



Introduction to PostgreSQL

Sample manual - first two chapters



Wise Owl
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	GETTING STARTED WITH POSTGRESQL	Page
2.1	Getting Started with PostgreSQL	12
	<i>The Command Line Interface</i>	12
	<i>The GUI User Interface</i>	12
2.2	Object Explorer	13
	<i>Listing Databases</i>	13
	<i>Listing Schemas</i>	13
	<i>Listing Tables</i>	14
2.3	Setting Preferences	15
	<i>Change 1: Setting the Default Font Size</i>	15
	<i>Change 2: Changing the New Query Short-Cut</i>	16
	<i>Change 3: Enabling Intellisense Automatically</i>	16

3	CREATING TABLES	Page
3.1	Creating our Sample Database	17
3.2	Creating a Table	18
3.3	Creating Columns	19
	<i>Adding Columns to a Table while Creating It</i>	19
	<i>Adding Columns to an Existing Table</i>	20
	<i>Setting the Primary Key</i>	21
	<i>Setting Identity Columns</i>	21
	<i>Setting Default Values</i>	22
	<i>Allowing or Disallowing Nulls</i>	22
3.4	Postgres Data Types	23
	<i>Data Types Explained in this Chapter</i>	23
	<i>Integer Field Types</i>	23
	<i>Boolean Field Type</i>	24
	<i>Decimal and Numeric Field Types</i>	24
	<i>Double Precision and Real Data Types</i>	24
	<i>String Data Types</i>	25
	<i>Date/Time Data Types</i>	25
3.5	Entering Data in Tables	26

4	RELATIONSHIPS BETWEEN TABLES	Page
4.1	Understanding Relationships	27
4.2	Creating a Relationship	28
4.3	ERD Diagrams	29
	<i>Creating an ERD Diagram from a Database</i>	29
	<i>Creating an ERD Diagram from Scratch</i>	30
	<i>Adding Relationships</i>	31
	<i>Other Options Available</i>	32

5	CREATING QUERIES	Page
5.1	Creating Queries	33
	<i>Viewing a Query's Database</i>	34
	<i>Managing Query Windows</i>	34
5.2	Basic SELECT Statements	35
	<i>Where to Put your Commas</i>	35
	<i>Using Autocomplete</i>	36
5.3	Running Queries	37
	<i>Executing a Query</i>	37
	<i>Cancelling a Running Query</i>	37
	<i>Explaining a Query</i>	38
	<i>The Data Output Window</i>	38
	<i>Running Multiple SQL Commands in the Same Query</i>	39
	<i>Dealing with Errors</i>	39
	<i>Copying the Rows Returned from a Query</i>	40
5.4	Saving, Opening and Closing Queries	41
	<i>Saving Queries</i>	41
	<i>Opening Queries</i>	41
5.5	Using Upper and Lower Case	42
	<i>Changing the Default Case of SQL Keywords</i>	42
5.6	Indentation and Word Wrap	43
	<i>Changing Indentation</i>	43
	<i>Word Wrap</i>	43
5.7	Comments	44
	<i>Commenting Out Blocks of Code</i>	44
5.8	Colours in SQL	45
5.9	Auto-formatting SQL	46

TABLE OF CONTENTS (2 of 5)

6	THE SELECT STATEMENT	Page
6.1	SELECT Statement Syntax	47
	<i>Mnemonic for Order of Commands</i>	47
6.2	Qualified Tables and Columns	48
	<i>Dragging Tables/Columns onto a Query Schemas</i>	48
6.3	Table Aliases	48
	<i>Reason 1 for Aliases – Easier to Refer to Field Names</i>	49
	<i>Reason 2 for Aliases – Joins</i>	50
	<i>Changing a Table Alias</i>	50
6.4	Column Aliases	51
	<i>Basic Column Aliases</i>	51
	<i>The Word AS is Optional</i>	51
	<i>Aliases in WHERE and ORDER BY Clauses</i>	52
6.5	Ordering Rows	53
	<i>Simple Sorting</i>	53
	<i>Sorting by Multiple Columns</i>	53
6.6	Miscellaneous SELECT Tricks	54
	<i>Selecting All Columns Using the Asterisk</i>	54
	<i>Selecting Unique Rows</i>	54
	<i>Showing Top and Bottom Rows</i>	55
	<i>Including Ties</i>	55
6.7	Using UNION to Combine Results	56

