

Database

SQL



What is a Database?

What is a Database?

A [database](#) is an organized collection of structured information, or data, typically stored electronically in a computer system.

A database is usually controlled by a [database management system](#) (DBMS).

Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

Databases Versus Spreadsheets

Databases and [spreadsheets](#) (such as Microsoft Excel) are both convenient ways to store information. The primary differences between the two are:

- How the data is stored and manipulated
- Who can access the data
- How much data can be stored

Spreadsheets were originally designed for one user, and their characteristics reflect that. They're great for a single user or small number of users who don't need to do a lot of incredibly complicated data manipulation.

Databases, on the other hand, are designed to hold much larger collections of organized information—massive amounts, sometimes. Databases allow multiple users at the same time to quickly and securely access and query the data using highly complex logic and language.

Types of Database

There's a lot of different kinds of databases but we will be focussing on [relational databases](#).

Items in a relational database are organized as a set of tables with columns and rows. Entities/tables in a relational database have relationships with each other, which is how we will access rows/records of data in different tables/entities.

Relational database technology provides an incredibly efficient and flexible way to access structured information.

What is SQL?

SQL (Structured Query Language) is a standard database language used to create, maintain, and retrieve information from a relational database.

Good to know:

- SQL is case-insensitive but it is recommended to type keywords UPPERCASE (SELECT, JOIN, UPDATE, CREATE etc.) and things like table and column names in lowercase
- Comments are made using a double hyphen (--) at the beginning of a line
- MySQL is a slightly different flavour than MSSQL and queries written for one might not work for the other. We'll be focussing on MySQL

What is MySQL?

- Relational database management system (DBMS) based on SQL
- Designed and optimized for web applications
- Can run on any platform
- WordPress uses it
- AirBnB, Uber, LinkedIn, Facebook, Twitter, and YouTube are all powered by MySQL
- Used to be open source but was sold to Oracle so we'll be using MariaDB which is an open source version of MySQL

Glossary

| | |
|-------------|---|
| Entity | Something of interest to the database user community. Examples include customers, parts, geographic locations, etc. |
| Column | An individual piece of data stored in a table. |
| Row | A set of columns that together completely describe an entity or some action on an entity. Also called a record. |
| Table | A set of rows, held either in memory (nonpersistent) or on permanent storage (persistent). |
| Result set | Another name for a non persistent table, generally the result of an SQL query. |
| Primary Key | One or more columns that can be used as a unique identifier for each row in a table. |
| Foreign Key | One or more columns that can be used together to identify a single row in another table. |

Data Types

Data Types

A data type is an attribute that specifies the type of data that the object can hold: integer data, character data, monetary data, date and time data, binary strings, and so on.

[Here is a great summary of the different data types available to you in SQL!](#)

Make sure you stick to the MySQL data types, as these are what we'll be using in this course.

String (Text) Data Types

- CHAR
- VARCHAR
- BINARY
- VARBINARY
- TINYBLOB
- TINYTEXT
- TEXT
- BLOB
- MEDIUMTEXT
- MEDIUMBLOB
- LONGTEXT
- LONGBLOB
- ENUM, SET,

Number Data Types

- BIT
- TINYINT
- BOOL/BOOLEAN
- SMALLINT
- MEDIUMINT
- INT/INTEGER
- BIGINT
- FLOAT
- DOUBLE/DOUBLE PRECISION
- DECIMAL/DEC

*For some number data types, you may see the use of keywords: **SIGNED** or **UNSIGNED**. This simply refers to whether the value can be positive and negative (signed), or only positive (unsigned.)

Date Data Types

- DATE
- DATETIME
- TIMESTAMP
- TIME
- YEAR

Selecting Data Types

The trick with data types is to pick carefully which one will best represent the data you'll be storing.

As an example, say you're hoping to store first names. INT or DATE are not likely to be useful candidates for this data, as they aren't meant to represent a flexible assortment of text characters. Perhaps something like TEXT could work... however, if you look into its definition it is capable of storing up to 2 gigabytes of information. This would be way too much, first names are never that long. Something more reasonable might be VARCHAR, as it can be text of varying length up to 8000 characters long. It even allows us to set a smaller maximum to suit our needs, which may help us save on how many resources the data can take up. Perfect!

