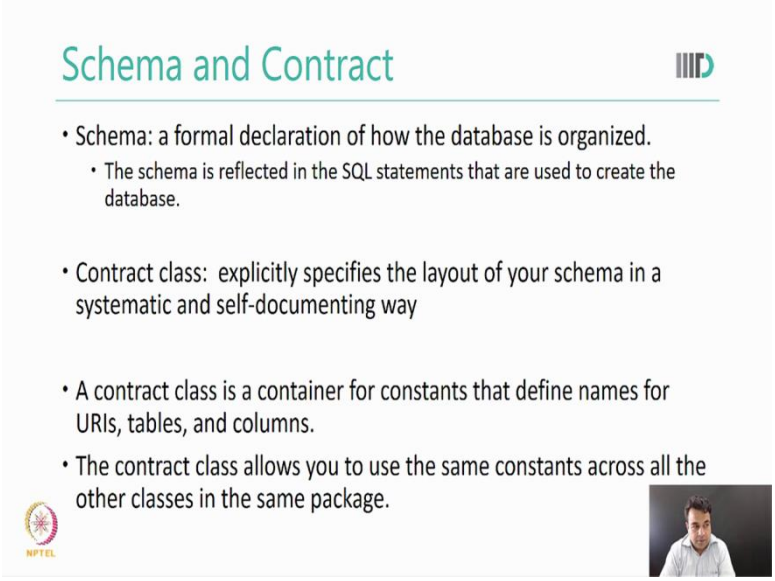
Mobile Computing
Professor Pushpendra Singh
Indraprastha Institute of Information Technology Delhi
Lecture 29
Saving Data in SQL Databases

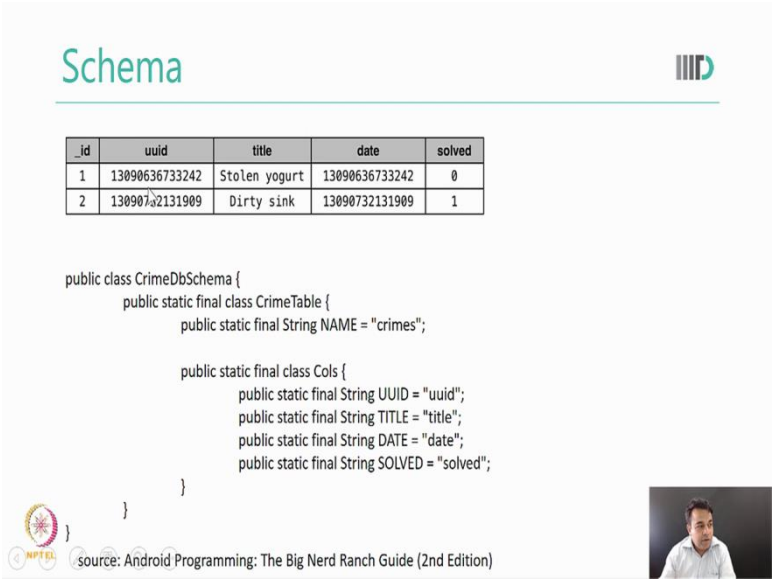
Welcome everyone; in last lecture we saw how to store a very little amount of data using Shared Preferences, then we also saw how to store data using files. Today we will learn how to store data using the SQLite database that is available with android devices and with android operating system. SQLite database is a lite weight database version of the normal SQL databases. So if you are familiar with SQL databases you will find some of that knowledge useful here as well.

(Refer Slide Time: 1:02)



Let us get started as you know the most important thing at the database is schema. Schema is nothing but how a data is organized. So we will see that how to create a good schema and how to reflectthat schema in this SQL statements that we use. So in SQLite database just like SQL databases we have we will have a schema which will be reflected in our SQL statements. And we will also have a contract class, the contract class here explicitly specifies the layout of our schema in a systematic and self-documenting way. So that looking at the contract class can give us a intuition about what kind of schema that we are using. So for example the contract class is used as a container for constants that define names for URIs, tables, columns.

