CRUD Databases
(Create, Read, Update and Delete)
Part 1 - Arrays

1 Introduction

A demonstration of **CRUD** database programming, based on using arrays as the database format. CRUD stands for **create, read, update and delete**. These are the four basic operations every database must perform.

The earliest databases used arrays to represent the records in the database. It is probably the simplest and most efficient way to build a database. It does not include any of the overhead involved in a format like **JSON**. If you were creating a database on an embedded computing platform with limited memory, a database with records based on arrays would be a good choice.

All databases incorporate arrays at some point. An array is simply a grouping of **adjacent** memory locations.

For this demo we will create a basic application that manages a list of **newsletter subscribers**. Since this is a demonstration, we will limit the database to a **maximum** of seven records. For simplicity, all the fields in the database records will be **strings** (arrays of characters).

The database will store the following information.

- the **email** address of the subscriber
- the date the subscriber **joined** the list
- the date the subscriber's subscription will **expire**
- whether this person has **member** or **guest** status in the list
- whether or not a **renewal notice** has been sent to this subscriber

At the beginning of the demo, and after every change to the database, the contents of the database will be displayed. The data will be displayed both in **table** format as well as in **literal notation** format. These are actual screenshots of the data as it is displayed by the demo application.

The demo was written using **HTML, CSS, VanillaJS and SVG graphics**, and runs in your Web browser. It has been tested under **Android™, iOS, and Windows using the Chrome, Firefox, iOS Safari and Microsoft Edge browsers**.

The demo is Copyright © 2021 - 2022. All rights reserved.

The video is Copyright © 2021 - 2022. All rights reserved.

2 Database

In order to create the database demo, we need to make some design decisions about how the database is structured.

Since this is a demo, we just make every field be a **string** (array of characters).
For example:

- ***email*** = 35 characters.
- ***first subscribed*** = 19 characters.
- ***expires*** = 19 characters.
- ***member*** or ***guest*** = 19 characters.
- ***renewal notice*** = 3 characters.

So the size of every record would be 95 characters.

There are a couple of ways to handle records based on arrays.

1) ***Fixed length fields***
The fields in every record are the same length. For example, the ***email*** field in every record would always have a length of 35 characters.

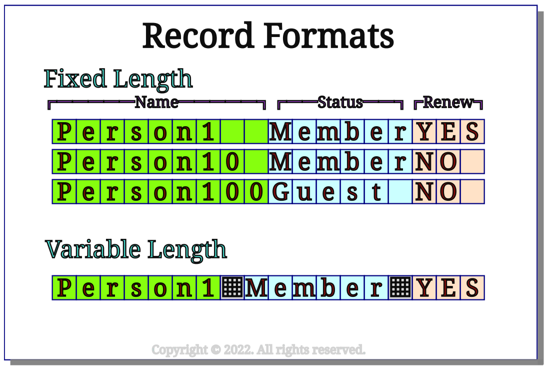


Figure 1: Fixed length versus variable length format.

If the user enters a string that is shorter than the length of the field, we need to ***right pad*** the string (append spaces to the right side of the string) until the length of the string is equal to the length of the field.

One advantage of this is that it simplifies the software code necessary to read the records in the database. When we read the database records, ***every record will be the same length***. To access a particular record we simply calculate its location by multiplying the number of the record by the length of the record; ie, record 0 would start at character 0, record 1 would start at character 95. Record 2 would start at character 190, and so on. We know that we can read one record by reading 95 characters starting at a particular location.

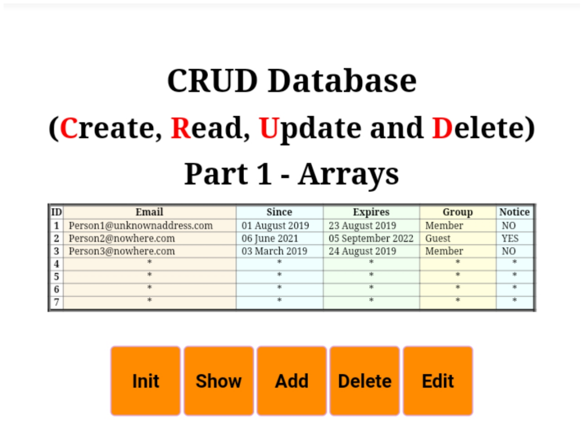
Fixed length fields also simplify accessing the data within a record because we know where the field starts within the record and the length of the field.