This should be your thought process for any data you store—consider what you're representing, the characters that make up that data, and how large you expect it to be. Read the [list and definitions](#) carefully to find the most suitable option.

Planning a Database and ERDs

Planning a Database

Firstly you'll need to sit down and think about the information you are going to need to store. Once you have a list of this data, see if you can group it into smaller and re-usable groups. This becomes easier over time as you become more familiar with the idea of relational databases, and are exposed to a greater number of common patterns found within them.

After you've listed these groups, start thinking about any relationships that may be present between pieces of data. For instance, consider a list of people and cities. If it is assumed that each person lives in a city, there is a relationship between the list of cities, and the list of people! It is important to identify where information intersects.

Once you've made note of the information, its data types, and any relationships, it is time to begin a formal plan—often in the form of an entity relationship diagram (ERD.)

What is an ERD?

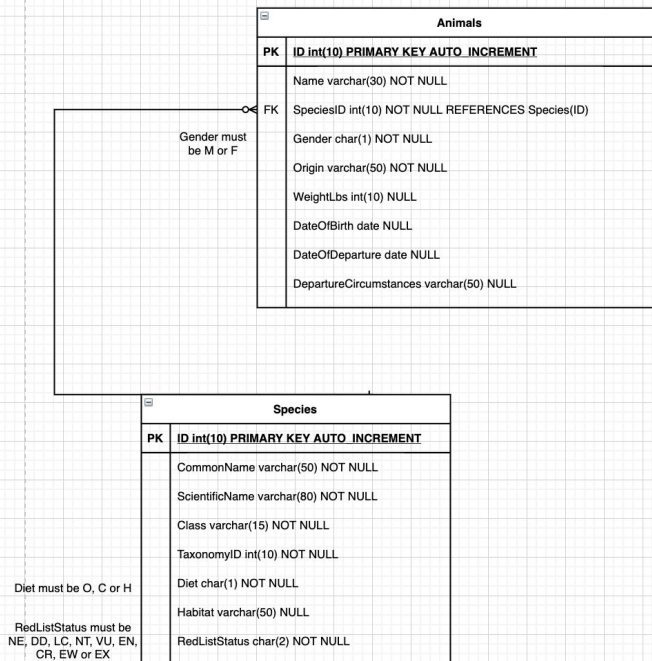
An [ERD](#) (entity-relationship diagram) is a descriptive and visual representation of a database.

Databases can be modelled as either:

- A collection of entities
- Relationships between entities

An ERD is used to communicate the logical structure of a database to users.

It allows us to 'sketch out' database designs and is a graphical tool for modelling data.



Components of an ERD

There are 3 basic elements in an ERD: entity, attribute, relationship. Cardinality and ordinality are two other notations used to help further define relationships.

Entity - An object, usually a noun, like person, place, event, etc. A zoo would have animals and species. A school would have students, courses, teachers, etc.

Attribute - An attribute is a property, trait, or characteristic of an entity, relationship or another attribute. For example, a Person has a Name.

Relationship - Describes how entities interact.

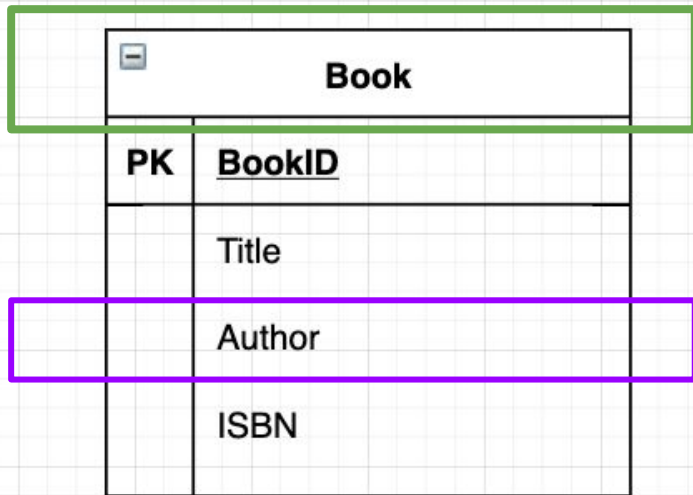
Cardinality/Ordinality - Places the relationship in the context of numbers.

