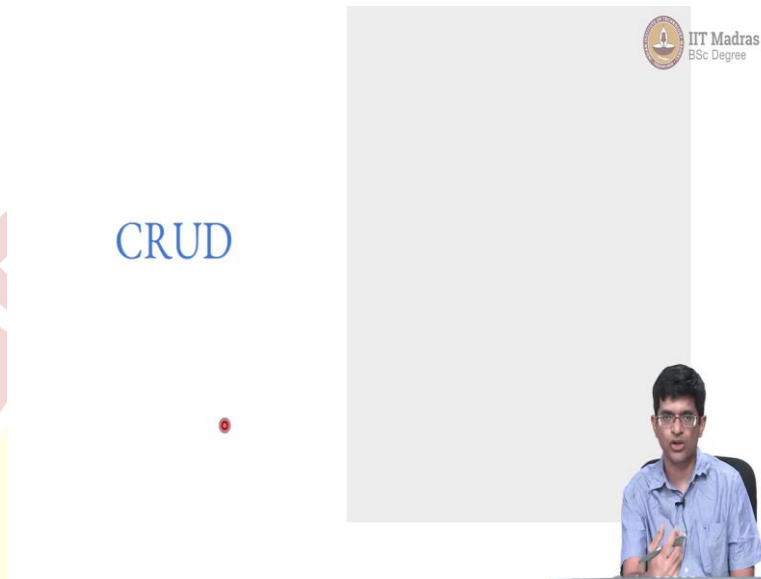


IIT Madras

ONLINE DEGREE

Modern Application Development – I
Professor Nitin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
CRUD

(Refer Slide Time: 00:16)



Hello, everyone, and welcome to this course on Modern Application Development. So, what is CRUD? Does not sound like a particularly nice name, but it has a very specific meaning. And has it is an acronym, obviously, we will see what it stands for in a moment, it is a very useful way once again, of thinking of certain actions that we commonly do.

And once again, at the end of the day, all of this is the result of distilling out years of experience. People find that these are common operations, it is not that you can just prior sit and think of a acronym like this, you realize that whenever you are manipulating data, there are a certain common set of operations that you are doing. And based on that, you say okay, these are the most common ones, this needs to be implemented, no matter what kind of data model I have, let us try and distill that. So, let us try and understand this a bit better.

(Refer Slide Time: 01:03)



Types of operations - Create

- Create a new entry
- Must not already exist
- Check within database to avoid conflicts
- Mention mandatory vs optional fields (name, address, mobile number....)



So, as I was mentioning earlier, you can take the student grade book as sort of running example, but I am going to be talking without referring to a particular example, most of the time. One type of operation that we might want to do with any kind of data is to create a new object of that type. So, create a new entry. Once again going back to the old example, create a new student in the database.

There are certain constraints probably. I mean, when I am creating something new, one of the things must be that it must not already exist. That is pretty clear. Now, I can sort of go into the database and check and make sure that does not happen. In other words, the student does not already exist, to avoid any potential conflicts. Now, in the case of students, it is entirely possible that two people have the same name.

So, there has to be some other way of distinguishing and saying okay, these are two different people. But at the end of the day, the creation of a new object of any type, ultimately, should be one of the most basic ideas that we can think of. While doing this, you might also want to mention certain fields that are mandatory, let us say the name is mandatory. But address may be mandatory, mobile number may be optional. Depends on what you want to do with it. So, the create operation, in other words, is one of the things which I commonly want to do with data of any type where I am trying to construct an underlying model.

(Refer Slide Time: 02:34)



Types of operations - Read

- Get a list of students
- Summarize number of students, age distribution, geographic locations
- Plot histograms of marks



The next thing I would probably want to do is to read, once I have created several entries, I would want to read them, not just read them back one by one, but probably get a list of all the students in the system. Maybe do something like summarizing how many students are there? What is their age distribution? What is their geographic location? Which parts of the country are they sort of concentrated in?

Or, once I have more information, I might want to start plotting histograms of marks. Find out whether there are correlations between how well a person performs in one course versus another course, all of that is reading information from the underlying models. So, we have create and read so far.

(Refer Slide Time: 03:16)



Types of operations - Update

- Change of address
- Update marks
- Change start date of course



The next thing might be to update. So, a student changes their address, go in and modify it, they change their phone number, update that. They took a supplementary exam and did better in one course, change their marks. I offer a course, but then I realized that I need to know delay it by a couple of weeks, change the start date of the course, all kinds of information that might need to be updated, I need to be able to go into the database and make changes to it.

(Refer Slide Time: 03:47)

Types of operations - Delete

- Remove graduated students
- Delete mistaken entries
- Unenroll student from course



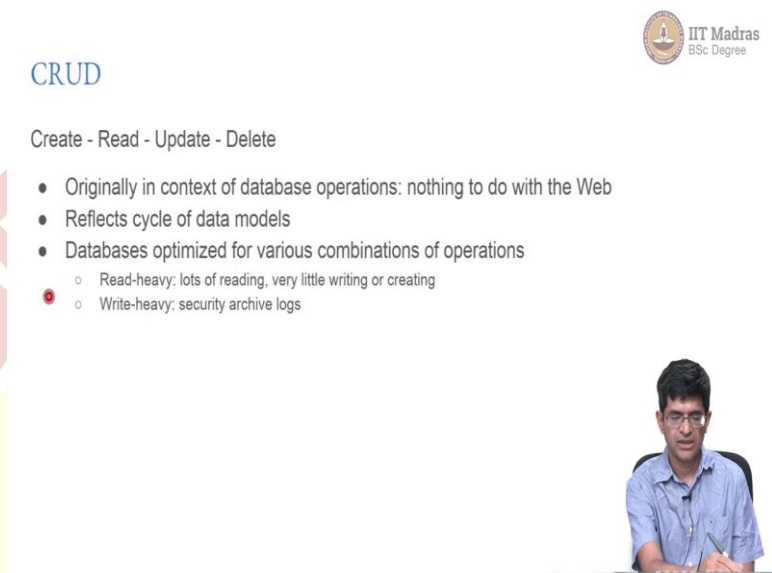
And lastly, I would also want to delete information from the database. So, maybe a graduated student, I remove them from the database. Usually, you would prefer to do something like archiving. But there are sort of good reasons to want to delete them from a table, not from the entire database. But at least from a table.