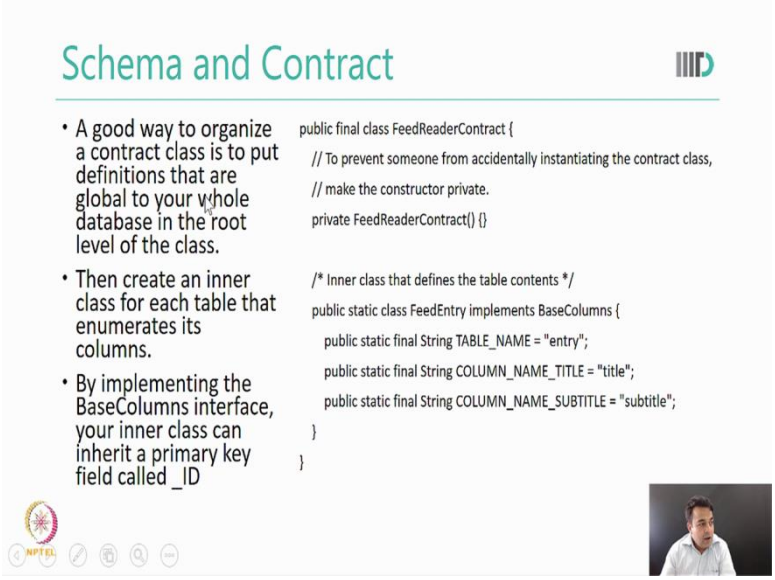
(Refer Slide Time: 2:24)



So we go through these names, we can understand that what kind of tables, columns, Uri are present in our database. And because we are using a contract class we can use the same constants across other classes in this. So here is our simple example, suppose this is the table that we want to store in our program. Now this table has 4 fields, one is a uuid, one is a title, one is a date, one is a solved field. There is also a Id field which we would ideally like to get automatically generated.

Now in this you can see that we have to store this four values uuid, title, date and solved. So for example, this could be our schema where we can say that there is a final String UUID, TITLE, DATE and SOLVED and these are the values that it holds. Now if I look at it I can very easily see that what kinds of calls I am going to have because by just by looking at this I can say that you know in the my column entries would be divided into UUID, TITLE, DATE and SOLVED.

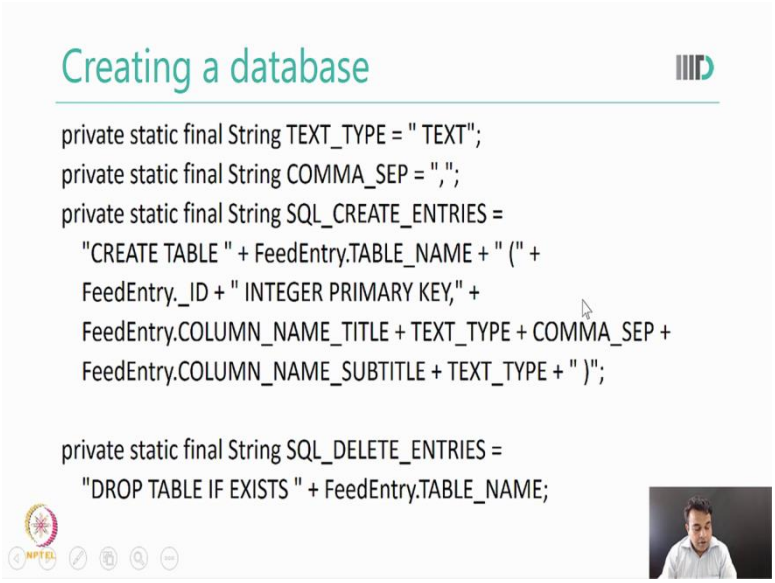
(Refer Slide Time: 3:23)



So a good way to organize a contract class is to put definitions that are global to your whole database in the root level of the class. So once you do this then that is accessible to everything else in your program. And then for each table we create an inner class, so just like here this was one of the tables of our database. And for this one of the table we created an inner class or we can see the example here for one, so this is our top level class and then for each (cla) and then for each table we create an inner class. So just like here. Now what android also provides us that we can implement an interface called BaseColumns and this will automatically give us the primary key field_ID. So as you saw earlier we were ignoring it because we knew that we can get it directly by just implementing the BaseColumns interface.

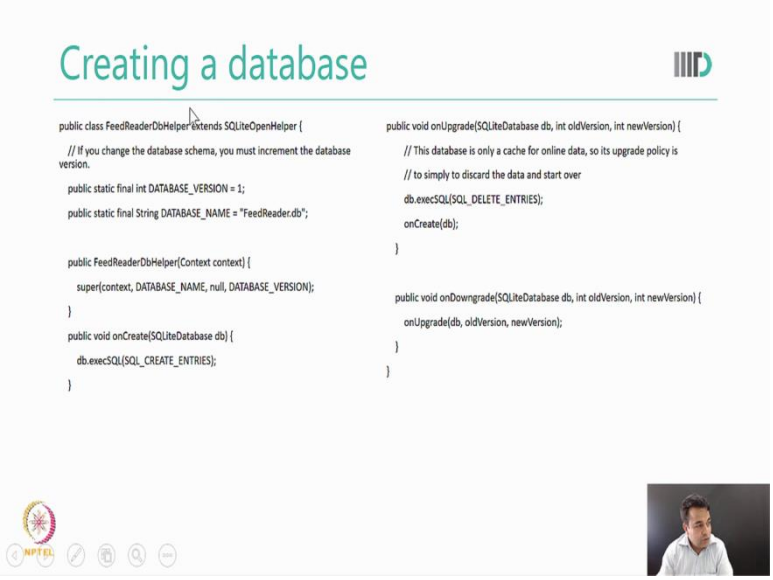
Now you can see that for each table we will be creating an inner class. Try to think that why do we do this, what is the advantage of it and what is the disadvantage of it. Your hint is to think that what is the advantage of having an inner class in java and why do we have inner classes in java. If you can think in that direction you can think what the advantage of this approach is.

