SQL and VB aren't the same but it's a common design pattern to use Microsoft SQL Server or Access as a back-end for a .NET application (which would be written in VB, VC#, etc).

I'm assuming since you mention Access, VB, and other Microsoft products that you are indeed referring to Microsoft SQL Server, but you should be aware that there are *many* dialects ("flavors") of SQL out there. If you'd like to know more, just reply and say so and I'll shed some light.

I wouldn't call SQL a programming language *per se* - it is fully featured for use as a programming language in some implementations (I'm not sure if MS SQL Server is one of them or not), but that's just not what it's most commonly used for.

The language used in Microsoft SQL Server is specifically called T-SQL, short for Transact-SQL.

SQL stands for Structured Query Language. It's mostly used to query databases. You can do things like stored procedures, which are indeed programmed subroutines at the database level, but 80% or so of the time you just want simple CRUD (create/read/update/delete a record) operations - so that's what I'll jot down here.

To start off, you need a database. I'm not really sure what's involved in setting up a Microsoft SQL Server instance - every time I've worked with it it's been a database that someone else already setup.

After you have the database, you need a table. You'll want to issue a [CREATE TABLE](http://technet.microsoft.com/en-us/library/ms174979.aspx)[1] statement. The top of that page shows the structure of the statement - don't let that intimidate you, you don't need most of those options to start out. Scroll down far enough and you'll see some simple (and some more complicated) examples. You can use one of those or craft one of your own.

I'll wait.

...

Done? Great, you've got a table now. Fantastic. We can start with those CRUD operations.

The first one, C (create), is an [INSERT](http://technet.microsoft.com/en-us/library/ms174335.aspx)[2] statement in the SQL language, [examples here](http://technet.microsoft.com/en-us/library/dd776381.aspx)[3] . You'll want to craft an INSERT statement that will work against your table. This means that if you created a table with three columns (let's say INT, DATE, and TEXT), you'll do:

INSERT INTO [tablename] (column1\_name, column2\_name, column3\_name) VALUES (1, '2014-04-03 19:49', 'herp derp');

You can also do multiple rows in one statement - I will split it to multiple lines for readability:

INSERT INTO [tablename] (column1\_name, column2\_name, column3\_name) VALUES

(1, '2014-04-03 19:49', 'herp derp'),

(2, '2013-08-16 20:25', 'text #2'),

(3, '2013-12-25 00:00', 'Christmas!');

If you know the *exact* table schema and want to insert to all columns at once (the previous syntax lets you do a subset of the columns and let the others just be the defaults, as defined in the CREATE TABLE statement), you can shorten it to this:

INSERT INTO [tablename] VALUES (1, '2014-04-03 19:49', 'herp derp');

That covers the basics of INSERT statements - you can check out the links for more.

Let's move on to the R, or "read". In SQL, that's a [SELECT statement](http://technet.microsoft.com/en-us/library/dd776381.aspx)[4] . In its most basic form, you can simply do:

SELECT \* FROM [tablename];

This will pull all the columns and all the rows from the table. Usually, that's not what we want - we want to limit it to a specific row or a set rows matching a certain set of criteria. That's where the wonderful WHERE clause comes in:

SELECT \* FROM [tablename] WHERE id = 47;

SELECT \* FROM people WHERE first\_name = 'Herp' AND birthdate = '1990-06-18';

You can get arbitrarily complex with the logic in the WHERE clause, and if your table(s) are keyed right, not run into any awful performance issues. Relational databases are *awesome* at querying.

In addition to limiting the rows, you can limit the columns as well. These are starting to look like a query you're more likely to see in the real world, in a well-architected application:

SELECT column1\_name, column2\_name FROM tablename WHERE column3\_name = desired\_value;

SELECT id FROM people WHERE first\_name = 'Herp' AND birthdate = '1990-06-18';

SELECT first\_name, last\_name, birthdate FROM people WHERE id = 106;

You can also pull from multiple tables in a single query. This is called a join. I don't use T-SQL enough to know its join syntax, so I'm not going to cover it here, but you can read about it in [this article](http://www.made2mentor.com/2011/02/different-types-of-joins/)[5] - I only skimmed that, so hopefully it doesn't suck.

Now, we're onto the U, for update. In SQL this is actually an [UPDATE statement](http://technet.microsoft.com/en-us/library/ms177523.aspx)[6] . These follow the following syntax:

UPDATE [tablename] SET [column1\_name] = column1\_value, [column2\_name] = column2\_value WHERE [column3\_name] = column3\_value;

UPDATE people SET last\_name = 'Maidenname-Marriedname' WHERE id = 46002;

That example might be what you'd issue if a woman you were tracking in your database got married and hyphenated.

Be careful, as you can update multiple rows at once, depending on your WHERE clause - it's often a good idea to do a SELECT \* with your WHERE clause before executing an UPDATE with the same WHERE clause. You can even leave off the WHERE clause and update every row in the table:

UPDATE people SET last\_name = 'Derp';

I've just set *everybody's* last names to Derp because I forgot my WHERE clause.

UPDATE statements can be complicated as well - they accept the exact same set of logic in the WHERE clause as a SELECT statement, and you can update multiple tables simultaneously (joins again), or update one table based on the values of another.

Finally, the D - for delete. In SQL, that's a [DELETE statement](http://technet.microsoft.com/en-us/library/ms189835.aspx)[7] . These, like SELECT/UPDATE statements, accept the same WHERE logic, including none at all. Like UPDATEs, it's a good idea to test your WHERE clause with a SELECT before executing a DELETE.

DELETE FROM [tablename] WHERE [where clause];

DELETE FROM people WHERE id = 16;

DELETE FROM people WHERE birth\_date < '1988-01-01';

Also like SELECTs and UPDATEs, you can execute a DELETE against multiple tables - to delete rows from one or more tables, based on the contents of one or more other tables.

Now that we're all done with our table, we can get rid of it. Simply DROP TABLE:

DROP TABLE people;

This has been a very quick primer on T-SQL and SQL in general. If you're interested I'd invite you to read through some of the links I posted and some of the pages linked from those. If you want to do actual programming in T-SQL and not just use it to manage your data records stored in an application written in a different language (like VB), you'll want to check out the [CREATE PROCEDURE](http://technet.microsoft.com/en-us/library/ms187926.aspx)[8] statement as a starting point.

Hope it helps!