2) ***Variable length fields***

Since we don't know how long the fields are, we don't have any idea as to the length of a record. It is necessary to ***parse*** (break up a sentence or group of words into separate components) the data to determine the field and record length. [21, 22] We can insert a character which we know the user will not type to mark the boundaries between the fields. This special character is called a ***delimiter***. We could also insert a different delimiter to mark the boundaries between records. This allows us to parse the records and the fields within the records.

3 Initial database state - Read

This is the demo application ***main*** screen. We display the contents of the database both within a ***table*** view as well as in a ***literal notation*** view. After every change to the database, the contents of the database will be displayed. These are actual screenshots of the data as it is displayed by the demo application.



CRUD Database

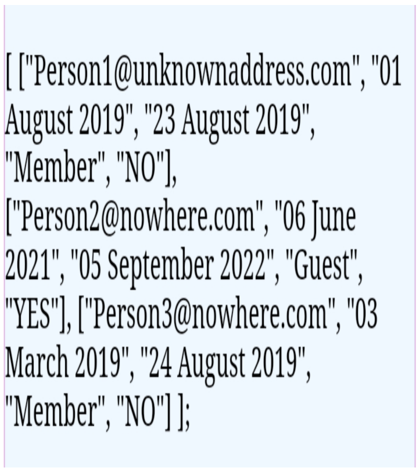
(Create, Read, Update and Delete)

Part 1 - Arrays

ID	Email	Since	Expires	Group	Notice
1	Person1@unknownaddress.com	01 August 2019	23 August 2019	Member	NO
2	Person2@nowhere.com	06 June 2021	05 September 2022	Guest	YES
3	Person3@nowhere.com	03 March 2019	24 August 2019	Member	NO
4	*	*	*	*	*
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*

Figure 2: Table view of the initial database.

The demo starts with ***three*** predefined records. Above the buttons we display the contents of the database within a ***table***. This verifies that we can ***read*** the database records. Since we have limited the database to a ***maximum*** of seven records, there is no problem to display the entire database.



[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"]];

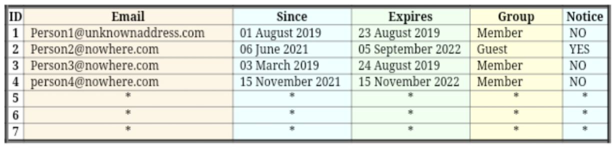
Figure 3: Literal notation view of the initial database.

The screenshot of the demo application's ***literal notation*** view looks like this:

[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"]]

4 Create

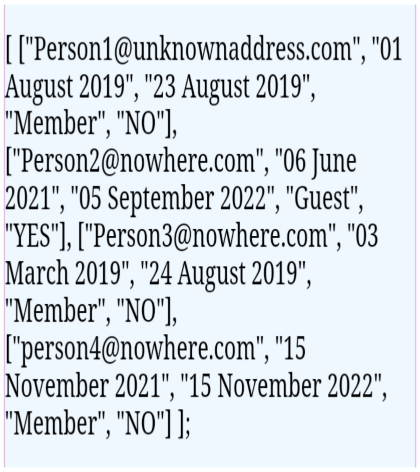
Let's ***add*** a record to the database. (***create***) This will be the ***fourth*** record (record 3 since arrays normally start at 0).



ID	Email	Since	Expires	Group	Notice
1	Person1@unknownaddress.com	01 August 2019	23 August 2019	Member	NO
2	Person2@nowhere.com	06 June 2021	05 September 2022	Guest	YES
3	Person3@nowhere.com	03 March 2019	24 August 2019	Member	NO
4	person4@nowhere.com	15 November 2021	15 November 2022	Member	NO
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*

Figure 4: Table view of the database after adding record.

The screenshot of the demo application's ***table*** view now shows that we have successfully added a record.



```
[ ["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"], ["person4@nowhere.com", "15 November 2021", "15 November 2022", "Member", "NO"] ];
```

Figure 5: Literal notation view of the database after adding record.

The screenshot of the demo application's ***literal notation*** view also shows that we have successfully added a record. It originally looked like this:

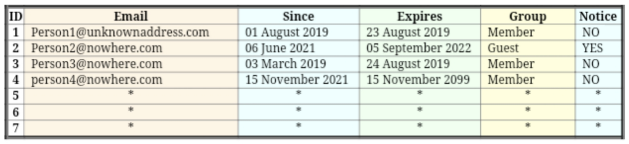
[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"]]

It now looks like this:

[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"], ["person4@nowhere.com", "15 November 2021", "15 November 2022", "Member", "NO"]]

5 Update

Let's ***edit*** a record in the database. (***update***) Let's edit the ***fourth*** record (record 3 since arrays normally start at 0), which is the record we just added to the database. Let's change the ***year*** that the subscription ***expires*** to ***2099***, and ***save*** the record.

