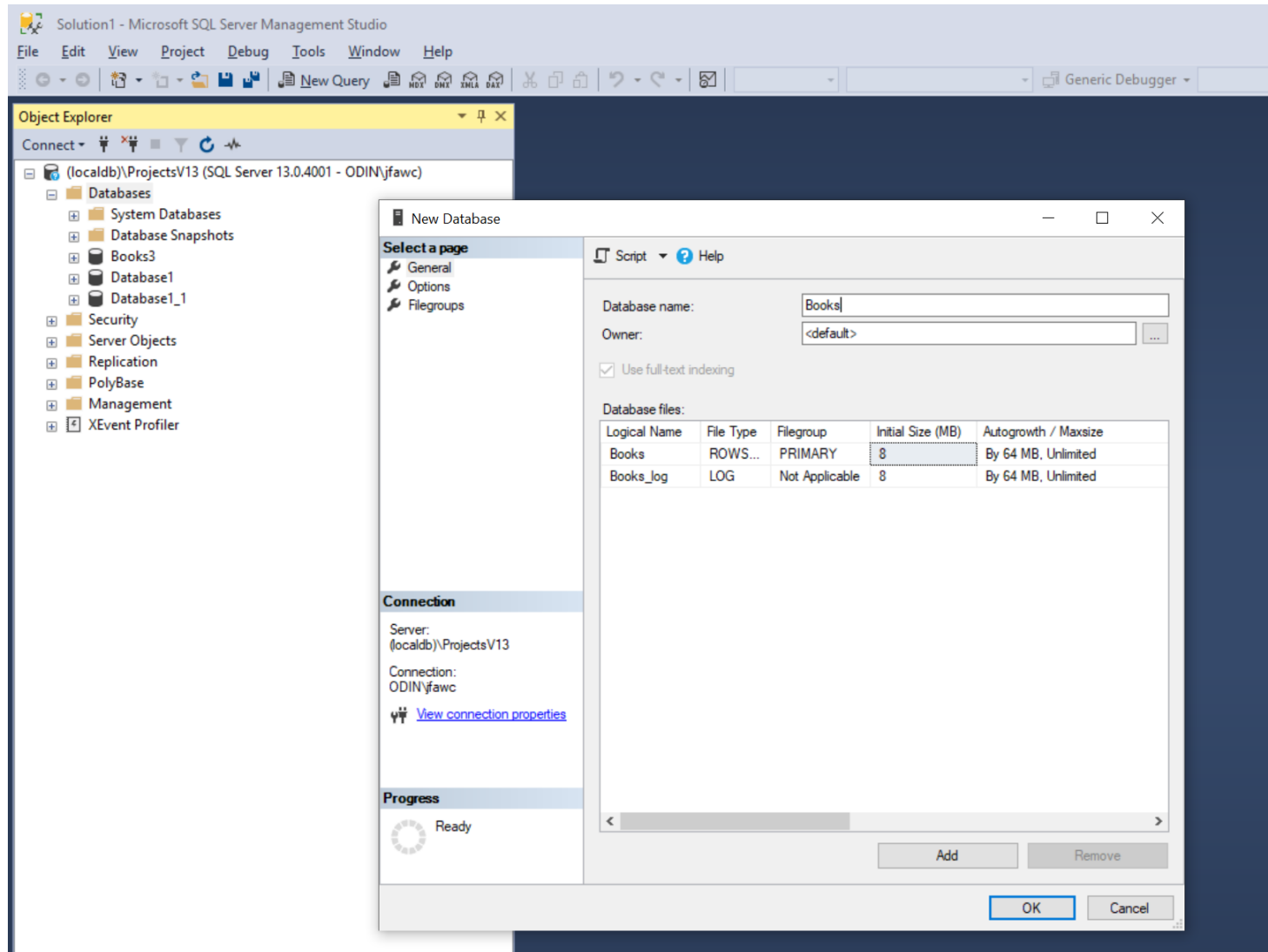
Attach to Existing Database



Creating a SQL Server Database



Add Database Books



Set PublisherID as Primary Key

ODIN\LOCALDB#5E28...s - dbo.Publisher

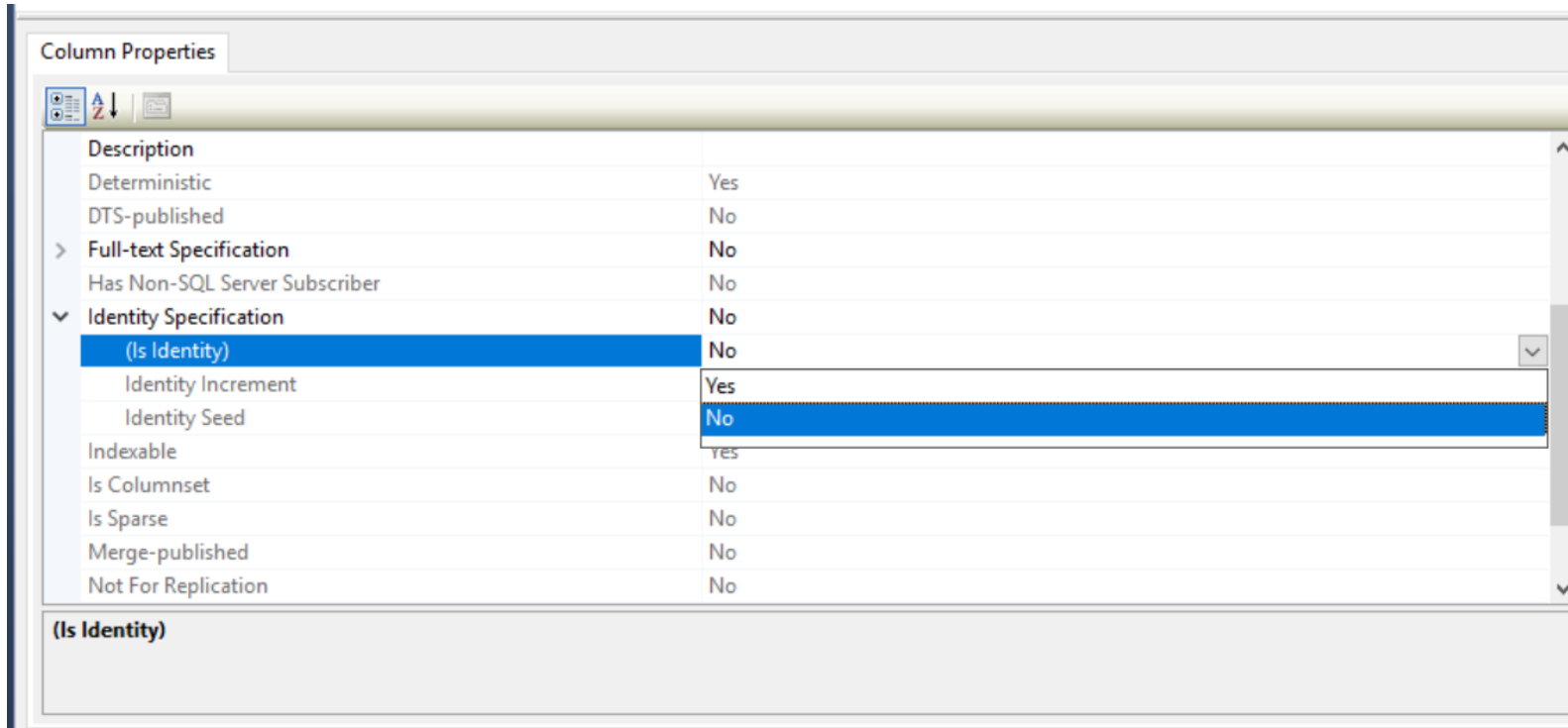
Column Name	Data Type	Allow Nulls
PublisherID	int	<input type="checkbox"/>
Name	nvarchar(50)	<input type="checkbox"/>
		<input type="checkbox"/>

