



Introduction to SQL

Sample manual - first two chapters



WiseOwl
Training

TABLE OF CONTENTS (1 of 5)

1	DESIGNING DATABASES	Page
1.1	The Four Stages of Database Design	7
	<i>Stage 1 – Deciding what to Include</i>	7
	<i>Stage 2 – Dividing Data into Tables</i>	8
	<i>Stage 3 – Choosing a Primary Key for each Table</i>	9
	<i>Stage 4 – Creating Relationships and a Database Diagram</i>	10
1.2	Many-to-Many Relationships	11

2	SQL SERVER MANAGEMENT STUDIO	Page
2.1	Starting to Use Management Studio	12
2.2	Object Explorer	13
	<i>Useful Start-up Options</i>	13

3	CREATING A DATABASE AND TABLES	Page
3.1	Creating the Database	14
	<i>The Files Created</i>	14
3.2	Creating Tables	15
3.3	Setting an Identity Primary Key	16
3.4	Creating Columns	17
	<i>Data Types Explained in this Chapter</i>	17
	<i>Other Data Types in SQL Server</i>	17
3.5	Whole Numbers	18
	<i>Integer Field Types</i>	18
	<i>Logical Field Types</i>	18
3.6	Other Numerical Fields	19
	<i>Decimal and Numeric Field Types</i>	19
	<i>Float and Real Data Types</i>	19
3.7	Character Data Types	20
	<i>Types of Character Storage</i>	20
	<i>Variable Length Data Types</i>	20
	<i>Fixed Length Data Types</i>	20
3.8	Date/Time Data Types	21
	<i>Entering Dates into a Table</i>	21
3.9	Default and Null Values	22
	<i>Allowing Nulls</i>	22
	<i>Default Values</i>	22
3.10	Database Diagrams	23
	<i>Creating a Database Diagram</i>	23
	<i>Creating Relationships</i>	24
	<i>Database Diagram Support Objects Error</i>	24

4	QUERIES	Page
4.1	Basic SELECT Statements	25
	<i>Where to Put your Commas</i>	25
4.2	Creating Queries	26
	<i>Starting a New Query</i>	26
	<i>Choosing the Right Database</i>	26
4.3	Running Queries	27
	<i>Parsing a Query</i>	27
	<i>Executing a Query</i>	27
	<i>Viewing Information on a Query's Execution</i>	28
	<i>Cancelling a Running Query</i>	28
	<i>Redirecting Query Output</i>	28
4.4	Dealing with Errors	29
	<i>Displaying Line Numbers</i>	29
4.5	Using IntelliSense	30
	<i>Refreshing IntelliSense</i>	30
4.6	Multiple SQL Commands	31
4.7	Saving, Opening and Closing Queries	32
	<i>Saving Queries</i>	32
	<i>Opening Queries</i>	32
	<i>Closing Queries</i>	33

5	LAYING OUT QUERIES	Page
5.1	Using Case	34
5.2	Indentation and Word Wrap	35
	<i>Changing Tab Settings</i>	35
	<i>Word Wrap</i>	35
5.3	Comments	36
	<i>Commenting Out Blocks of Code</i>	36
5.4	Colours in SQL	37
	<i>Changing the Default Colours</i>	37
5.5	Auto-formatting SQL	38

TABLE OF CONTENTS (2 of 5)

6	THE SELECT STATEMENT	Page
6.1	SELECT Statement Syntax	39
	<i>Mnemonic for Order of Commands</i>	39
6.2	Qualified Tables and Columns	40
	<i>Dragging Tables/Columns onto a Query</i>	40
	<i>Specifying the DBO Schema and Database</i>	40
6.3	Table Aliases	40
	<i>Reason 1 for Aliases – Easier to Refer to Field Names</i>	41
	<i>Reason 2 for Aliases – Joins</i>	42
	<i>Changing a Table Alias</i>	42
6.4	Column Aliases	43
	<i>Basic Column Aliases</i>	43
	<i>Other Ways to Create Column Aliases</i>	43
	<i>Aliases in WHERE and ORDER BY Clauses</i>	44
6.5	Ordering Rows	45
	<i>Simple Sorting</i>	45
	<i>Sorting by Multiple Columns</i>	45
6.6	Miscellaneous SELECT Tricks	46
	<i>Selecting All Columns Using *</i>	46
	<i>Selecting Unique Rows</i>	46
	<i>Showing Top and Bottom Rows</i>	47
	<i>Including Ties</i>	47
6.7	Using UNION to Combine Results	48

7	QUERY DESIGNER	Page
7.1	Starting Query Designer	49
	<i>What Query Designer is and does</i>	49
	<i>Starting Query Designer</i>	49
7.2	Using Query Designer	50
	<i>Choosing Tables</i>	50
	<i>Adding/Removing Tables</i>	50
	<i>The Parts of Query Designer</i>	51
	<i>The Non-Existent Results Pane/Execute Button</i>	51
	<i>Working with Columns</i>	52
	<i>Finishing Work in Query Designer</i>	52
7.3	Editing Generated SQL	53
7.4	Advanced Features	54
	<i>Inner and Outer Joins</i>	54
	<i>Grouping</i>	54