7	CRITERIA USING WHERE	Page
7.1	The WHERE Clause	57
	<i>Relational Operators</i>	57
7.2	Criteria with Numbers	58
	<i>Using Comparisons</i>	58
	<i>Finding Numbers in a Given Range</i>	58
7.3	Criteria using Text	59
	<i>Exact Matches</i>	59
	<i>Wildcard Matches using LIKE and ILIKE</i>	59
	<i>Using Special Characters with LIKE</i>	60
	<i>Ranges and Regular Expressions</i>	60
	<i>Using Similar To</i>	61
7.4	Criteria for Dates	62
	<i>Using Dates in Criteria</i>	62
7.5	Combining Criteria	63
	<i>Example using AND</i>	63
	<i>Example using OR</i>	63
	<i>Criteria Explained using Venn Diagrams</i>	64
	<i>Using Parentheses</i>	64
7.6	Nulls	65
	<i>An Example of Testing for Nulls</i>	65

8	IMPORTING AND EXPORTING DATA	Page
8.1	Copying and Pasting	66
8.2	Exporting Rows for Tables	67
	<i>Starting the Export Process</i>	67
	<i>Choosing to Include Columns</i>	68
	<i>Choose Columns to Include</i>	68
	<i>Finishing the Export</i>	69
	<i>Viewing Errors</i>	69
8.3	Exporting Queries	70
	<i>Exporting a Query Via a Table</i>	70
	<i>Exporting a Query by Pasting its SQL</i>	70
8.4	Loading Data into Excel or Power BI	72
	<i>Installing the Driver</i>	72
	<i>Add an ODBC Data Source</i>	72
	<i>Loading Data into Excel</i>	73

TABLE OF CONTENTS (3 of 5)

9	CALCULATIONS	Page
9.1	Creating Calculated Columns	74
	<i>Giving Calculated Columns Aliases</i>	74
	<i>Using Column Aliases in ORDER BY Clauses</i>	74
	<i>Column Aliases Don't Work in WHERE Criteria</i>	75
9.2	Using SQL Functions	76
	<i>Typing an SQL Function</i>	76
	<i>Getting Help on PostgreSQL Functions</i>	76
9.3	Casting Data Types	77
	<i>The Need for Casting</i>	77
	<i>Casting Using the CAST Function</i>	77
	<i>Casting using the Cast Operator</i>	78
9.4	Numerical Calculations	79
	<i>Mathematical Symbols and BODMAS</i>	79
	<i>The Modulus Operator (%)</i>	79
	<i>Powers and Roots</i>	80
	<i>Rounding</i>	80
	<i>Other Useful Mathematical Functions</i>	80
	<i>The Importance of Casting Numbers for Calculations</i>	81
	<i>A Short-Cut to Forcing the Right Number Type</i>	81
9.5	Text Calculations	82
	<i>Concatenating Text</i>	82
	<i>Changing the Case of Text</i>	82
	<i>Getting Numerical Information</i>	83
	<i>Extracting Text</i>	83
	<i>Manipulating Text</i>	84
9.6	Examples of Text Functions	85
	<i>Example: Counting Words</i>	85
	<i>Example: Getting the Last Word (1)</i>	85
	<i>Example: Getting the Last Word (2)</i>	86
9.7	Dealing with Nulls – the COALESCE Function	87

11	DATE CALCULATIONS	Page
11.1	How Dates and Times Work	93
	<i>How PostgreSQL Stores Dates</i>	93
	<i>How PostgreSQL Stores Times</i>	93
11.2	Basic Dates and Times	93
	<i>Date and Time Types</i>	94
	<i>Getting the Current Date/Time</i>	94
11.3	Representing Dates (and Times)	95
	<i>The Main Components of a Date</i>	95
	<i>The FM Modifier</i>	96
	<i>Ordinal Suffices</i>	96
	<i>Putting Everything Together</i>	96
	<i>Time Format Codes</i>	97
11.4	Extracting Parts of Dates	98
	<i>The Extract Function</i>	98
	<i>Keywords for Extract</i>	99
	<i>The Date_Part Function</i>	99
	<i>Truncating Dates/Times</i>	100
11.5	Getting the Interval Between Dates	101
	<i>Simple Date Subtraction</i>	101
	<i>Introducing Intervals</i>	101
	<i>Date/Time Subtraction Returning Intervals</i>	101
	<i>Getting True Ages</i>	102
	<i>Using Extract with Intervals</i>	103
11.6	Adding and Subtracting Dates	104
	<i>Adding and Subtracting Days</i>	104
	<i>Adding and Subtracting Other Time Intervals</i>	104
	<i>Using Date_Add and Date_Subtract</i>	105
	<i>Generating Intervals from Numbers</i>	106