ODIN\LOCALDB#5E28...s - dbo.Publisher

Column Name	Data Type	Allow Nulls
PublisherID	int	<input type="checkbox"/>
Name	nvarchar(50)	<input type="checkbox"/>
		<input type="checkbox"/>

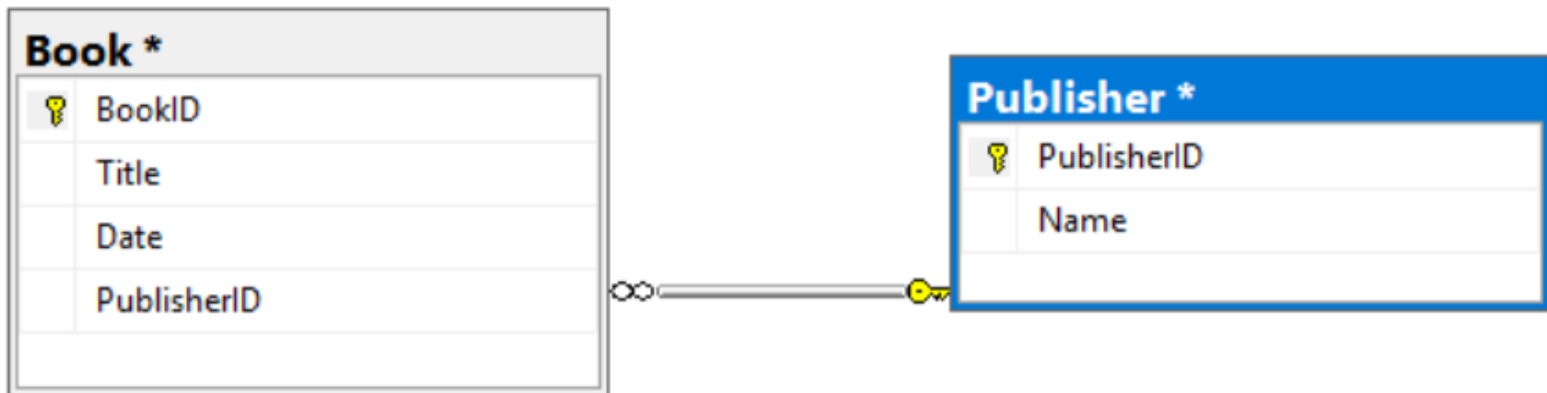
- Set Primary Key
- Insert Column
- Delete Column
- Relationships...
- Indexes/Keys...
- Fulltext Index...
- XML Indexes...
- Check Constraints...
- Spatial Indexes...
- Generate Change Script...
- Properties Alt+Enter

Set Key as Identity (auto generated)



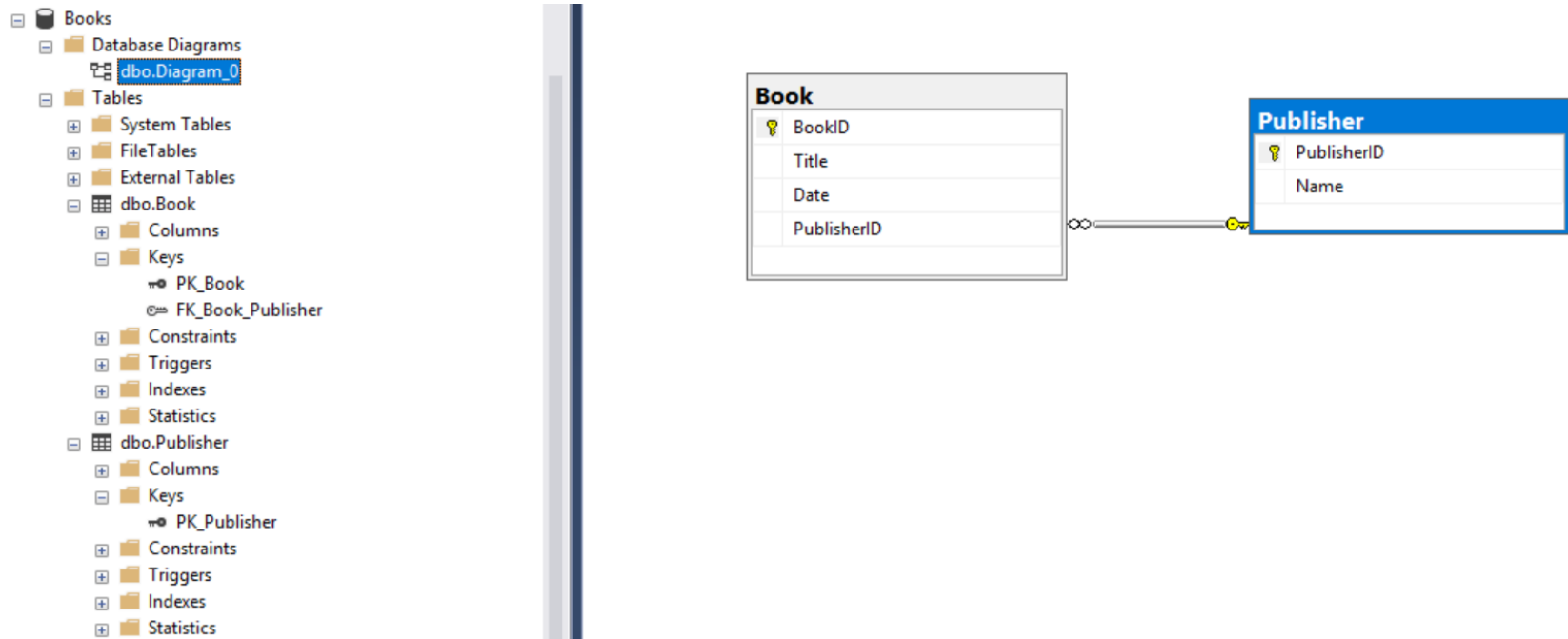
Add Relationship

- After creating two tables with appropriate keys, we can create a relationship between them.
- Add a diagram, then drag the PublisherID key to Book table.