8	CRITERIA USING WHERE	Page
8.1	The WHERE Clause	55
	<i>Relational Operators</i>	55
8.2	Criteria with Numbers	56
	<i>Using Comparisons</i>	56
	<i>Finding Numbers in a Given Range</i>	56
8.3	Criteria using Text	57
	<i>Exact Matches</i>	57
	<i>Wildcard Matches using LIKE</i>	57
	<i>Using Special Characters with LIKE</i>	58
	<i>Ranges and Wildcards</i>	58
	<i>Using Relational Operators with Text</i>	59
	<i>Case Sensitivity</i>	59
8.4	Criteria for Dates	60
	<i>Using Dates in Criteria</i>	60
	<i>Using Dates with Wildcards</i>	60
8.5	Combining Criteria	61
8.6	Nulls	62
	<i>An Example of Testing for Nulls</i>	62
	<i>Entering Nulls into a Table</i>	62

9	EXPORTING TO EXCEL	Page
9.1	Copying and Pasting	63
	<i>Copying Column Headers by Default</i>	63
9.2	Exporting Data	64
	<i>Step 1 – Getting the Query</i>	64
	<i>Step 2 – Starting to Export Data</i>	64
	<i>Step 3 – Choosing the Source</i>	64
	<i>Step 4 – Choosing the Destination</i>	65
	<i>Step 5 – Specifying the Data to Export</i>	66
	<i>Step 6 – Specifying how to Export</i>	66
	<i>Step 7 – Finishing the Export</i>	66

TABLE OF CONTENTS (3 of 5)

10	CALCULATIONS	Page
10.1	Creating Calculated Columns	68
	<i>Giving Calculated Columns Aliases</i>	68
	<i>Using Column Aliases in ORDER BY Clauses</i>	68
	<i>Column Aliases Don't Work in WHERE Criteria</i>	69
10.2	Using SQL Functions	70
	<i>Typing an SQL Function</i>	70
	<i>Getting the Full List of Functions</i>	70
10.3	Casting Data Types	71
	<i>The Need for Casting</i>	71
	<i>Data Type Precedence</i>	72
	<i>The CAST Function</i>	73
	<i>The CONVERT Function</i>	73
10.4	Numerical Calculations	74
	<i>Mathematical Symbols and BODMAS</i>	74
	<i>The Modulus Operator (%)</i>	74
	<i>Mathematical Functions</i>	75
	<i>The Importance of Casting Numbers for Calculations</i>	76
	<i>A Short-Cut to Forcing the Right Number Type</i>	76
10.5	Text Calculations	77
	<i>Concatenating Text using the CONCAT Function</i>	77
	<i>Concatenating Text Using the Plus Sign (with Data Conversion)</i>	77
	<i>Functions to Turn Numbers into Text</i>	79
	<i>Functions to Search for and Replace Text</i>	79
	<i>Functions for Extracting Text</i>	80
	<i>Changing the Case of Text</i>	80
	<i>Functions for Trimming Text</i>	81
	<i>Other Text Functions</i>	81
	<i>Worked Example – 1</i>	82
	<i>Worked Example – 2</i>	82
	3	82
10.6	Dealing with Nulls	83
	<i>The ISNULL Function</i>	83
	<i>The COALESCE Function</i>	84
10.7	Testing Conditions using IIF	85

11	THE CASE EXPRESSION	Page
11.1	The Searched Case Expression	86
	<i>Example: Film Bands</i>	86
	<i>Example: Film Era</i>	87
	<i>Using CASE in WHERE Criteria</i>	87
11.2	The Simple Case Statement	88
11.3	Nested CASE Statements	89

12	DATE CALCULATIONS	Page
12.1	How Dates and Times Work	90
	<i>How SQL Server Stores Dates and Times</i>	90
	<i>Displaying Dates/Times</i>	90
	<i>GETDATE – the Current Date/Time</i>	91
	<i>Dates Prefer American Format</i>	91
12.2	Formatting Dates using FORMAT	92
	<i>The Available Codes</i>	92
	<i>Using the Culture Argument</i>	93
	<i>How Slow is the Format Function?</i>	93
12.3	Formatting Dates using CONVERT	94
12.4	Parts of a Date: DATEPART and DATENAME	95
	<i>Displaying a Day Suffix</i>	96
12.5	Getting the Difference between Dates	96
	<i>Subtracting One Date from Another</i>	97
	<i>The DATEDIFF Function</i>	97
12.6	Calculating Ages Correctly	98
	<i>Using DateDiff</i>	98
	<i>Dividing Someone's Age in Days by 365</i>	98
	<i>Getting the Exact Age</i>	99
12.7	Adding Dates using DATEADD	100