We'll talk more about this in a bit.

What an Entity Looks Like

Entity names should be singular. (i.e. Book and not Books) and a noun (i.e. Book not Read)

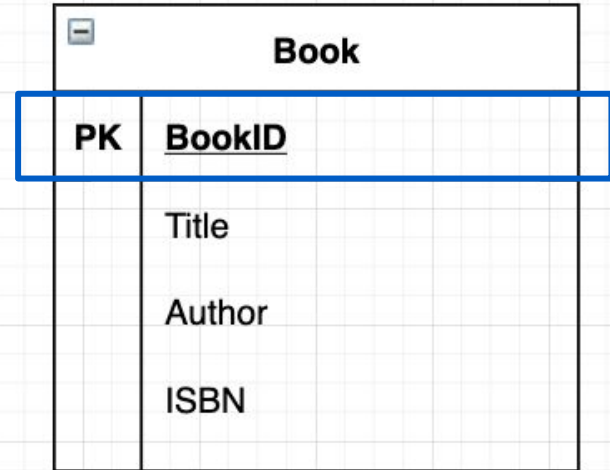
Attribute names must be descriptive. (i.e. Author not Auth or A)



Keys

Every table has a name and a unique way to identify an entry called a **PRIMARY KEY**.

To indicate a primary key add **PK** beside the column name.



Foreign Keys

If there is a relationship between two entities we indicate this with a **FOREIGN KEY**.

To indicate a foreign key add **FK** beside the column name.

| Book | |
|------|-----------------|
| PK | <u>ID</u> |
| FK1 | <u>AuthorId</u> |
| | Title |
| | ISBN |

Relationships (There are 3 Basic Kinds)

1. One to One

Single instance of an entity is associated with a single instance of another entity. E.g. A Person can only have one passport and a passport can only belong to one person.

2. One to Many





A single instance of an entity is associated with more than one instance of another entity. E.g. A customer can place many orders but an order can only belong to one customer.

3. Many to Many

More than one instance of an entity is associated with more than one instance of another entity. E.g. A student can be assigned many projects and a project can be assigned to many students.

How to Represent Relationships in an ERD

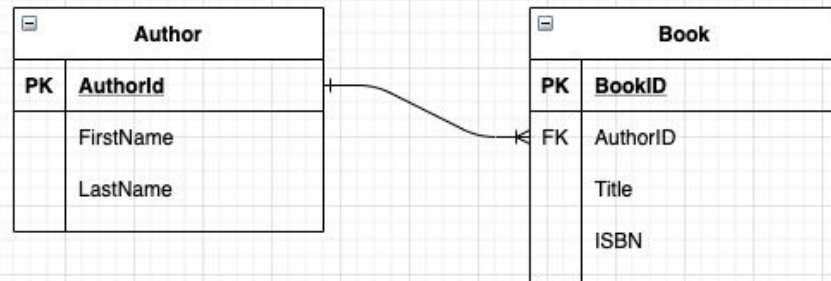
This is known as [Crow's Foot Notation](#). It is used to indicate cardinality (that is, representing a selection of data using a number/relationship instead.) [Click here for some extra reading about Cardinality.](#)

| | |
|--------------|--|
| One |  |
| Zero or one |  |
| Zero or many |  |
| One or many |  |

A Simple Example

Let's say, for example, we wanted to store information about books. Each book has a title, an International Standard Book Number (ISBN,) a title, and an author. Now, because an author may write more than one book, we want to be careful with author!

To repeat the author's name on each stored book record will bloat our database, and make it less efficient. Because authors may be repeated, we should store them *separately*. Then, each author and any information we enter about them, can simply be represented in a book record by a single number (or ID) that is unique to each entered author.