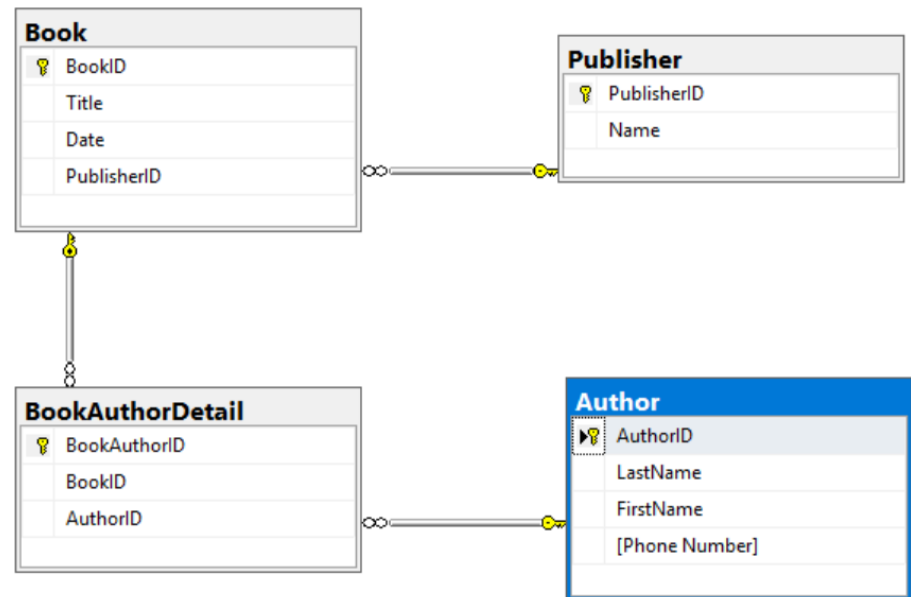
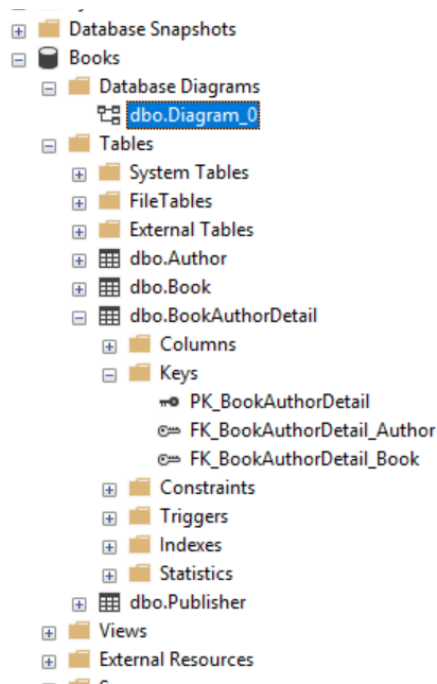
Resulting Foreign Key in Book Table