TABLE OF CONTENTS (4 of 5)

13	JOINS	Page
13.1	Overview of Joins	101
	<i>What is a Join?</i>	101
	<i>The Types of Join</i>	101
13.2	Understanding your Database	102
	<i>How Relationships Work (Reminder)</i>	102
13.3	Easy Joins, using Query Designer	103
13.4	Inner Joins	104
	<i>The Syntax of an Inner Join</i>	104
	<i>Our Example – Joining the Film and Director Tables</i>	104
	<i>Joining more than One Table</i>	105
	<i>Variations on Inner Join Syntax</i>	105
	<i>Composite Joins</i>	106
	<i>Joining by Expressions</i>	106
13.5	Outer Joins	108
	<i>Outer Joins using Query Designer</i>	108
	<i>Outer Joins in SQL</i>	109
	<i>Left and Right Outer Joins</i>	109
	<i>Picking Out Unmatched Rows</i>	110
	<i>Full Outer Joins</i>	110
13.6	Cross Joins	111
	<i>A Practical Example of Cross Joins</i>	111
13.7	Self-Joins	112

14	SUMMARISING DATA	Page
14.1	Simple Summarising	113
	<i>Syntax of a Simple Summary</i>	113
14.2	Counting	114
	<i>Counting All of a Table's Rows</i>	114
	<i>Counting Non-Null Columns</i>	114
	<i>Counting Unique Values</i>	115
14.3	Grouping	116
	<i>Why you Need GROUP BY</i>	116
	<i>The GROUP BY Clause</i>	116
	<i>Grouping by Multiple Columns</i>	117
	<i>Grouping by Expressions</i>	117
	<i>Grouping without Aggregating</i>	118
14.4	Filtering Results using HAVING	119
14.5	Casting Data for (eg) Averages	120
14.6	Dealing with Nulls	121
	<i>The Default Treatment of Nulls</i>	121
	<i>Forcing SQL to Include Nulls</i>	121
14.7	Additional Options when Grouping	122
	<i>Using ALL to Show Missing Rows</i>	122
	<i>Using CUBE to Show All Combinations</i>	122
	<i>Using GROUPING to Show Levels</i>	123

15	VIEWS	Page
15.1	Why Views are Useful	124
	<i>Use 1: Pre-Joining Tables</i>	124
	<i>Use 2: Virtually Renaming Columns</i>	125
15.2	Views using the Designer	126
	<i>Starting the Designer</i>	126
	<i>Choosing Columns</i>	126
	<i>Sorting and Filtering</i>	127
	<i>Adding Grouping</i>	127
	<i>Executing a View</i>	128
	<i>Saving and Closing Views</i>	129
	<i>Seeing your View in Object Explorer</i>	129
	<i>Running a View</i>	130
	<i>Changing a View</i>	130
15.3	Scripting Views	131
	<i>Creating a New View</i>	131
	<i>Changing an Open View in Script</i>	132
	<i>Changing a View's Script from Object Explorer</i>	132
15.4	Switching between the Designer and Scripting	133

TABLE OF CONTENTS (5 of 5)

16	CTES AND DERIVED TABLES	Page
16.1	Multi-Stage Queries	134
16.2	Derived Tables	135
16.3	Single CTEs (Common Table Expressions)	136
	<i>Syntax of Single CTEs</i>	136
	<i>The CTE for our Example</i>	136
16.4	Multiple CTEs	137
	<i>Syntax of Multiple CTEs</i>	137
	<i>Example of a Multiple CTE</i>	138

17	SUBQUERIES	Page
17.1	Single-Value Subqueries	139
	<i>Example: Showing the Name of the Longest Film</i>	139
17.2	ANY, ALL, IN and NOT IN	140
17.3	Correlated Subqueries	141
	<i>Correlated Subqueries: Definition and Example</i>	141
	<i>Alternatives to Correlated Subqueries</i>	141
	<i>Considering Speed</i>	142
	<i>Using EXISTS to Check whether Rows are Returned</i>	142

18	RANKING AND PERCENTILES	Page
18.1	Ranking and Numbering	143
	<i>Simple Row Numbering</i>	143
18.2	Leading and Lagging	144
	<i>Example of LAG: Actors Born One Week Apart</i>	144
18.3	Percentiles	145
	<i>Percentile Rankings</i>	145

CHAPTER 1 - DESIGNING DATABASES

The world runs on relational databases. If you understand the principles upon which these are built, you'll find it much easier to write SQL to get information out of them!



*This manual gives an overview only of database design principles. If you want to delve deeper, try Googling phrases like **Third Normal Form**, **Database Normalisation** or **Entity Diagram**. If nothing else, this will give you an impressive search history in your browser!*

1.1 The Four Stages of Database Design

There are four stages to designing a relational database, shown below (using the example of creating a simple database to hold films; or movies, if you must).