10	THE CASE EXPRESSION	Page
10.1	The Searched Case Expression	88
	<i>Example: Film Bands</i>	88
	<i>Example: Film Era</i>	89
	<i>Using CASE Expressions in WHERE Criteria</i>	90
10.2	The Simple Case Statement	91
10.3	Nested CASE Statements	92

TABLE OF CONTENTS (4 of 5)

12	JOINS	Page
12.1	Overview of Joins	107
	<i>What is a Join?</i>	107
	<i>The Types of Join</i>	107
12.2	Understanding your Database	108
	<i>How Relationships Work (Reminder)</i>	108
12.3	Inner Joins	109
	<i>The Syntax of an Inner Join</i>	109
	<i>Our Example – Joining the Film and Director Tables</i>	109
	<i>Joining more than One Table</i>	110
	<i>Variations on Inner Join Syntax</i>	111
	<i>Composite Joins</i>	112
	<i>Joining by Expressions</i>	112
12.4	Outer Joins	114
	<i>Creating Outer Joins in SQL</i>	114
	<i>Right Outer Joins</i>	114
	<i>Picking Out Unmatched Rows</i>	115
	<i>Full Outer Joins</i>	115
12.5	Cross Joins	116
	<i>A Practical Example of Cross Joins</i>	116
12.6	Self-Joins	117

13	SUMMARISING DATA	Page
13.1	Simple Summarising	118
	<i>Syntax of a Simple Summary</i>	118
13.2	Counting	119
	<i>Counting All of a Table's Rows</i>	119
	<i>Counting Non-Null Columns</i>	119
	<i>Counting Unique Values</i>	120
	<i>Using Filter to Count a Subset of Rows</i>	120
13.3	Grouping	121
	<i>Why you Need GROUP BY</i>	121
	<i>The GROUP BY Clause</i>	121
	<i>Grouping by Multiple Columns</i>	122
	<i>Grouping by Expressions</i>	122
	<i>Grouping without Aggregating</i>	123
13.4	Filtering Results using HAVING	124
13.5	Dealing with Nulls	125
	<i>The Default Treatment of Nulls</i>	125
	<i>Forcing SQL to Include Nulls</i>	125
13.6	The CUBE and ROLLUP Operators	126
	<i>Using CUBE to Show All Combinations</i>	126
	<i>Using ROLLUP to Show Most Combinations</i>	127
13.7	String Aggregation	128

14	VIEWS	Page
14.1	What Views Are	129
14.2	Why Views are Useful	130
	<i>Use 1: Pre-Joining Tables</i>	130
	<i>Use 2: Virtually Renaming Columns</i>	131
14.3	Creating Views	132
14.4	Changing Views	133
	<i>Scripting Changes</i>	133
	<i>Why Changes Rarely Work</i>	133
	<i>Solution 1 – Delete and Recreate the View</i>	134
	<i>Solution 2 – Amend the Existing View</i>	134
14.5	Sorting in Views	135
14.6	Joining to Views	136

15	CTES AND DERIVED TABLES	Page
15.1	Multi-Stage Queries	137
15.2	Derived Tables	138
	<i>Single Derived Tables</i>	138
	<i>Multiple Derived Tables</i>	139
15.3	Single CTEs (Common Table Expressions)	140
	<i>Syntax of Single CTEs</i>	140
	<i>The CTE for our Example</i>	140
15.4	Multiple CTEs	141
	<i>Syntax of Multiple CTEs</i>	141
	<i>Example of a Multiple CTE</i>	142

16	SUBQUERIES	Page
16.1	Single-Value Subqueries	143
	<i>Example: Showing the Name of the Longest Film</i>	143
16.2	ANY, ALL, IN and NOT IN	144
16.3	Correlated Subqueries	145
	<i>Definition and Example</i>	145
	<i>Alternatives to Correlated Subqueries</i>	145
	<i>Using EXISTS to Check whether Rows are Returned</i>	146

TABLE OF CONTENTS (5 of 5)

17	POSTGRES AND AI	Page
17.1	Choosing an AI Tool	147
17.2	Blind Queries	148
17.3	Querying a Specific Database	149
	<i>Creating a Backup of a Schema</i>	149
	<i>Uploading your Schema File</i>	150
	<i>Requesting a Query</i>	150
17.4	Tidying Up Queries	151
17.5	Optimising Queries	152
17.6	Debugging SQL	153

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.