- We now have a Foreign key in the Book table



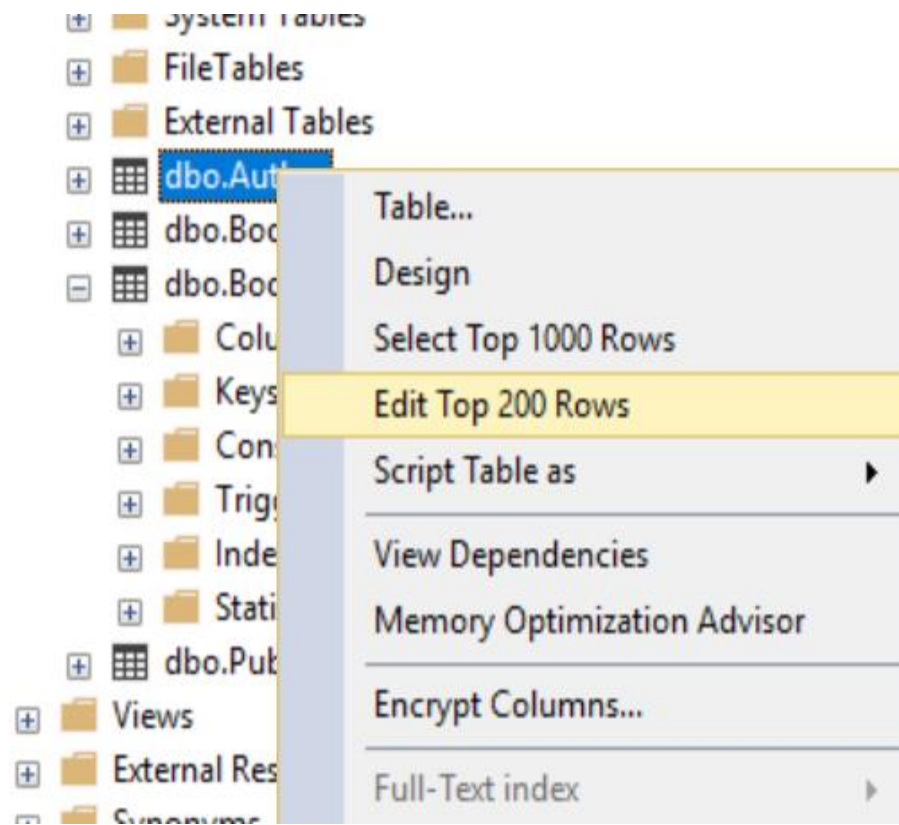
Many to Many Relationships

- For many to many relationships we need a Details table with two foreign keys



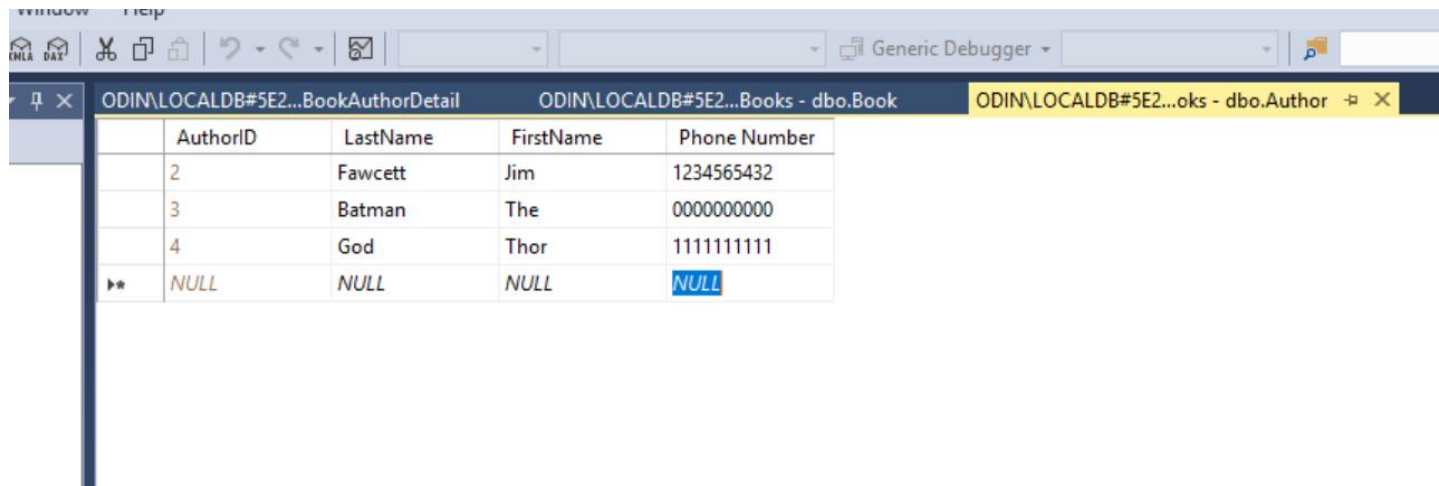
Adding Data to Table

- Right-click on table and select Edit



Adding Data

- Adding data to the Author table

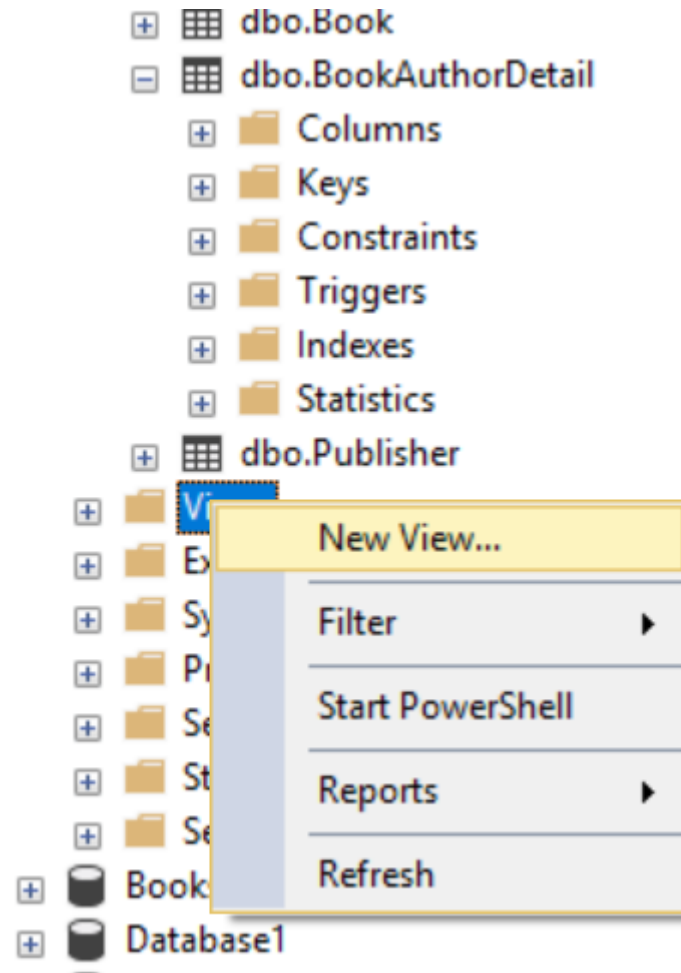


The screenshot shows a SQL Server Enterprise Manager window with three tabs: 'ODIN\LOCALDB#5E2...BookAuthorDetail', 'ODIN\LOCALDB#5E2...Books - dbo.Book', and 'ODIN\LOCALDB#5E2...oks - dbo.Author'. The 'dbo.Author' tab is active and displays a table with the following data:

AuthorID	LastName	FirstName	Phone Number
2	Fawcett	Jim	1234565432
3	Batman	The	0000000000
4	God	Thor	1111111111
▶*	NULL	NULL	NULL

Views

- Adding view to Books Database



Specifying View

The screenshot shows the SQL Server Enterprise Manager interface. At the top, there are four table views: Author, Book, BookAuthorDetail, and Publisher. Lines connect them to show relationships. Below the table views is a table with columns: Column, Alias, Table, Outp..., Sort Type, Sort Order, Filter, Or..., Or..., Or... The table contains the following data:

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
FirstName		Author	<input checked="" type="checkbox"/>						
LastName		Author	<input checked="" type="checkbox"/>						
Title		Book	<input checked="" type="checkbox"/>						
Date		Book	<input checked="" type="checkbox"/>						
PublisherID		Publisher	<input checked="" type="checkbox"/>						
			<input type="checkbox"/>						

Below the table is the SQL query:

```
SELECT dbo.Author.FirstName, dbo.Author.LastName, dbo.Book.Title, dbo.Book.Date, dbo.Publisher.PublisherID
FROM   dbo.Author INNER JOIN
       dbo.BookAuthorDetail ON dbo.Author.AuthorID = dbo.BookAuthorDetail.AuthorID INNER JOIN
       dbo.Book ON dbo.BookAuthorDetail.BookID = dbo.Book.BookID INNER JOIN
       dbo.Publisher ON dbo.Book.PublisherID = dbo.Publisher.PublisherID
```

Executing View Code

The screenshot shows a SQL Server Enterprise Manager interface with four tables: Author, Book, BookAuthorDetail, and Publisher. The Author table has columns: AuthorID, LastName, FirstName, and Phone Number. The Book table has columns: BookID, Title, Date, and PublisherID. The BookAuthorDetail table has columns: BookAuthorID, BookID, and AuthorID. The Publisher table has columns: PublisherID and Name. A context menu is open over the table grid, with 'Execute SQL' selected. The SQL query in the background is:

```
SELECT  
FROM  
    dbo.Author.FirstName, dbo.Author.LastName, c  
    dbo.Author INNER JOIN  
    dbo.BookAuthorDetail ON dbo.Author.Autho  
    dbo.Book ON dbo.BookAuthorDetail.BookID :  
    dbo.Publisher ON dbo.Book.PublisherID = db
```

View Results

ODIN\LOCALDB#5E2...oks - dbo.View_2* | SQLQuery1.sql - (lo...s (ODIN)\fawc (53)) | ODIN\LOCALDB#5E2...oks - dbo.View_1

The diagram shows four tables: Author, Book, BookAuthorDetail, and Publisher. Author is linked to BookAuthorDetail via AuthorID. Book is linked to BookAuthorDetail via BookID. Book is linked to Publisher via PublisherID.

