An Introductory Guide to Databases

Justin Tojeira, April 1, 2019

In this guide, I'll cover the basics of what every aspiring developer should know about databases:

- Databases and database software
- Relational databases and SQL
- SQL tutorials
- Non-relational databases and NoSQL
- Choosing the right database for a project

Databases and database software:

A database is a collection of data that is organized so that it can be easily accessed, managed, and updated.

A DBMS (Database Management System) is the software used to create and manage your database.

Widely-used DBMSs include:

Oracle	MySQL	SQL Server	PostgreSQL (aka Postgres)
MongoDB	IBM Db2	Microsoft Access	Redis
SQLite	Cassandra	MariaDB	Neo4j

Additionally, cloud database services are available from most major tech companies (Microsoft, Google and Amazon all have both free and paid cloud storage services).

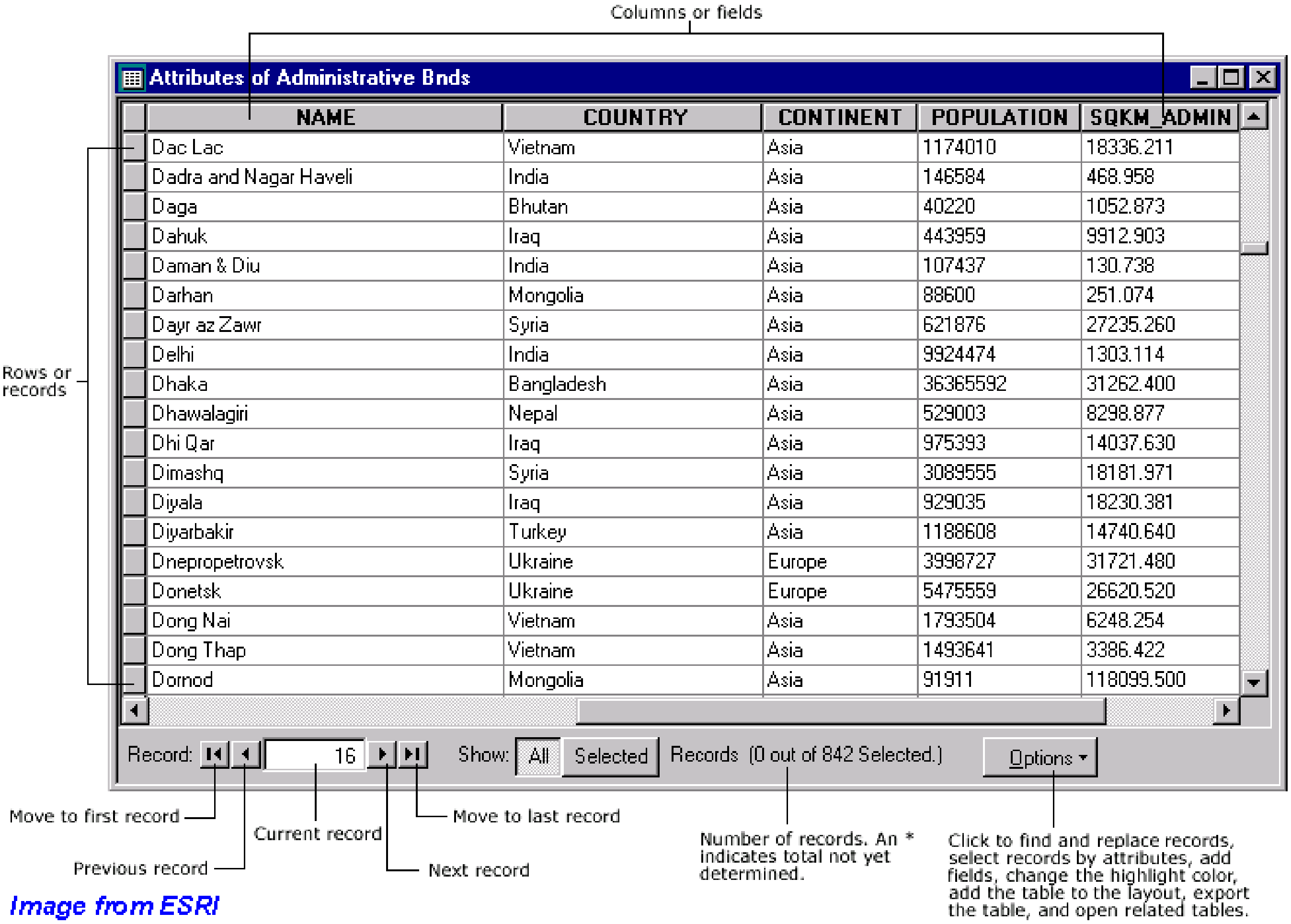
Most, but not all, of these DBMSs are **relational** and use **SQL**. SQL stands for Structured Query Language, and is the standard language used by relational database management systems (RDBMSs) to manage and access the data. Versions of SQL used by different RDBMSs may vary slightly, but in general most RDBMSs are pretty similar to each other.

Relational databases and databases that use SQL are not strictly the same, though there is a huge amount of overlap. While pretty much all relational databases use SQL, some **NoSQL** (i.e. non-relational) databases can also use SQL for some purposes. Thus, while NoSQL originally meant “no SQL”, many sources now refer to it as meaning “not only SQL”, and use “NoSQL” interchangeably with “non-relational”.

Don’t worry if you’re confused. This kind of thing happens all the time in the tech industry. Just keep reading.