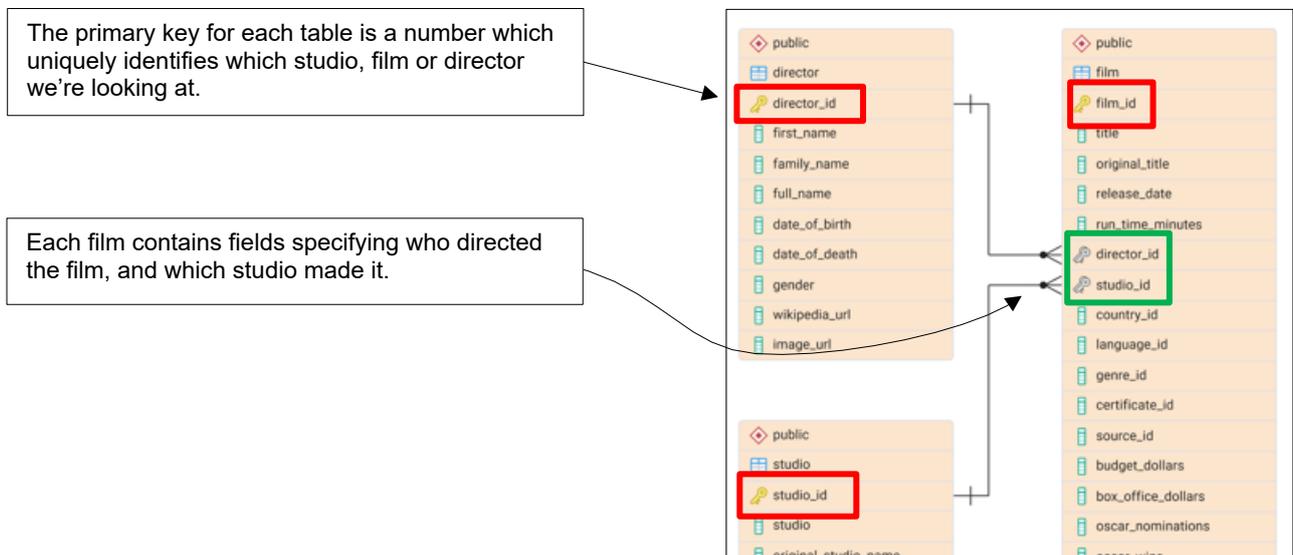
Stage 3 – Choosing a Primary Key for each Table

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 **director_id** 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 *PostgreSQL* 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:

film_id [PK] integer	title character varying (255)	oscar_wins smallint	director_id integer	studio_id integer
638	The Sound of Music	5	89	4

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

director_id [PK] integer	full_name text	date_of_birth date	gender character vary
89	Robert Wise	1914-09-10	Male