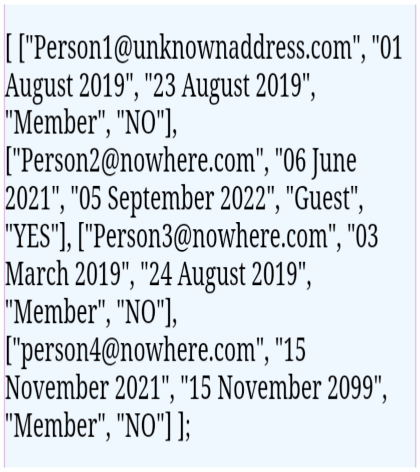


ID	Email	Since	Expires	Group	Notice
1	Person1@unknownaddress.com	01 August 2019	23 August 2019	Member	NO
2	Person2@nowhere.com	06 June 2021	05 September 2022	Guest	YES
3	Person3@nowhere.com	03 March 2019	24 August 2019	Member	NO
4	person4@nowhere.com	15 November 2021	15 November 2099	Member	NO
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*

Figure 6: Table view of the database after updating record.

You can see that the update is now reflected in the application's ***table*** view as well as in the application's ***literal notation*** view.

The screenshot of the demo application's ***table*** view now shows that we have successfully updated a record.



[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"], ["person4@nowhere.com", "15 November 2021", "15 November 2099", "Member", "NO"]];

Figure 7: Literal notation view of the database after updating record.

The screenshot of the demo application's ***literal notation*** view also shows that we have successfully updated a record. It originally looked like this:

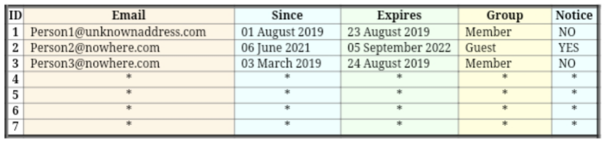
[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"], ["person4@nowhere.com", "15 November 2021", "15 November 2022", "Member", "NO"]]

It now looks like this:

[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"], ["person4@nowhere.com", "15 November 2021", "15 November 2099", "Member", "NO"]]

6 Delete

Let's ***delete*** a record in the database. Let's delete the ***fourth*** record (record 3 since arrays normally start at 0), which is the record we just added to the database.

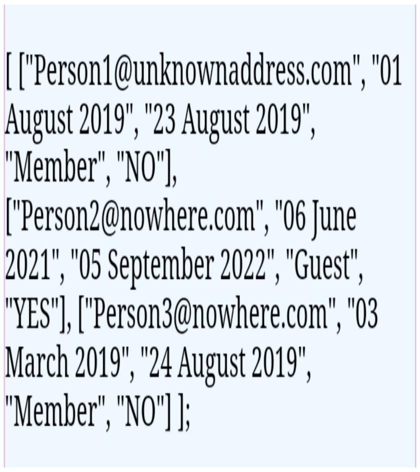


ID	Email	Since	Expires	Group	Notice
1	Person1@unknownaddress.com	01 August 2019	23 August 2019	Member	NO
2	Person2@nowhere.com	06 June 2021	05 September 2022	Guest	YES
3	Person3@nowhere.com	03 March 2019	24 August 2019	Member	NO
4	*	*	*	*	*
5	*	*	*	*	*
6	*	*	*	*	*
7	*	*	*	*	*

Figure 8: Table view of the database after deleting record.

You can see that this database update is now reflected in the database.

The screenshot of the demo application's ***table*** view shows that the database has returned to the ***original state*** it had before we added the ***fourth*** record.



[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"]];

Figure 9: Literal notation view of the database after deleting record.

The screenshot of the demo application's ***literal notation*** view also shows that we have successfully deleted a record. It originally looked like this:

[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"], ["person4@nowhere.com", "15 November 2021", "15 November 2099", "Member", "NO"]]

It now looks like this:

[["Person1@unknownaddress.com", "01 August 2019", "23 August 2019", "Member", "NO"], ["Person2@nowhere.com", "06 June 2021", "05 September 2022", "Guest", "YES"], ["Person3@nowhere.com", "03 March 2019", "24 August 2019", "Member", "NO"]]

References

[1] "HTML 5.2." *W3C*, 14-Dec-2017. [Online]. Available: https://www.w3.org/TR/html52/. [Accessed: 13-Jun-2021].