Draw.IO

Draw.IO

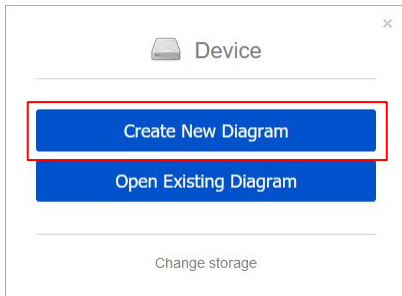
It is alright to sketch out ERDs on grid paper to begin planning out your database, but this can be difficult to keep clean and share with your team or client. When you're happy with your database design, or would like to submit it for feedback, it helps to use appropriate tools that make it not only more legible but more available to the people you're working with.

One such tool is [Draw.IO](#), a website and online diagramming software that happens to provide excellent options for preparing ERDs!

Getting Started

[Open up the website!](#) It may ask you where you'd like to save your projects to, and/or where you'd like to open up existing ones from. There are a variety of options available to help you collaborate! It is up to you what you'd like to experiment with for now, but to keep things simple it may be best to start out with a folder on your computer—click “device” for this option.

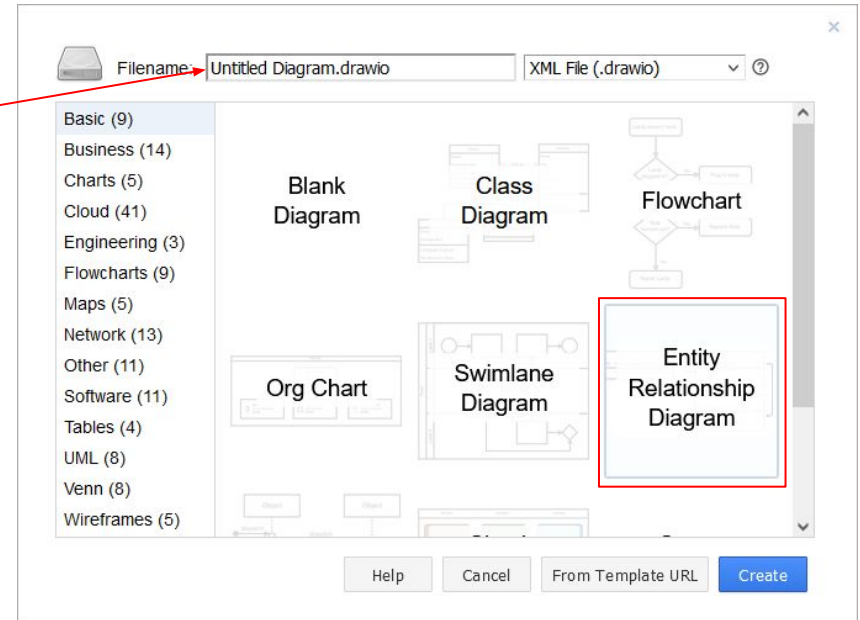
Now it will prompt you to “Create New Diagram,” or “Open Existing Diagram.” To try things out and start a new project, click the former.



Select the Type of Diagram

This website has many different templates to help you get started with all sorts of diagrams, including Entity Relationship Diagrams!

Enter a suitable filename for your project, select “Entity Relationship Diagram” as your type, and click the “Create” button to get started.