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

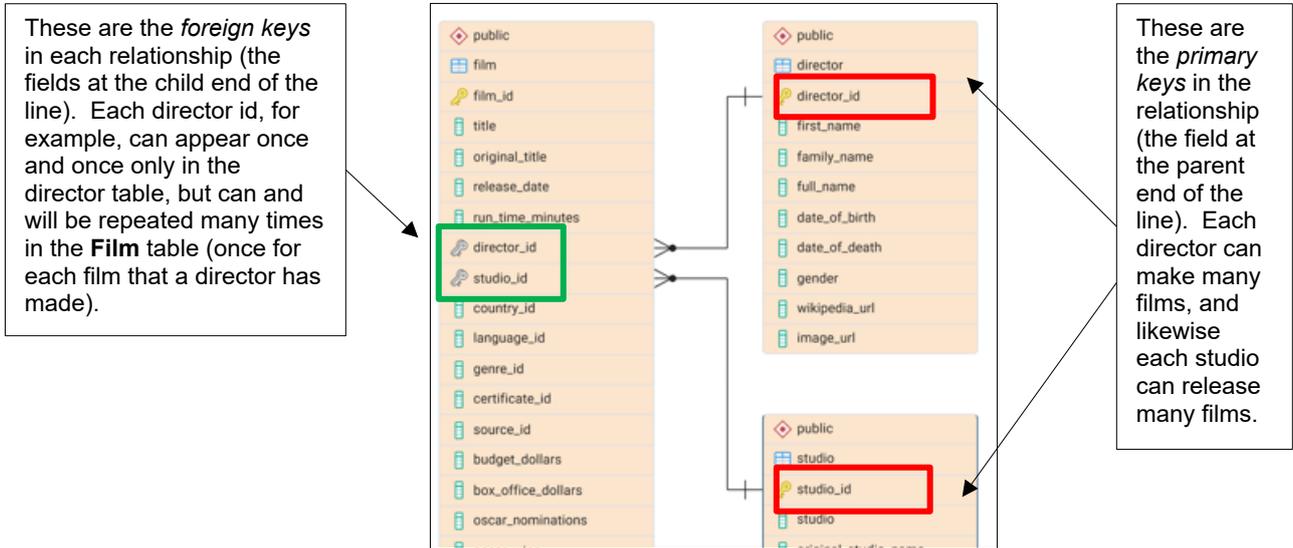
studio_id [PK] integer	studio character varying (255)
4	20th Century Studios



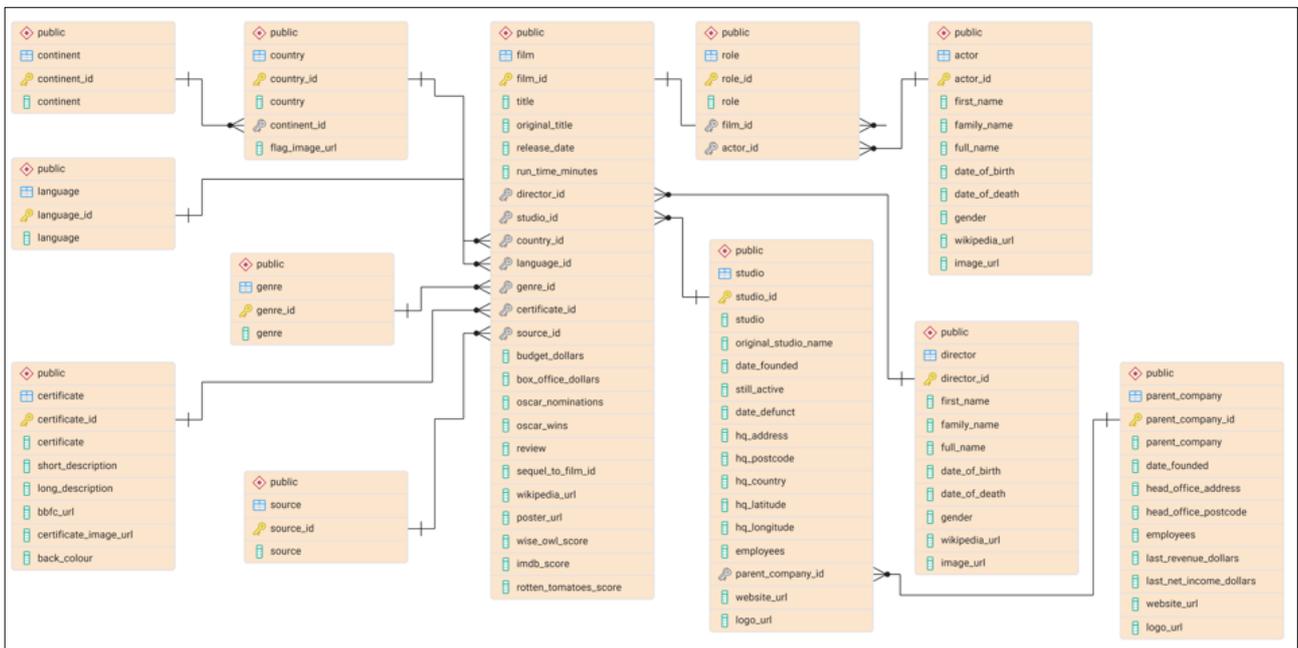
If you're beginning to think that relational databases are just like a lot of spreadsheets joined together with a more efficient version of a **VLOOKUP** or **XLOOKUP** formula in Excel, you're absolutely right!