Stage 1 – Deciding what to Include

A good way to do this is to create a spreadsheet of the data you want to include for each film:

Title	Oscars	Director	Date of birth	Studio
Armageddon	0	Michael Bay	17/02/1965	Touchstone Pictures
Bad Boys	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Bad Boys II	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Dead Poets Society	1	Peter Weir	21/08/1944	Touchstone Pictures
Master and Commander ...	2	Peter Weir	21/08/1944	20th Century Fox
Pearl Harbor	1	Michael Bay	17/02/1965	Touchstone Pictures
The Rock	0	Michael Bay	17/02/1965	Hollywood Pictures
The Truman Show	0	Peter Weir	21/08/1944	Scott Rudin Productions

We want to assign each film to a director, but we don't want to have to type each director's name in over and over again!

The aim of designing a relational database is to ensure that you don't hold information twice:

Title	Oscars	Director
Dead Poets Society	1	Peter Weir
Master and Commander ..	2	Peter Wier
The Truman Show	0	Peter Weird

Not only is holding duplicate information inefficient, but it also means that spelling mistakes will creep in. Here listing out films directed by **Peter Weir** would miss out the last two films, as his name has been misspelt.

Stage 2 – Dividing Data into Tables

Having decided what data you want to include, the next stage of database design is to decide which table each bit of information belongs to:

Title	Oscars	Director	Date of birth	Studio
Armageddon	0	Michael Bay	17/02/1965	Touchstone Pictures
Bad Boys	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Bad Boys II	0	Michael Bay	17/02/1965	Jerry Bruckheimer Films
Dead Poets Society	1	Peter Weir	21/08/1944	Touchstone Pictures
Master and Commander ...	2	Peter Weir	21/08/1944	20th Century Fox
Pearl Harbor	1	Michael Bay	17/02/1965	Touchstone Pictures
The Rock	0	Michael Bay	17/02/1965	Hollywood Pictures
The Truman Show	0	Peter Weir	21/08/1944	Scott Rudin Productions

↑

These are all details to do with the film itself.

↑

These are to do with the director (name / birthday).

↑

These are details to do with the studio.



There's no magic wand to make this easier, other than bitter experience of getting it wrong and having to start again! A good guideline is that if you find yourself typing in something twice, it probably belonged in a different table.

For our example above, there are clearly 3 separate entities: films, the directors who made them and the studios which produced them. Here are the fields that each table could contain:

Table	Fields
<i>Film</i>	Title and Oscars Won , plus something to identify which director and which studio made it
<i>Director</i>	Director name and Date of birth , plus some unique identifier for the director
<i>Studio</i>	Studio name , plus some unique identifier for the studio

What you need to do next is to decide what form these unique identifiers should take.

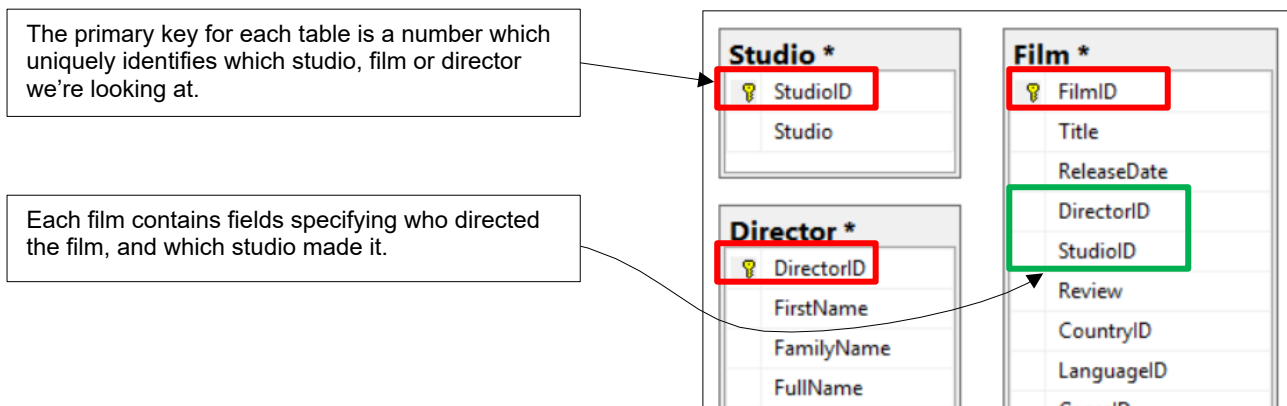
Stage 3 – Choosing a Primary Key for each Table

The *primary key* for a table is a field which tells you exactly which record you're considering (for example, if you know a film's **DirectorID** you can look up all of the director's other details).



From the above definition, it follows that two records in a table can't have the same value for the primary key field – the field is unique.