Relational databases and SQL:

Relational databases are what most people think of when they think of databases. Relational databases use tables to organize data and use SQL to access the data. Here is an example of a database table:



Columns or fields
Attributes of Administrative Bnds
Rows or records
Record:
Show: All Selected
Records (0 out of 842 Selected.)
Options
Move to first record
Current record
Move to last record
Previous record
Next record
Number of records. An * indicates total not yet determined.
Click to find and replace records, select records by attributes, add fields, change the highlight color, add the table to the layout, export the table, and open related tables.

NAME	COUNTRY	CONTINENT	POPULATION	SQKM_ADMIN
Dac Lac	Vietnam	Asia	1174010	18336.211
Dadra and Nagar Haveli	India	Asia	146584	468.958
Daga	Bhutan	Asia	40220	1052.873
Dahuk	Iraq	Asia	443959	9912.903
Daman & Diu	India	Asia	107437	130.738
Darhan	Mongolia	Asia	88600	251.074
Dayr az Zawr	Syria	Asia	621876	27235.260
Delhi	India	Asia	9924474	1303.114
Dhaka	Bangladesh	Asia	36365592	31262.400
Dhawalagiri	Nepal	Asia	529003	8298.877
Dhi Qar	Iraq	Asia	975393	14037.630
Dimashq	Syria	Asia	3089555	18181.971
Diyala	Iraq	Asia	929035	18230.381
Diyarbakir	Turkey	Asia	1188608	14740.640
Dnepropetrovsk	Ukraine	Europe	3998727	31721.480
Donetsk	Ukraine	Europe	5475559	26620.520
Dong Nai	Vietnam	Asia	1793504	6248.254
Dong Thap	Vietnam	Asia	1493641	3386.422
Dornod	Mongolia	Asia	91911	118099.500

Record: 16

Image from ESRI

Relational databases consist of one or more tables, and each table contains any number of records. A record contains data on a single object, and each record is one row in the table. In the table above, each record describes a geographic area (like a province/county/district).

This is an example of **structured data**, which is what relational databases are best used for. Note that each district has a name, country, continent, population, and sqkm_admin. Furthermore, the name, country, and continent is always text, while the population and sqkm_admin is always a number. So all records contain the exact same types of data.

Contrast this with if you were trying to pick a vacation destination, and had a stack of travel brochures. Some brochures might have pictures of beaches and hotels while others might have lists of cultural events and shows. You would have different kinds of information for each district, and most information wouldn't be easy to describe using a number or a few words. This is known as **unstructured data**, which relational databases are not good for.

You can read more about structured and unstructured data here:
https://www.datamation.com/big-data/structured-vs-unstructured-data.html

And here's an article about a standard set of properties know as ACID that pretty much all RDBMSs have, and that you should have basic knowledge of:
https://database.guide/what-is-acid-in-databases/

SQL Tutorials:

Despite the fact that most of the data out there is unstructured, structured data is easier to actually get useful information from, thus relational databases are still by far the most widely used type of database. And as previously stated, even NoSQL databases can use SQL for some tasks. Because of that, I highly recommend picking one of the following SQL tutorials and putting in a few hours to learn some basic SQL.

https://sqlbolt.com/
Bare bones, concise, gets right to the point, is useful as a basic reference as well.

https://www.w3schools.com/sql/default.asp
Provides a lot of diagrams and examples, and useful as an in-depth reference.

https://www.khanacademy.org/computing/computer-programming/sql
Video examples and simple exercises, explains a lot beyond just writing queries.

https://www.codecademy.com/learn/learn-sql
A tutorial based around guided hands-on work - interactive lessons, projects, quizzes.

Quick Review:
After going through the tutorial, everything on the first page of this list should be familiar:
https://www.kdnuggets.com/2016/07/database-key-terms-explained.html

Non-relational databases and NoSQL:

Unlike relational databases, NoSQL databases use a variety of methods to organize data. Database.guide does a great job of outlining the generally agreed upon 4 categories of NoSQL databases. These 5 short articles will give you a solid understanding of the basics of NoSQL:
https://database.guide/nosql-database-types/
https://database.guide/what-is-a-key-value-database/
https://database.guide/what-is-a-document-store-database/
https://database.guide/what-is-a-column-store-database/
https://database.guide/what-is-a-graph-database/

Choosing the right database for a project:

First, let’s review what you learned so far. If you’ve gone through this guide up to this point, you should be able to read and understand these articles, which discuss the pros and cons of different database types:

https://dzone.com/articles/the-types-of-modern-databases
https://www.infoworld.com/article/3240644/what-is-nosql-nosql-databases-explained.html

Now let's look at some basic considerations you may have when picking a database for a student project, which may be different from the considerations a company has when picking a database. You'll probably be less concerned with things like scalability and possibly security (unless security is included in the focus of your project), and more concerned with things like cost (free, please), how easily you can connect your database to the other technologies you're using, and where/how to host your database.

Here's a video where the narrator discusses some of the issues you might consider when choosing a database. The information he gives is excellent, though he only discusses 4 databases, and is more focused on CAP properties (the video will explain what they are) than anything else.
https://www.youtube.com/watch?v=v5e_PasMdXc

And here's a different tech professional who bases his choice mostly on how his data is best organized:
https://arcentry.com/blog/choosing-a-database-in-2018/

I'm going to add one more thing to consider if you're doing this for a portfolio project: how widely used the database is. Since one of your main goals will be to gain experience with databases, you should use one that a potential employer is likely to care about.

To that end, here's a massive list of DBMSs, ranked by popularity, with in-depth information about all of them:
https://db-engines.com/en/ranking
For an RDBMS, generally speaking your best bets are any of the top 4, possibly MariaDB, or Hive if you're working with Hadoop.

And if you decide to go with a NoSQL database, here's one more article that might help:
https://www.improgrammer.net/most-popular-nosql-database/