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:



Our **Movies** database contains just 12 tables, all of which are shown above, and hence is untypically simple.



The above diagrams were created using the ERD Entity-Relationship Diagram facility in PG Admin (read on for how to install this).

1.2 Many-to-Many Relationships

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



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



Sam Neill
Tom Cruise
Laura Dern
Jeff Goldblum
Jon Voight
Vanessa Redgrave
Kirsten Dunst
Naomi Watts

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

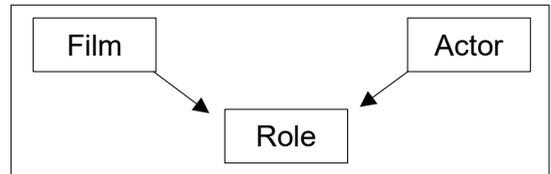


Spider-Man
Jurassic Park
Mission Impossible
Superman Returns
Top Gun
Rain Man
Titanic
Waterworld



Sam Neill
Tom Cruise
Laura Dern
Jeff Goldblum
Jon Voight
Vanessa Redgrave
Kirsten Dunst
Naomi Watts

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:

public
film
film_id
title
original_title
release_date
run_time_minutes
director_id
studio_id

public
role
role_id
role
film_id
actor_id

public
actor
actor_id
first_name
family_name
full_name
date_of_birth
date_of_death
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:

role_id [PK]	role	film_id	actor_id
1	Ray Ferrier	33	1
2	Dr. Alan Grant	1	2
3	Dr. Ellie Sattler	1	3
202	Nathan Algren	41	1

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

role_id	role	film_id	actor_id	title	full_name
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 - GETTING STARTED WITH POSTGRESQL

2.1 Getting Started with PostgreSQL

Postgres was created in 1986 as the successor to the older *INGRES* database system (hence the name *Postgres*, although it's also often called *PostgreSQL*). *Postgres* usually comes in two parts.

The Command Line Interface

You first need to install the core *PostgreSQL* database from <https://www.postgresql.org/download/> :



Installing *PostgreSQL* from the official website is simple – and free!

```
C:\Program Files\PostgreSQL\18\bin>psql -U postgres -d movies_02
Password for user postgres:

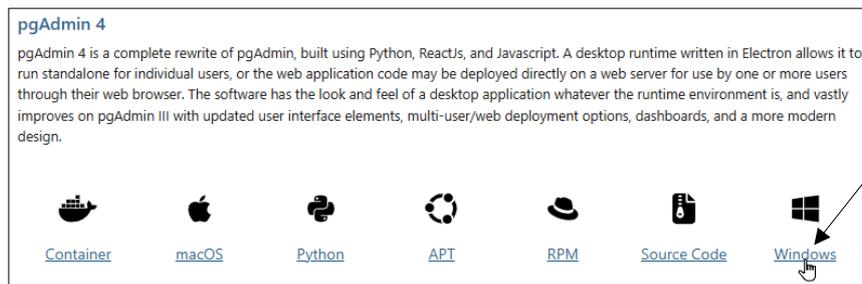
psql (18.1)
WARNING: Console code page (850) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.

movies_02=# SELECT * FROM public.film;
 film_id |
          original_title
          | director_id | studio_id | country_id | language_id | genre_id | c
office_dollars | oscar_nominations | oscar_wins |
          review
          | sequel_to_film_id |
```

An example of using the command line interface to run *PostgreSQL* commands. There is no reason the author can think why you would ever want to do this!

The GUI User Interface

There are a few graphical interfaces available for using Postgres, but by far the most common (and hence the best choice) is *PG Admin*:

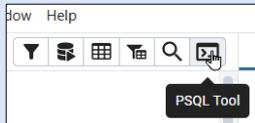


PG Admin is also simple to install, and also free! It gives you this icon if you add it to your desktop:





Should you ever want to get at the PostgreSQL command line interface from within PG Admin, you can. Just click on this tool on the Object Explorer toolbar:

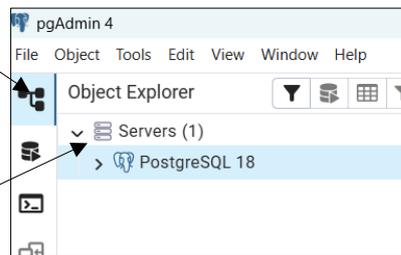


2.2 Object Explorer

You will spend nearly all your time in *PostgreSQL* using *Object Explorer*:

Should you ever accidentally leave Object Explorer, you can return to it by clicking on this icon.