For our example, we could use the director and studio names as our primary keys, but SQL Server works most efficiently if the primary key is as short as possible, so we'll create new fields instead:



Here's what **The Sound of Music** would now look like:

FilmID	Title	OscarWins	DirectorID	StudioID
638	The Sound of Music	5	89	4

Including the director's unique number allows us to look up all their other details:

DirectorID	FullName	DoB	Gender
89	Robert Wise	10/09/1914	Male

Including the studio's unique number allows us to look up its name:

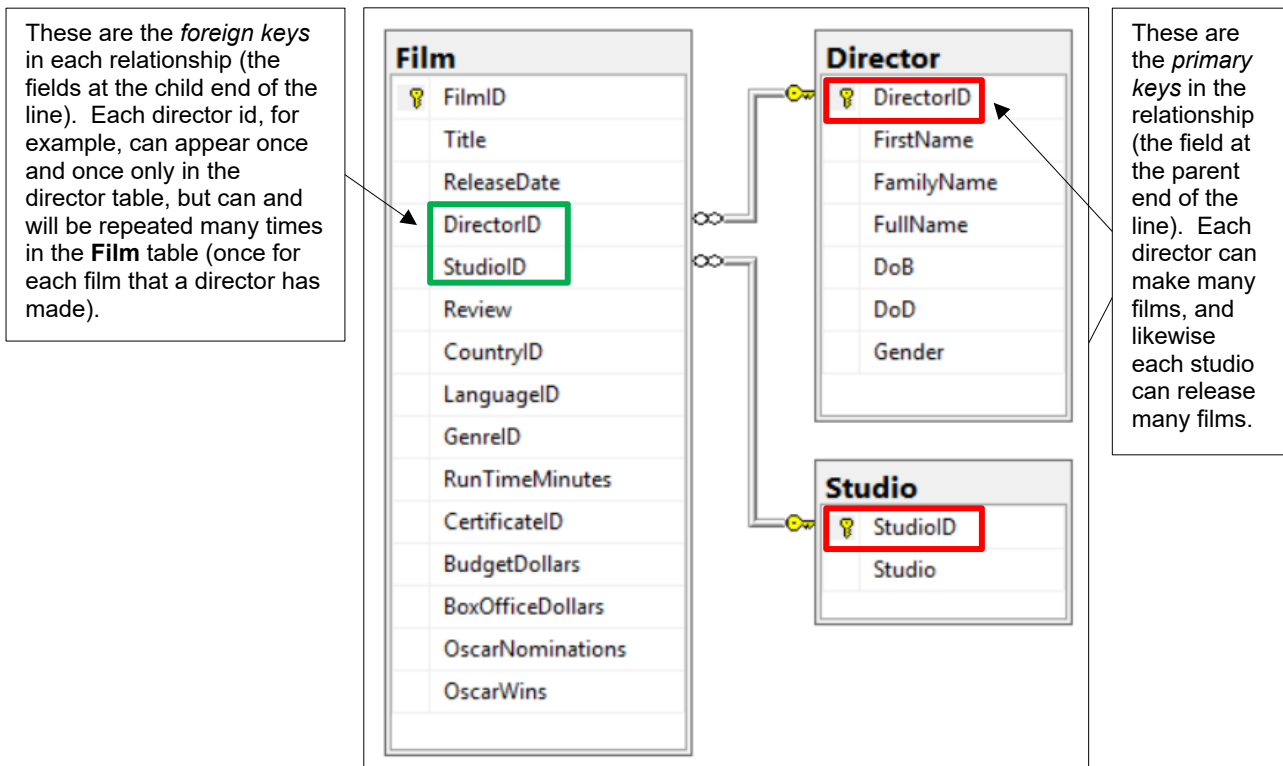
StudioID	Studio
4	20th Century Fox



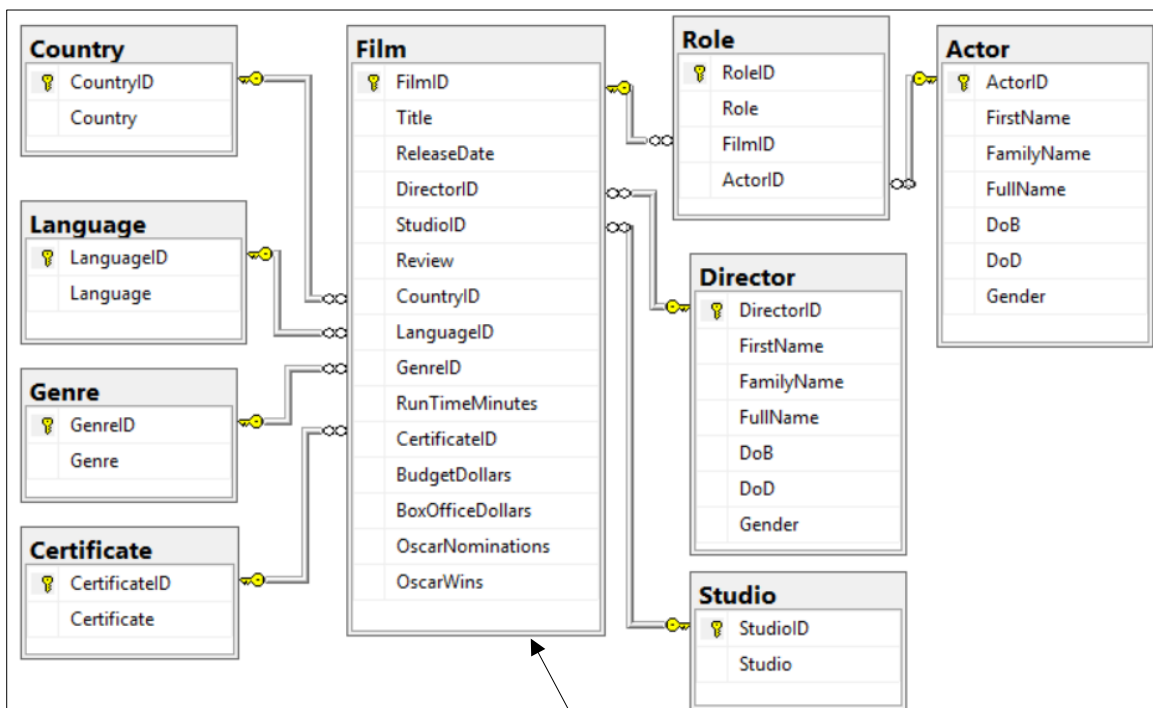
If you're beginning to think that relational databases are just like a lot of spreadsheets joined together with a more efficient version of a **VLOOKUP** formula in Excel, you're absolutely right!

Stage 4 – Creating Relationships and a Database Diagram

The last step in designing a database is to decide for each relationship that you create whether it is *one-to-many* or *many-to-one* (parent-child or child-parent):




Database diagrams often involve hundreds of tables:




The **Movies** database we'll use in this courseware contains just 9 tables, and hence is untypically simple.

1.2 Many-to-Many Relationships

There's no such thing as a many-to-many relationship in SQL Server, but they do exist in real life:




- Spider-Man
- Jurassic Park
- Mission Impossible
- Superman Returns
- Top Gun
- Rain Man
- Titanic
- Waterworld
- Pearl Harbor
- Transformers




- Sam Neill