One example could be that when you start hitting very large numbers of students, then you are going into the lakhs or millions or crores of students, you get to a point where the database is so large that searching through for a given student becomes a problem, it starts becoming a slow operation. So, in most of the cases, at least for the live courses, you know that only let us say a few lakh students are registered.

So, the remaining students you archive them, move them to a different table altogether in the database or possibly even to another database and delete them from the existing model. So, if I go into the live, so I am offering and try to sort of look up what courses I did, it might even tell me that look, you are not currently registered for any. Simply because it has to do some archiving to keep the system in good shape.

So, delete, in other words, is one thing that you can think of. The other simpler option as you know, even simpler possibility is that I made a mistake while entering something, I just want to delete it and create a new entry. And all of those are reasons why you might want to delete an object.

(Refer Slide Time: 05:09)



IIT Madras
BSc Degree

CRUD

Create - Read - Update - Delete

- Originally in context of database operations: nothing to do with the Web
- Reflects cycle of data models
- Databases optimized for various combinations of operations
 - Read-heavy: lots of reading, very little writing or creating
 - Write-heavy: security archive logs

So, that is where CRUD comes from. create, read, update, delete, that is sort of the life cycle of data in any data model, you first just create it, then you are reading what has been created, you might occasionally make a few updates. Finally, you are done with it, you delete it and get rid of it. Now, the important thing to keep in mind over here is originally defined in the context of databases, as such CRUD has nothing to do with the web.

So, please think about that, even though your CRUD API's and CRUD in various different concepts. In context, CRUD was not created for the web, it was sort of Co-opted as a useful way of thinking about what is happening in web applications. But the underlying reason why it is there, it is more of a data model, the lifecycle of a data, rather than something fundamentally to do with the web.

And that sort of shows up in certain ways in what needs to get implemented and so on. And databases themselves could be optimized in different ways. You might, for example, have systems that are read-heavy. There is a lot of reading, but very little writing or creating. So, there is like, let us say a lot of back information about the courses offered on the IITM BSc degree or Swayam online, and you want to collect statistics about it, lots of reading from the database, but not too much new information going in.

On the other hand, let us say that you are just archiving security information for logs, which you need to keep it for the next one year or something like that. That will be heavy, you keep writing the information, but you are never going to go and look at it unless there was an incident unless there was a problem. So, databases need to be optimized in different ways. The CRUD idea essentially helps you to conceptualize what is happening in different places.

(Refer Slide Time: 07:08)

API - Application Programming Interface

- Standardized way to communicate with a server
- Client only needs to know API - not how the server implements the database for example
- CRUD is a good set of functionality for a basic API
 - Usually considered the first level API to implement a web application
- Deals only with the data model life cycle - other control aspects possible



And what is usually done is that, in the context of the web at least, we take a concept like CRUD, which needs to get implemented, and then cast that in the form of an application programming interface or an API. So, an API is something important and we will be talking a little bit more about this as we move further. Ultimately, all that an API is defining is some standard way of communicating with a server in this case.

Actually speaking, an API is not necessarily, need not even have anything to do with the server. I could, for example, define an API, which tells me how a certain library in let us say, the C programming language needs to be used. And all that it says is this is a library to talk to databases to communicate with databases, if you want to use it, you can do this, you make these function calls.

So, that is the programming interface. That is basically where the original term programming interface came from. Application Programming Interface, because you are talking about programming the application in such a way that it can talk to certain libraries. Now, in the case of the web, the same term is used. Because it has a very similar kind of functionality, a similar kind of behaviour. What we are doing is we are defining certain ways by which I can implement an API.

The only differences, in this case, the communication, the API, rather than compiling everything into one program, one C program, which then just links against certain libraries. In this case, the communication is happening over a different protocol altogether. In a C API, what would happen is that you would include some header files, you would then link against the appropriate libraries, and when you run the program.

The communication between your code and the library is happening through the function calls. In the case of the web, the communication between the client end and the server end is happening through the HTTP protocol, that is the main difference. Ultimately, there is always this concept of somebody providing a service, somebody requesting use of that service, and how do they communicate with each other.

So, the good thing, of course, with any API is that the client only needs to know the API itself. They do not need to know how the server implemented the functionality. For example, if it is a database, I really do not care whether it was done using MySQL or Postgres, or SQLite, or noSQL MongoDB, something of that sort, it does not matter. As long as I can make a certain request and the server will give me back the information in the format that I want, that is what the API specifies.

The API specifies how you should request data from the server. And it also specifies what the response from the server will look like. Now, CRUD is a good set of functions, well suited to building a basic API. Because very clearly defined, it sort of tells you this is what needs to be done. And around that, you can then say this is how you can construct an API. You basically tell the user that look, if you want my server to create an object, you send a request like this, if you wanted to read an object, you will send a request like this and so on.

And therefore, CRUD is typically considered the first level API a very common thing that is used in implementing web applications. And remember, of course, that it deals only with the data model life cycle, although there are other control aspects possible in general.