Column	Alias	Table	Outp...	Sort Type	Sort Order	Filter	Or...	Or...	Or...
Date		Book	<input checked="" type="checkbox"/>						
Name		Publisher	<input checked="" type="checkbox"/>						
			<input type="checkbox"/>						
			<input type="checkbox"/>						
			<input type="checkbox"/>						

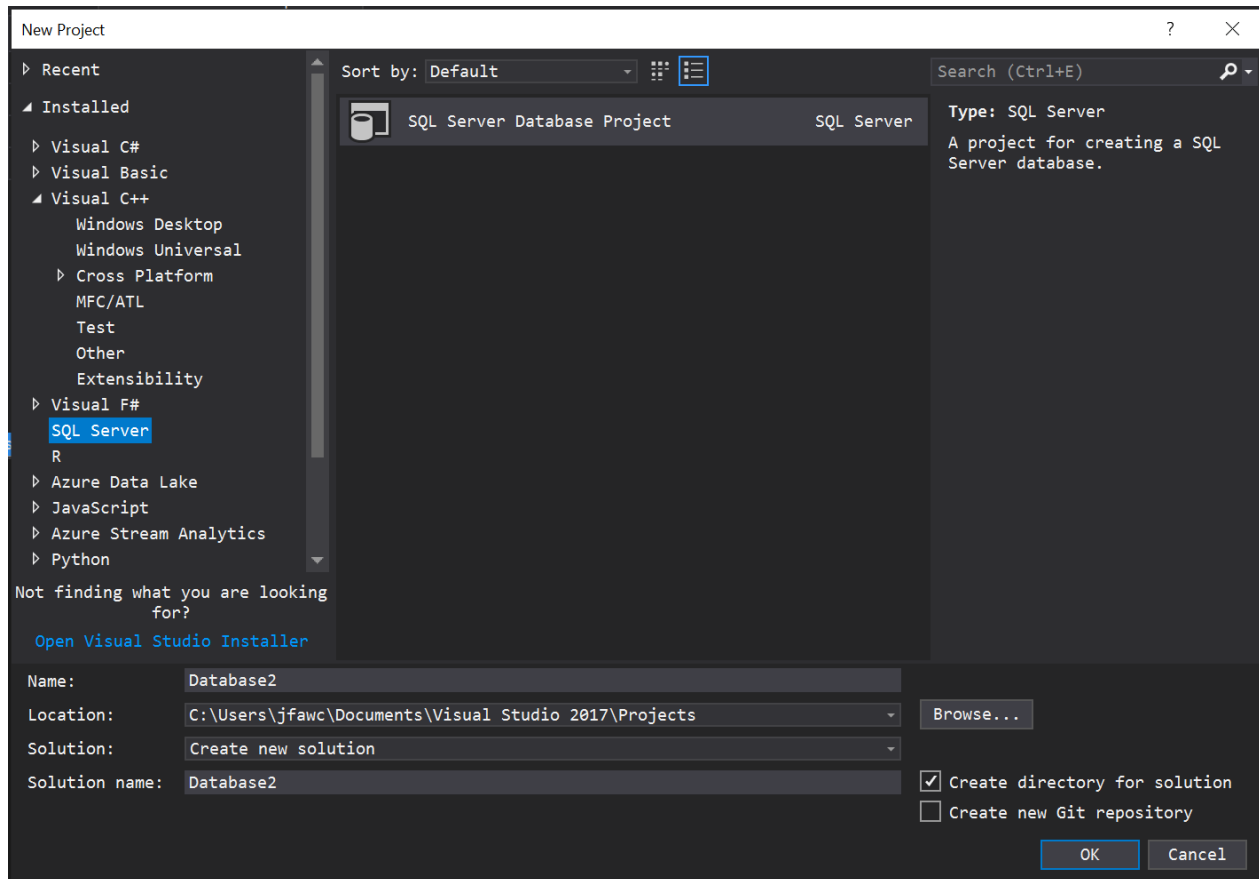
```

SELECT dbo.Author.FirstName, dbo.Author.LastName, dbo.Book.Title, dbo.Book.Date, dbo.Publisher.Name
FROM   dbo.Author INNER JOIN
       dbo.BookAuthorDetail ON dbo.Author.AuthorID = dbo.BookAuthorDetail.AuthorID INNER JOIN
       dbo.Book ON dbo.BookAuthorDetail.BookID = dbo.Book.BookID INNER JOIN
       dbo.Publisher ON dbo.Book.PublisherID = dbo.Publisher.PublisherID
    
```

	FirstName	LastName	Title	Date	Name
▶	Jim	Fawcett	The Good, The Bad, The Ugly	1960-01-01	BadPress
	The	Batman	The Good, The Bad, The Ugly	1960-01-01	BadPress
	Jim	Fawcett	I Love C++	2019-02-25	BoringPress