(Refer Slide Time: 5:12)



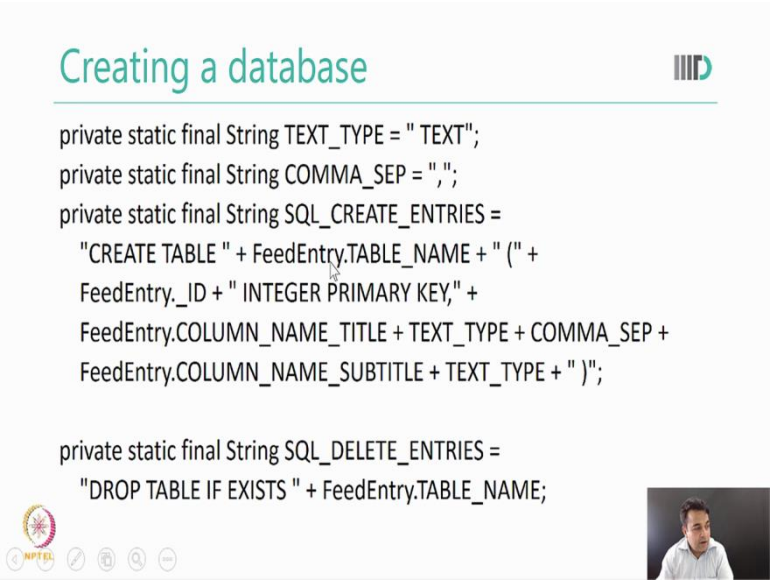
Now let us look at creating a database through some basic example. We will try to create the same database that we are creating that is we would like to have an entry called title and an entry called subtitle. So let us first initialize few things, as you will see in this examples that mainly we are trying to run SQL statements, so that is how our database works. So let us start from the beginning we have a static final String TEXT_TYPE, then a COMMA_SEPARATOR, then a String called SQL_CREATE_ENTRIES in that we are giving commands such as like CREATE TABLE as we know this is a valid command and then we are entering the TABLE_NAME some more values and then we have our COLUMN_NAME_TITLE and COLUMN_NAME_SUBTITLE, which we earlier choose COLUMN_NAME_TITLE and COLUMN_NAME_SUBTITLE.

(Refer Slide Time: 6:26)



And similarly we have the SQL statement for deleting a table which is called dropping a table as we know from our SQL knowledge. Now let us look and that how do we actually create it, so we will go through this code, the only thing that we need to do is to extend few functions. So one of the methods that we need to definitely extend is the onCreate ok. So here again we have a static int DATABASE_VERSION and then we can have a DATABASE_NAME FeedReader here.

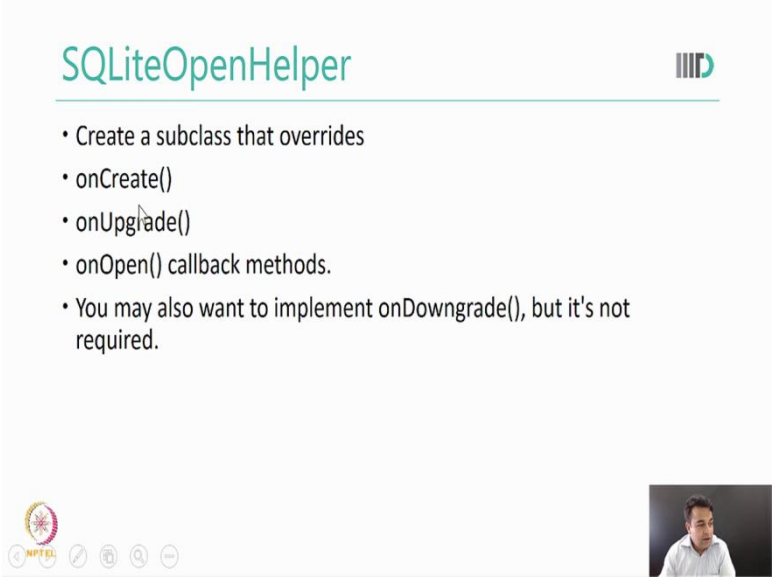
(Refer Slide Time: 7:09)