Theoretically you could use Object Explorer to navigate between different installed instances or versions of *PostgreSQL*, but in practice you'll only ever have one instance.

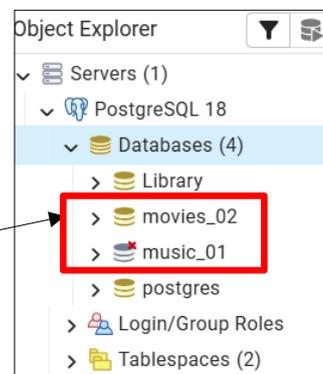


Listing Databases

You can expand the *PostgreSQL* server to see the databases created:

The **Library** and **Postgres** databases are system ones, which you should ignore. The other two databases are as follows:

Database	What it contains
movies_02	The films, directors, studios, actors and roles in the Wise Owl movies database, referenced throughout this courseware.
music_01	The artists, albums, songs and tours referenced by the exercises used in the Wise Owl SQL courses.

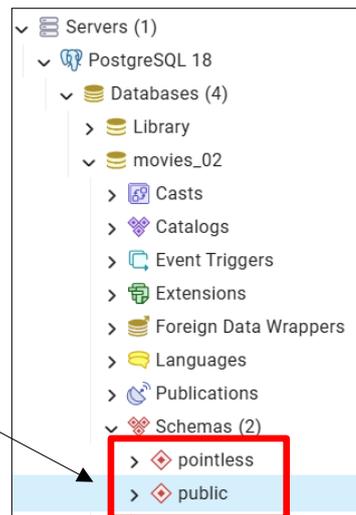


Listing Schemas

Expanding a database further will give you a list of the *schemas* it contains:

This database contains two schemas, although only one (**public**) is used. Creating different schemas serves two purposes:

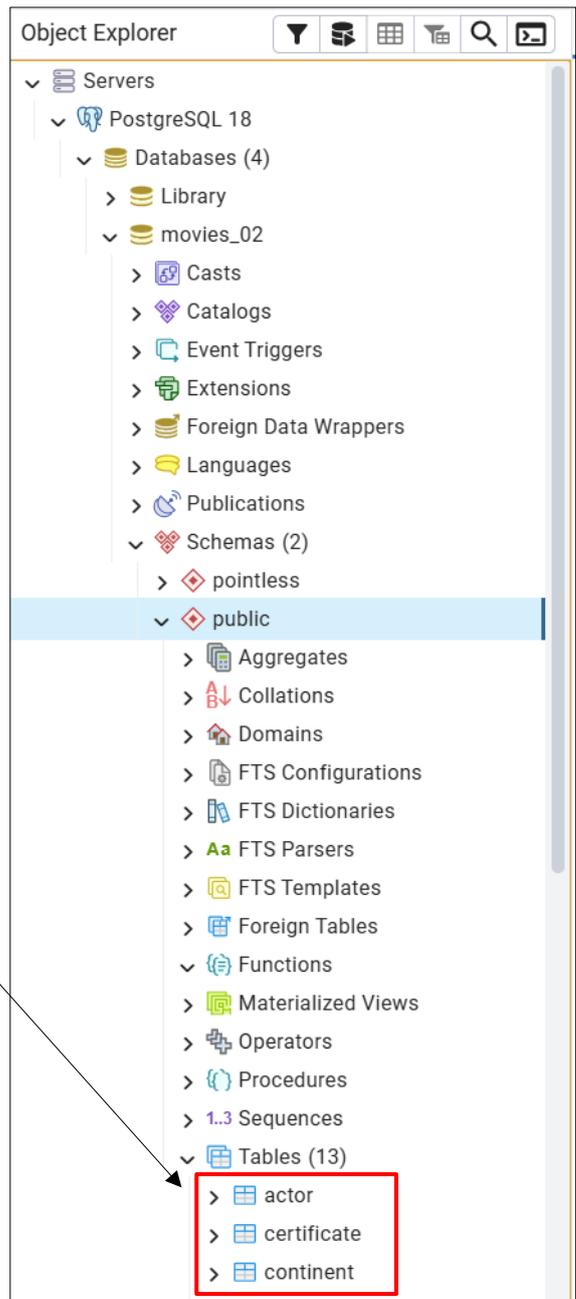
- It lets you group tables into different sets; and
- It allows you to apply different security rules to different sets of tables (for example, people in HR might be able to see the **hr** schema tables but not the **accounts** ones).