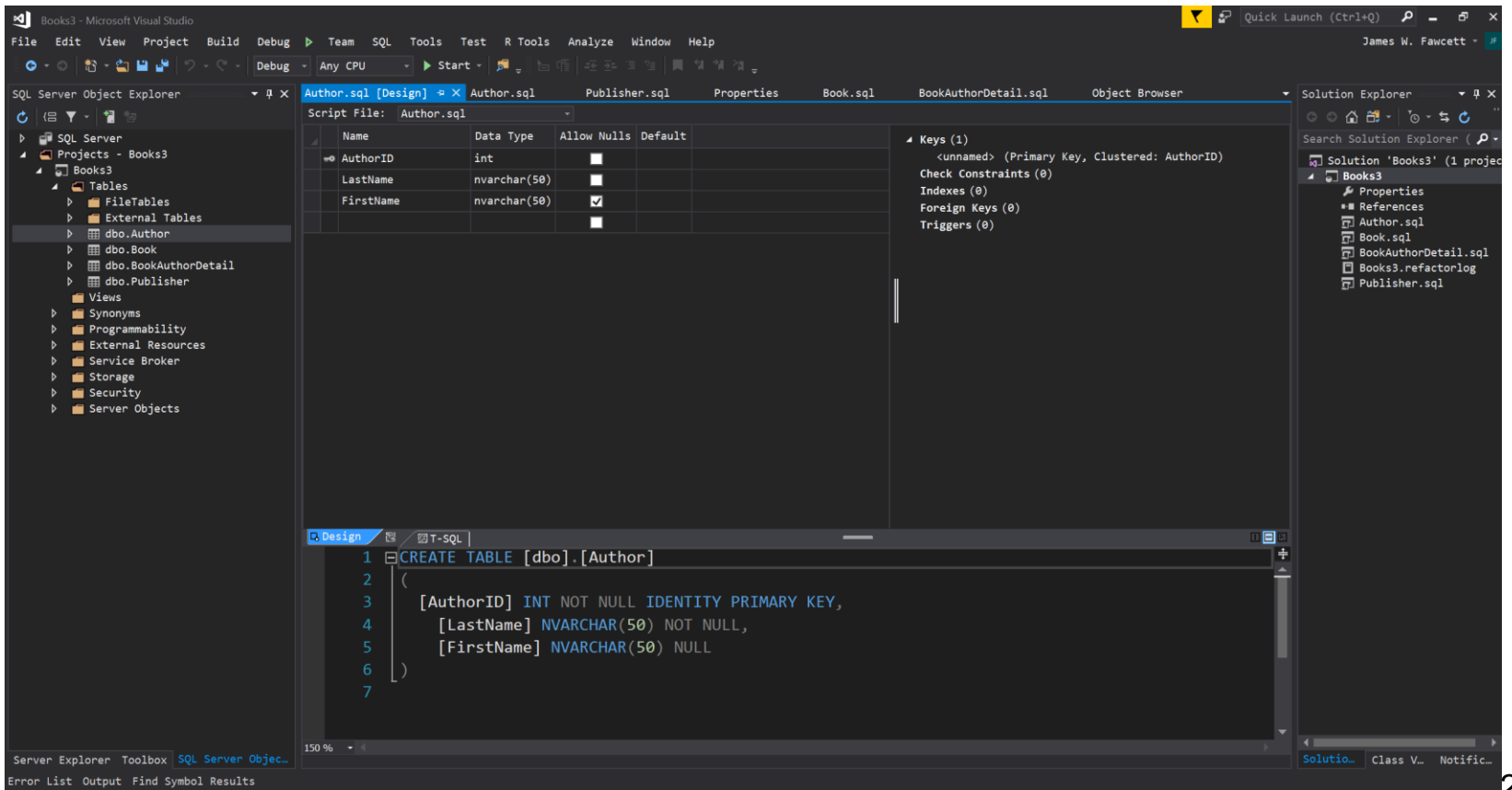
Creating DataBase Project in VS

- The following slides provide screen shots for building a database in Visual Studio 2017



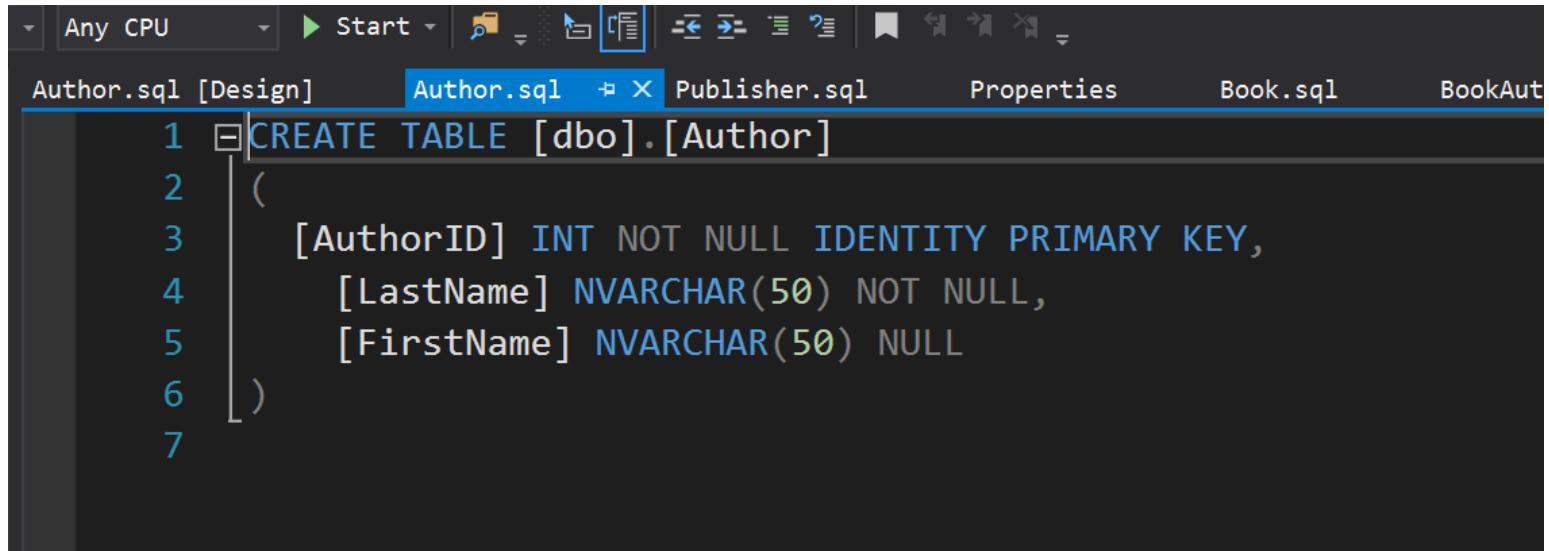
Database Project in Visual Studio

- File > New Project > SQL Server
- Add Tables with Tables context menu



Adding Table

- Adding Table puts you in Design View where you add columns.
- That builds CREATE TABLE script.
- You need to add properties like IDENTITY



The screenshot shows the SQL Server Enterprise Manager interface. The 'Author.sql' tab is active, displaying the following SQL script:

```
1 CREATE TABLE [dbo].[Author]
2 (
3     [AuthorID] INT NOT NULL IDENTITY PRIMARY KEY,
4     [LastName] NVARCHAR(50) NOT NULL,
5     [FirstName] NVARCHAR(50) NULL
6 )
7
```

Relationships

- Add attributes to the foreign key column

The screenshot displays the SQL Server Enterprise Designer interface. The top pane shows the 'Design' view of the 'Book' table. The columns are:

Name	Data Type	Allow Nulls	Default
BookID	int	<input type="checkbox"/>	
Title	nvarchar(50)	<input type="checkbox"/>	
Date	date	<input type="checkbox"/>	
PublisherID	int	<input type="checkbox"/>	