- Tom Cruise
- Laura Dern
- Jeff Goldblum
- Jon Voight
- Vanessa Redgrave
- Kirsten Dunst
- Naomi Watts
- Jack Black
- Adrien Brody

*Tom Cruise has appeared in lots of films, but equally **Mission: Impossible** has lots of actors in it.*

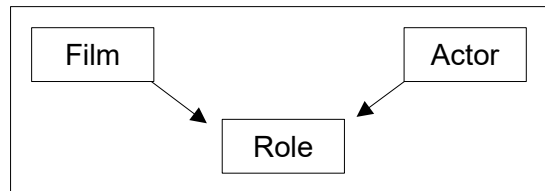


- Spider-Man
- Jurassic Park
- Mission Impossible
- Superman Returns
- Top Gun
- Rain Man
- Titanic
- Waterworld
- Pearl Harbor
- Transformers



- Sam Neill
- Tom Cruise
- Laura Dern
- Jeff Goldblum
- Jon Voight
- Vanessa Redgrave
- Kirsten Dunst
- Naomi Watts
- Jack Black
- Adrien Brody

The solution to this problem is to create a table that is a child to both of the two parent tables, as here:



Here's what the database would look like:

Film

FilmID
Title
ReleaseDate
DirectorID
StudioID
Review
CountryID
LanguageID

Role

RoleID
Role
FilmID
ActorID

Actor

ActorID
FirstName
FamilyName
FullName
DoB
DoD
Gender

The **Role** table links the **Film** and **Actor** tables. Each film can contain many roles (otherwise a film could only have a single actor), but likewise each actor can have many roles (otherwise they would never work again after completing their first film).

Here are 3 rows from the **Role** table:

RoleID	Role	FilmID	ActorID
1	Ray Ferrier	33	1
2	Dr. Alan Grant	1	2
3	Dr. Ellie Sattler	1	3
202	Nathan Algren	41	1

Here are the films and actors who are represented by these rows of data (the duplicate film name was **Jurassic Park**, and duplicated actor turns out to be **Tom Cruise**).

Film number 1 appears twice in this list, as does actor number 1.