[2] "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification." *W3C*, 12-Apr-2016. [Online]. Available: https://www.w3.org/TR/CSS22/. [Accessed: 13-Jun-2021].

[3] "Cascading Style Sheets, level 1." *W3C*, 11-Apr-2008. [Online]. Available: https://www.w3.org/TR/REC-CSS1/. [Accessed: 13-Jun-2021].

[4] "CSS Snapshot 2020." *W3C*, 22-Dec-2020. [Online]. Available: https://www.w3.org/TR/CSS/. [Accessed: 13-Jun-2021].

[5] "ECMA-262," *Ecma International*, Jun-2020. [Online]. Available: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/. [Accessed: 13-Jun-2021].

ECMAScript® 2020 language specification
11th edition, June 2020

This is the specification for JavaScript.

[6] "ECMAScript Internationalization API Specification – ECMA-402 Edition 1.0." *Ecma International*, Dec-2012. [Online]. Available: https://402.ecma-international.org/1.0/. [Accessed: 13-Jun-2021].

[7] "Scalable Vector Graphics (SVG) 2." *W3C*, 04-Oct-2018. [Online]. Available: https://www.w3.org/TR/SVG2/. [Accessed: 01-Jun-2021].

[8] "Scalable Vector Graphics (SVG) 1.1 (Second Edition)." *W3C*, 16-Aug-2011. [Online]. Available: https://www.w3.org/TR/SVG11/. [Accessed: 01-Jun-2021].

[9] Mozilla and individual contributors, "SVG: Scalable Vector Graphics | MDN." *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/SVG. [Accessed: 06-Dec-2021].

[10] J. Johnston, "What is a CRUD app and how to build one | Ultimate guide," *Budibase*, 06-Jul-2021. [Online]. Available: https://www.budibase.com/blog/crud-app/. [Accessed: 14-Sep-2021].

[11] "What is CRUD?," *Codecademy*, 2021. [Online]. Available: https://www.codecademy.com/articles/what-is-crud. [Accessed: 14-Sep-2021].

[12] N. Millard, "Why Are You Still Creating CRUD Apis?," *Medium*, 18-Aug-2021. [Online]. Available: https://levelup.gitconnected.com/why-are-you-still-creating-crud-apis-8790ca261bfb. [Accessed: 14-Sep-2021].

[13] V. Kurama, "Generate a CRUD app from any database with one click! | Appsmith." *Appsmith*, 24-Aug-2021. [Online]. Available: https://www.appsmith.com/blog/generate-a-crud-app-from-any-database-with-one-click. [Accessed: 14-Sep-2021].

[14] B. Payton, "What Is a CRUD App?," *DEV Community*, 11-Aug-2021. [Online]. Available: https://dev.to/paytondev/what-is-a-crud-app-4h2e. [Accessed: 14-Sep-2021].

References (continued)

[15] A. Kmetiuk, "Why CRUD Applications are hard?," *The blog of Anatolii Kmetiuk*, 17-Jun-2017. [Online]. Available: http://akmetiuk.com/posts/2017-06-17-crud-apps.html. [Accessed: 14-Sep-2021].

[16] Mozilla and individual contributors, "Array - JavaScript | MDN." *MDN Web Docs*, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array. [Accessed: 29-Oct-2021].

[17] S. Jaiswal, "JavaScript array - javatpoint," *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/javascript-array. [Accessed: 29-Oct-2021].

[18] I. Kantor, "Arrays." *The Modern JavaScript Tutorial*, 13-May-2021. [Online]. Available: https://javascript.info/array. [Accessed: 29-Oct-2021].

[19] "JavaScript Array." *TutorialsTeacher.com*, 2020. [Online]. Available: https://www.tutorialsteacher.com/javascript/javascript-array. [Accessed: 29-Oct-2021].

[20] D. Pavlutin, "Sparse vs Dense Arrays in JavaScript," *Dmitri Pavlutin Blog*, 27-Oct-2021. [Online]. Available: https://dmitripavlutin.com/javascript-sparse-dense-arrays/. [Accessed: 29-Oct-2021].

[21] “Definition of PARSE.” *Merriam-Webster.com Dictionary*. [Online]. Available: https://www.merriam-webster.com/dictionary/parse. [Accessed: 08-Sep-2022].

divide (a sentence) into grammatical parts and identify the parts and their relations to each other

to examine in a minute way : analyze

[22] "What is Parse? - Definition from Techopedia," *Techopedia.com*, 23-Mar-2017. [Online]. Available: http://www.techopedia.com/definition/3853/parse. [Accessed: 08-Sep-2022].

break up a sentence or group of words into separate components

End of document