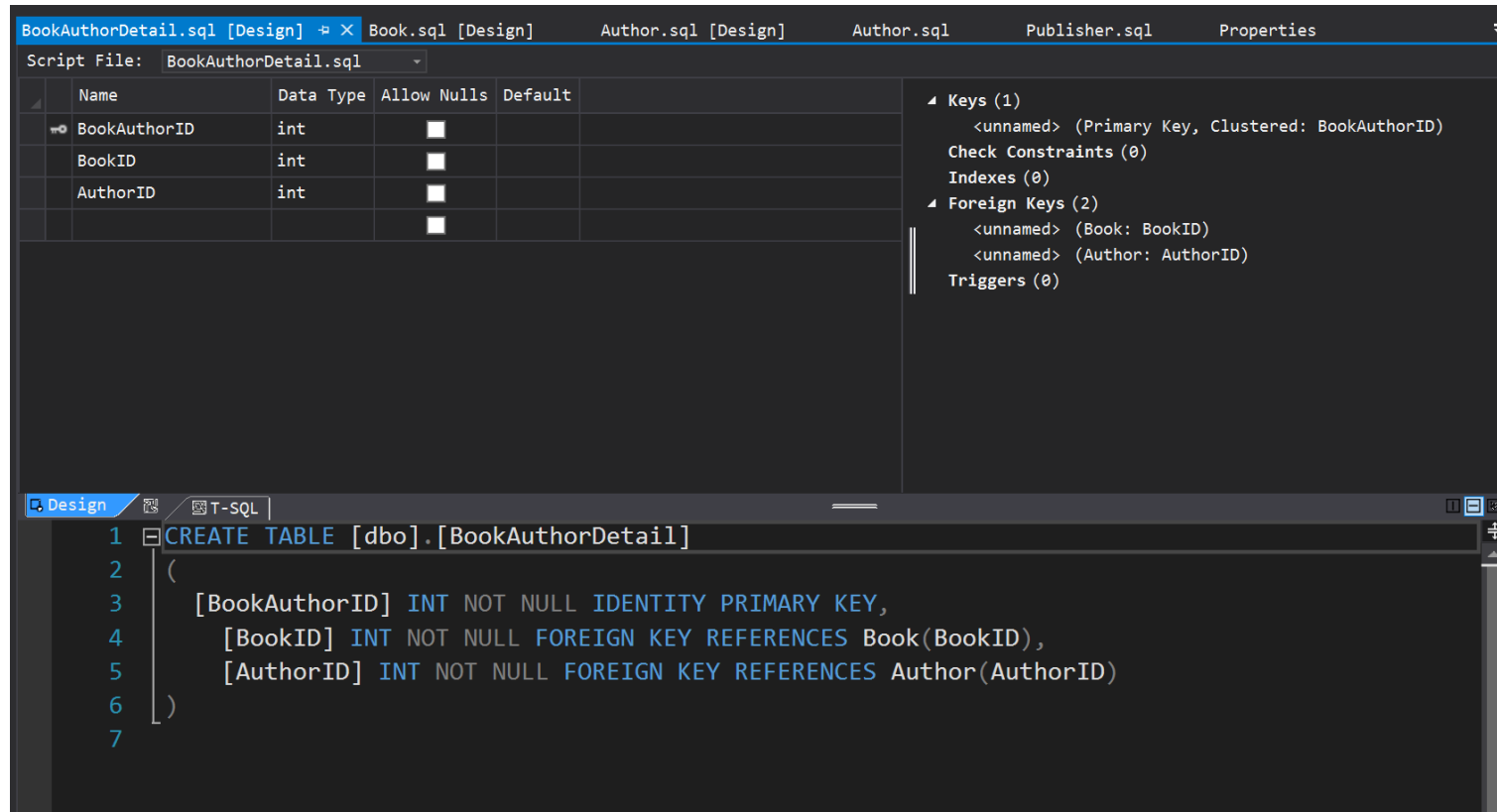
The right pane shows the 'Properties' window for the table, listing constraints: Keys (1), Check Constraints (0), Indexes (0), Foreign Keys (1), and Triggers (0).

The bottom pane shows the T-SQL script for creating the table:

```
1 CREATE TABLE [dbo].[Book]
2 (
3     [BookID] INT NOT NULL IDENTITY PRIMARY KEY,
4     [Title] NVARCHAR(50) NOT NULL,
5     [Date] DATE NOT NULL,
6     [PublisherID] INT NOT NULL FOREIGN KEY REFERENCES Publisher(PublisherID)
7 )
8
```

Many To Many Relationships

- For many to many you simply provide two foreign keys and decorate their element scripts, as shown.



The screenshot displays the SQL Server Enterprise Designer interface. The top pane shows the 'Design' view of the 'BookAuthorDetail' table. The table has three columns: 'BookAuthorID' (int, primary key), 'BookID' (int, foreign key), and 'AuthorID' (int, foreign key). The bottom pane shows the T-SQL script for the table creation.

Name	Data Type	Allow Nulls	Default
BookAuthorID	int	<input type="checkbox"/>	
BookID	int	<input type="checkbox"/>	
AuthorID	int	<input type="checkbox"/>	

```
1 CREATE TABLE [dbo].[BookAuthorDetail]
2 (
3     [BookAuthorID] INT NOT NULL IDENTITY PRIMARY KEY,
4     [BookID] INT NOT NULL FOREIGN KEY REFERENCES Book(BookID),
5     [AuthorID] INT NOT NULL FOREIGN KEY REFERENCES Author(AuthorID)
6 )
7
```