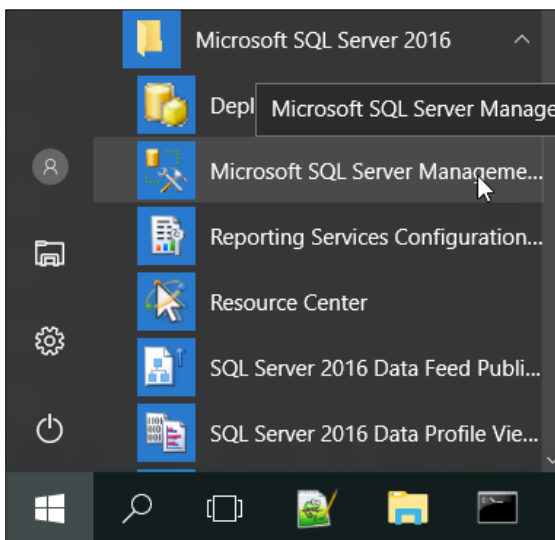
RoleID	Role	FilmID	ActorID	Title	FullName
1	Ray Ferrier	33	1	War of the Worlds	Tom Cruise
2	Dr. Alan Grant	1	2	Jurassic Park	Sam Neill
3	Dr. Ellie Sattler	1	3	Jurassic Park	Laura Dern
202	Nathan Algren	41	1	The Last Samurai	Tom Cruise

CHAPTER 2 - SQL SERVER MANAGEMENT STUDIO

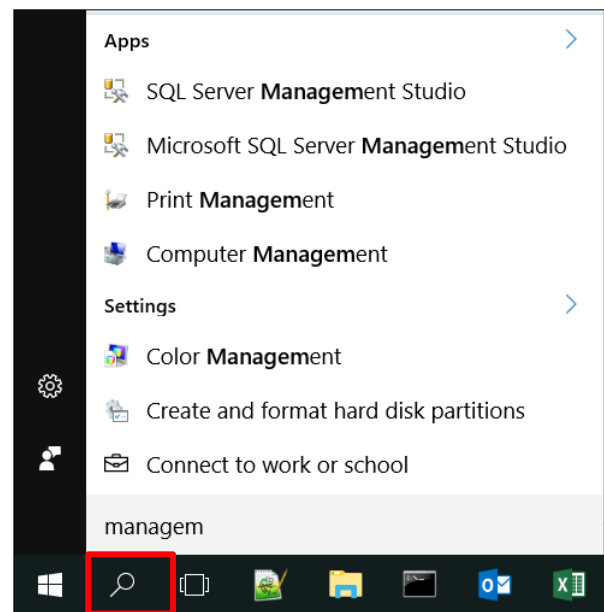
If you're writing SQL to get information out of a database created using SQL Server, the chances are that you'll use *SSMS (SQL Server Management Studio)* as your authoring tool.

2.1 Starting to Use Management Studio

You can start SSMS like any other application – here are a couple of ways using Windows 10:



Click on the Windows icon, and choose the program that you want to run ...



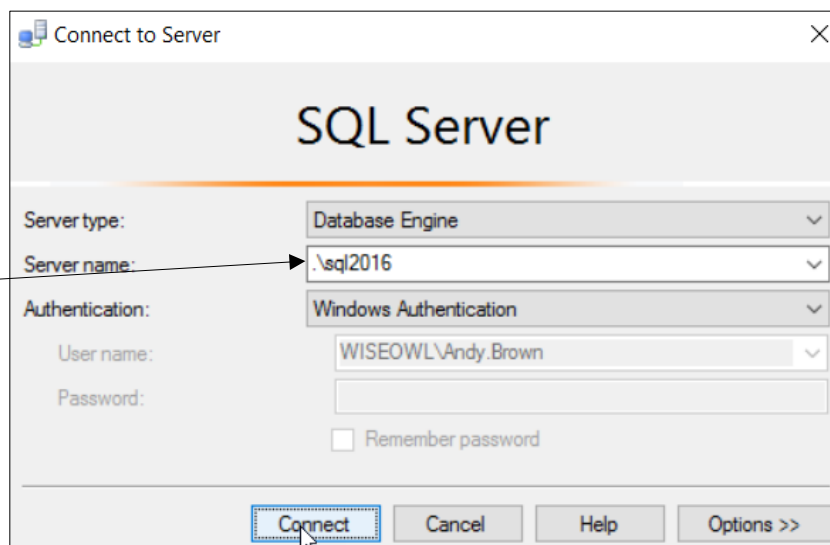
... or click on this tool and type in part of the program name (here typing **managem** has been enough to bring up the program name in the list).

You can then choose a database to use (or *connect to*):

You will see the SQL Server logo on screen while it loads:



You can then choose from the dropdown list which of your company's servers you want to connect to (your IT people should be able to advise on which to choose). If you use Windows authentication, you won't have to type in any more user names or passwords.