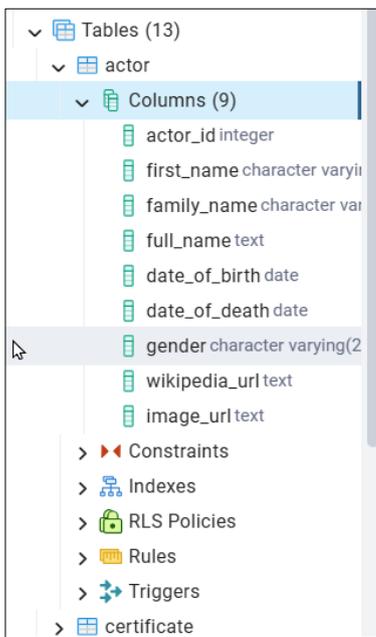
When you create a new table it will (by default) join the public schema. Usually the best thing to do is to accept this default, as it means that you can find all of your tables in the same place.

Listing Tables

If you keep expanding the Object Explorer nodes for long enough, you will eventually get to the tables in a database:



The first few tables in this database. You can expand any table further to see the columns that it contains. Here for example are the columns in the **actor** table:



2.3 Setting Preferences

Everybody will have their own preferred way of working, but the author will always change 3 preferences in PG Admin before doing anything else.

a) Choose this **Preferences** option from the PG Admin **File** menu.

b) Choose the preference category on the left that you want to work with, then change one or more properties within it.

c) Crucially, remember to click on the save button to save your changes!

Change 1: Setting the Default Font Size

This owl finds that SQL in PG Admin is much more readable at 20% bigger than the default size!

a) Type **font** into the search box to find properties to do with fonts.

b) Change the font size to a multiple of 1 (here we've gone for 1.2, or 120% of the default size) to get much more readable SQL!

```

Query    Query History
1
2      -- much more readable!
3      SELECT * FROM film
4
    
```

Change 2: Changing the New Query Short-Cut

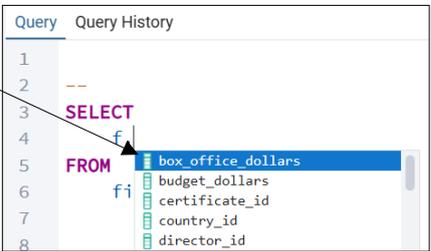
You will create many new queries in PG Admin. The default short-cut key is **Ctrl** + **Alt** + **Q** which is hard to remember – this owl changes this to **Ctrl** + **N** :

The screenshot shows the 'Keyboard shortcuts' category selected in the left sidebar. The 'Open query tool' row is highlighted with a red box. The key combination is set to Shift, Ctrl, Alt, and N. Callout boxes provide instructions: 'a) Choose the Keyboard shortcuts category within the Browser tab.' and 'b) Tick the combination of Shift, Ctrl and Alt keys which you want to have to press to create a new query and type in the letter to accompany them (here n).

Change 3: Enabling Intellisense Automatically

For some strange reason the automatic completion of commands shown on the right doesn't appear by default in PG Admin.

Normally you have to press **Ctrl** + **Space Bar** to get this list of possible columns to appear.



Here's how to change this:

The screenshot shows the 'Query Tool' category selected in the preferences. The 'Auto completion on key press' toggle is turned on. Callout boxes provide instructions: 'In preferences select the Auto completion tab in the Query Tool category.' and 'b) Turn this feature on. You'll thank yourself (and us)!'

What we do!

		Basic training	Advanced training	Systems / consultancy
Office	Microsoft Excel			
	VBA macros			
	Office Scripts			
	Microsoft Access			
Power BI, etc	Power BI and DAX			
	Power Apps			
	Power Automate (both)			
SQL Server	SQL			
	Reporting Services			
	Report Builder			
	Integration Services			
	Analysis Services			
Coding	Visual C#			
	VB programming			
	MySQL			
	Python			



WiseOwl
Training