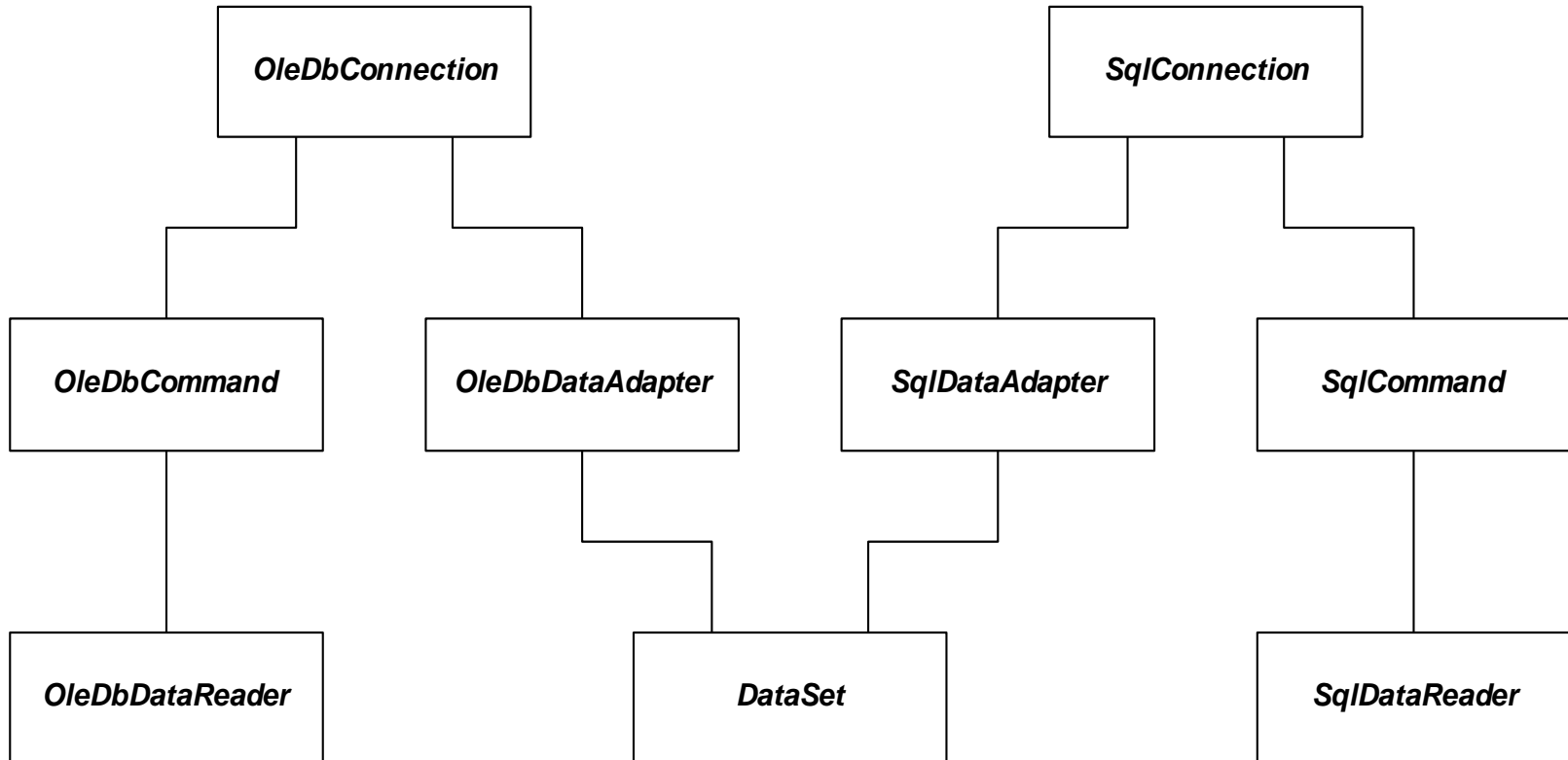
ADO.Net

1. Low-level support for DB management
2. Used by LINQ to SQL and Entity Framework

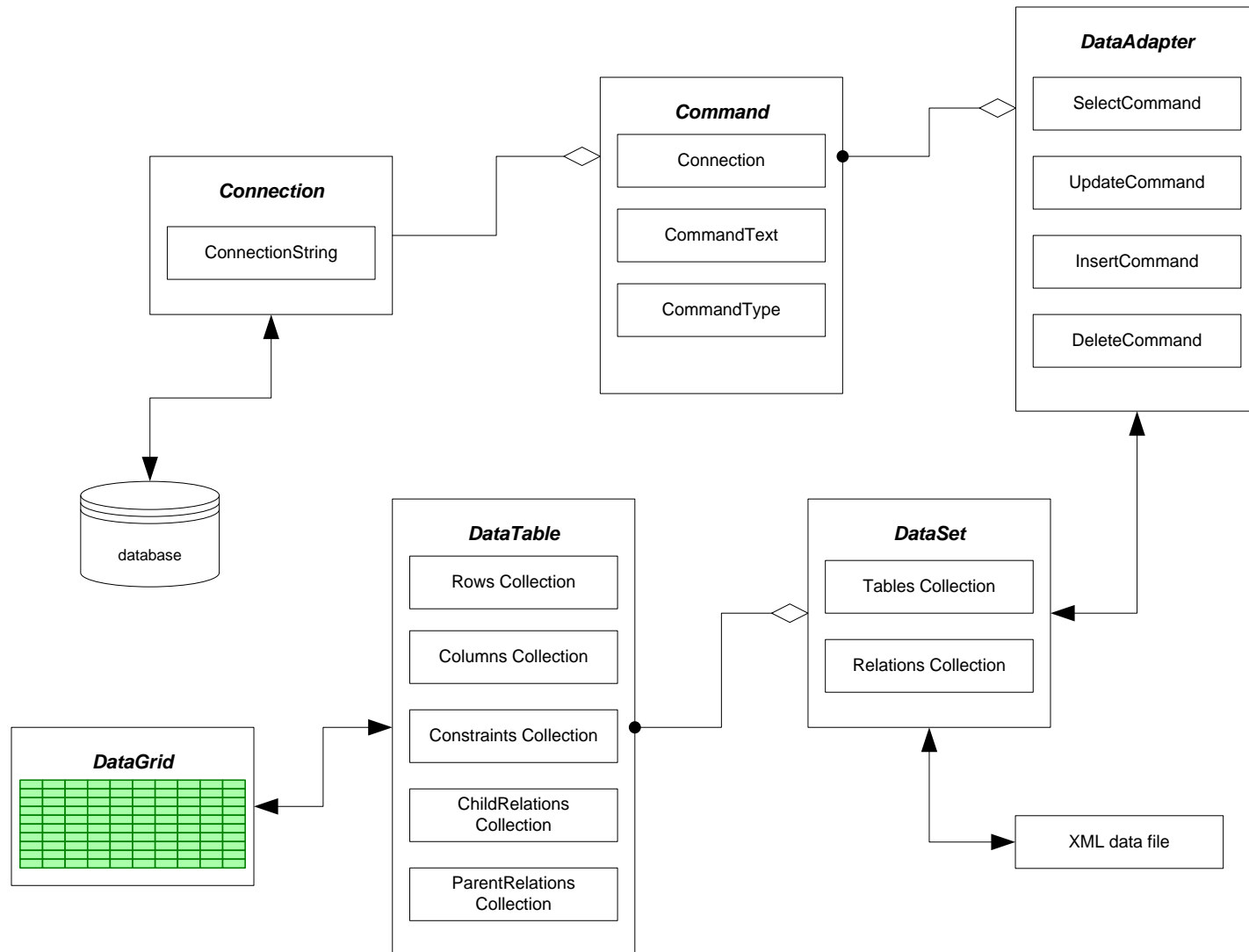
Support for Data in .Net

- Connected data access:
 - Use Connection object and Command to connect a DataReader object to database and read iteratively.
 - Use Connection object and Command to connect a DataReader and execute an SQL statement or stored procedure.
- Disconnected data access:
 - Use a Connection and Command to connect a DataAdapter to the database and fill a DataSet with the results.
 - Use a Connection and Command to connect a DataAdaptor to the database and then call Update method of the DataSet.

Data Provider Classes



ADO Objects



Connection Object

- Methods
 - Open()
 - Close()
 - BeginTransaction()
- Properties
 - ConnectionString

Command Object

- Used to connect Connection Object to DataReader or a DataAdapter object
- Methods
 - ExecuteNonQuery()
 - Executes command defined in CommandText property, e.g., UPDATE, DELETE, INSERT
 - ExecuteReader(CommandBehavior)
 - Returns a reader attached to the resulting rowset
 - ExecuteScalar()
- Properties
 - Connection
 - CommandText
 - CommandType

Data Adapter Object

- Used to:
 - extract data from data source and populate tables in a DataSet
 - Push changes in DataSet back to source
- Methods
 - Fill(DataSet, Table)
 - FillSchema(DataSet, SchemaType)
 - Update()
- Properties
 - SelectCommand
 - UpdateCommand
 - InsertCommand
 - DeleteCommand

DataSet Object

- Used for Disconnected manipulation of a source's data.
- Methods
 - Clear()
 - ReadXML(XmlReader)
 - WriteXML(XmlWriter)
 - AcceptChanges()
 - HasChanges()
 - AbandonChanges()
- Properties
 - Tables collection
 - `ds.Tables[tableStr].Rows[3]["Responsible Individual"] = userID;`
 - Relations collection

DataReader Object

- Supports one-way, forward-only, access to data
- Methods
 - Read()
 - Advances current row pointer
 - GetBoolean, GetInt16, GetChars, GetString, GetValue
 - Close()
- Properties
 - this[string]
 - this[int]

References

- Programming Microsoft .Net, Jeff Prosise, Microsoft Press, 2002
- Access Database Design & Programming, Steven Roman, O'Reilly, 2002
- Professional C#, Robinson et. al., Wrox Press, 2002
- www.w3schools.com/sql/default.asp