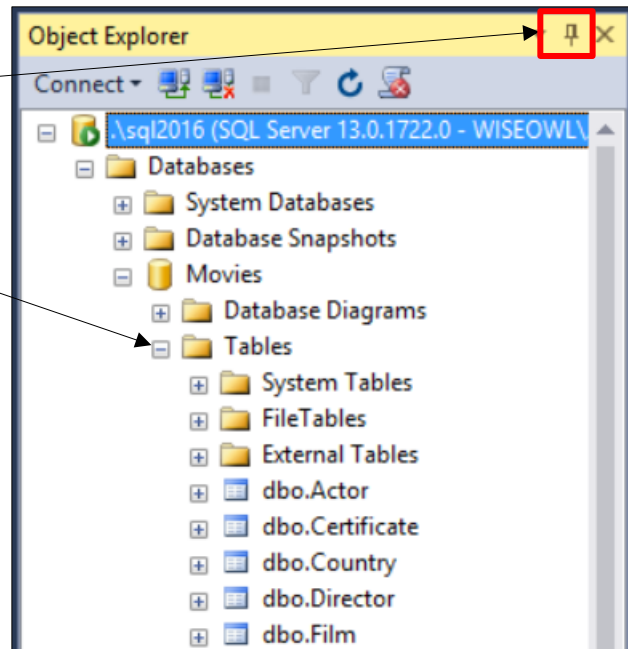
2.2 Object Explorer

When Management Studio loads, you should see the **Object Explorer** window (if it's not visible, press **F8** to show it):

If you think Object Explorer is taking up too much room, click on this pin (it will go from vertical to horizontal, and the window will collapse until needed). Click again on the pin to make the window permanently visible again.

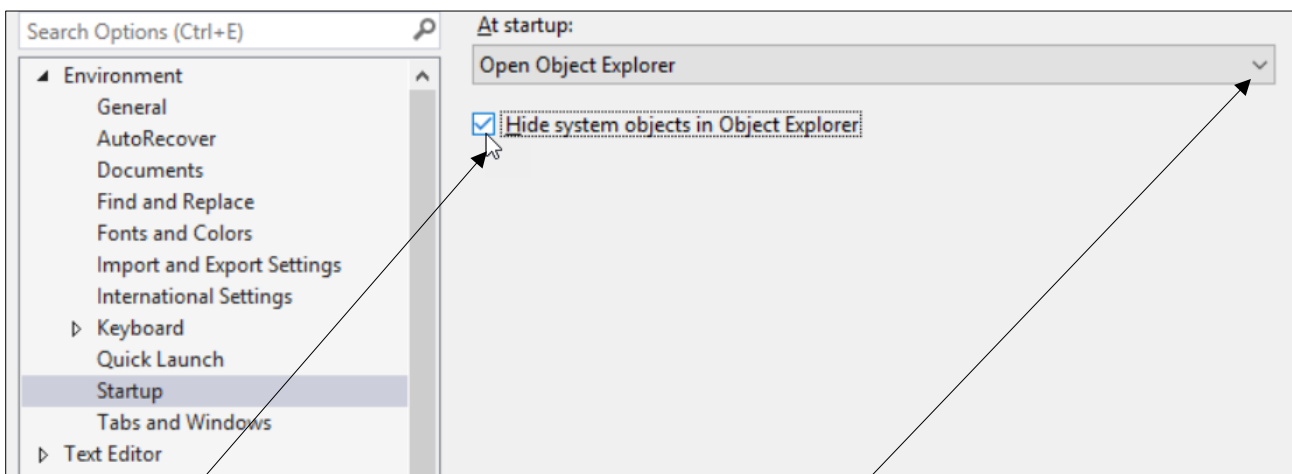
You can click on the **+** symbol to expand:

- **Databases**, to see the **Movies** database; then
- **Movies**, to see what it contains; then
- **Tables**, to see what tables there are.



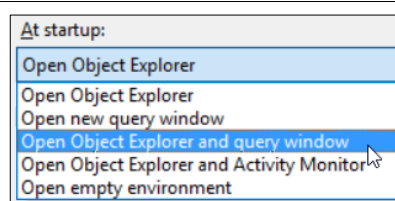
Useful Start-up Options

You can control what happens when you start Management Studio. To do this, from the menu select **Tools** → **Options**, then complete the dialog box which appears as follows:







a) System objects clutter up SQL Server, and (in this owl's opinion) are best hidden, although you won't see a huge amount of difference.

b) You can click on the drop arrow and choose (for example) to show a blank query as well as Object Explorer whenever you open Management Studio.



WHAT WE DO

	 ONLINE TRAINING	 MANCHESTER OR LONDON	 AT YOUR OFFICE	 BESPOKE CONSULTANCY	
OFFICE 365	Microsoft Excel	✓	✓	✓	✓
	VBA macros	✓	✓	✓	✓
	Office Scripts	✓		✓	
	Microsoft Access				✓
POWER PLATFORM	Power BI and DAX	✓	✓	✓	✓
	Power Apps	✓		✓	
	Power Automate	✓	✓	✓	✓
SQL SERVER	Reporting Services	✓	✓	✓	✓
	Report Builder	✓		✓	✓
	Integration Services	✓	✓	✓	✓
	Analysis Services	✓		✓	
CODING LANGUAGES	SQL	✓	✓	✓	✓
	Visual C#	✓	✓	✓	✓
	Python	✓	✓	✓	✓



WiseOwl
Training