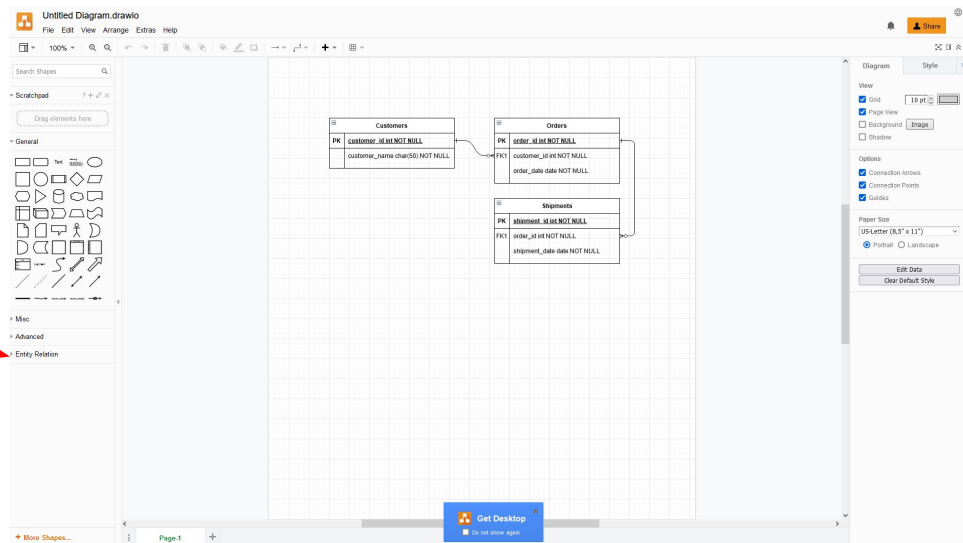


Draw.IO in Action

A small sample project will spin up for you, featuring a little sample database design.

Have a look at what is provided to you—especially in the Entity Relation dropdown, that's where the real goodies are!

Click options in there, drag, drop, and type to your heart's content! Keep in mind the naming conventions and relationships we discussed earlier as you mock up your databases using this method.

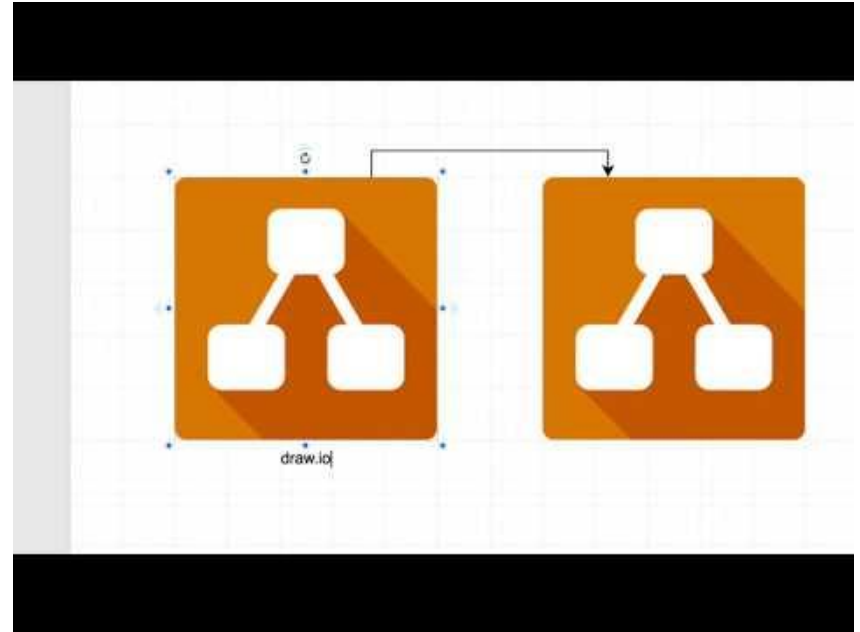


Know Your Tools

Watch the [“Quick Start Tutorial” Draw.IO video](#) to get a quick overview of the tool and how it can be used.

Feeling comfortable? Hoping to increase your work efficiency and speed? Look into the [keyboard shortcuts](#) the application affords to you!

There are also [support options](#) if you run into a snag, as well as a [desktop computer version of Draw.IO](#) for you to check out if you prefer a computer program instead of a website.



Recommended Reading

It's a good idea to read up more on what we've covered today. Have a look at the following:

- [Beaulieu, A. \(March 2020\). Learning SQL, 3rd Edition. O'Reilly Media, Inc.](#)
 - [Chapter 1. A Little Background](#)
 - [Chapter 2. Creating and Populating a Database](#)
 - [MySQL Data Types](#)