For example, the FeedReaderDbHelper nothing we need not to do anything. We are only concerned about this method called onCreate and what are we doing in onCreate, we are executing the SQL of this string which we set up in our last slide. SQL create entries so this is

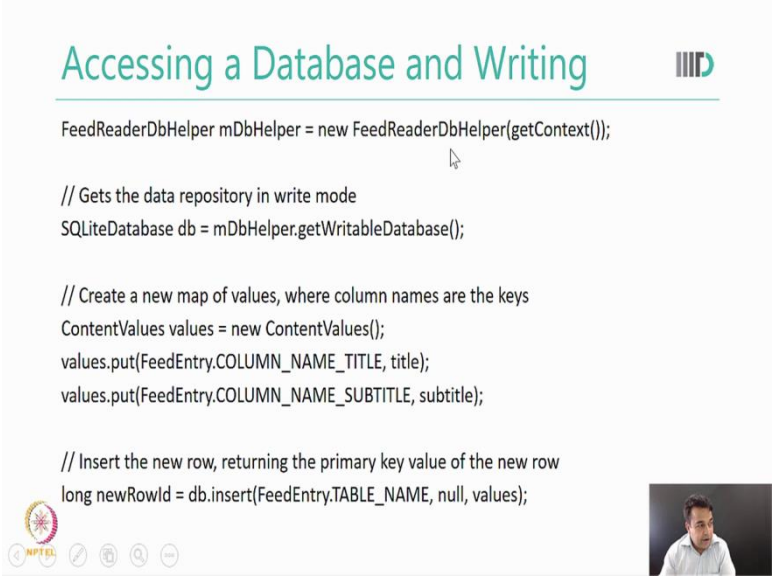
our SQL command that we want to execute on the onCreate. So this SQL command is equivalent to saying CREATE TABLE give the TABLE_NAME we have the TABLE_NAME here and then create the table Entry_ID INTEGER PRIMARY KEY, then gives the COLUMN_NAME_TITLE and SUBTITLE ok.

(Refer Slide Time: 8:12)



So this method when it runs, it runs the SQL command and creates our database. Similarly, we can extend the method called onUpgrade and onDowngrade, which upgrade and downgrade our database as new from our databaseknowledge. So just to revise we create a sub subclass that overrides onCreate(), onUpgrade(), onOpen() callback methods. We just saw the onCreate views here and here our class is FeedReaderDbHelper, which is a subclass of SQLiteOpenHelper and that is it, that is all needed to create a SQLite database in android.

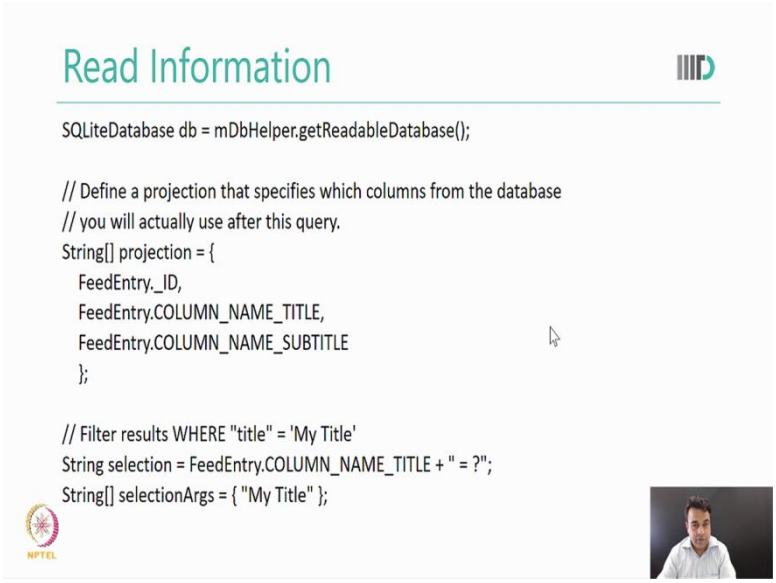
(Refer Slide Time: 8:39)



Now let us look another example of excessing the database and writing to it. So we go to our class object we create a class object FeedReaderDbHelper. We get the data repository in write mode, so from our mDbHelper we get the writableDatabase() now our database is in the write mode. After that what we will do is that we will create a map of values and then we will insert this into our database. So our map of values refers to the each row and then we will be inserting these rows. So as you know we had only two entries TITLE and SUBTITLE, so our map will also have only these two values so we are putting FeedEntry.COLUMN_NAME_TITLE, title and SUBTITLE subtitle. So now we have put these two values and then so we now have a row created separately and we now want to insert this row into our database.

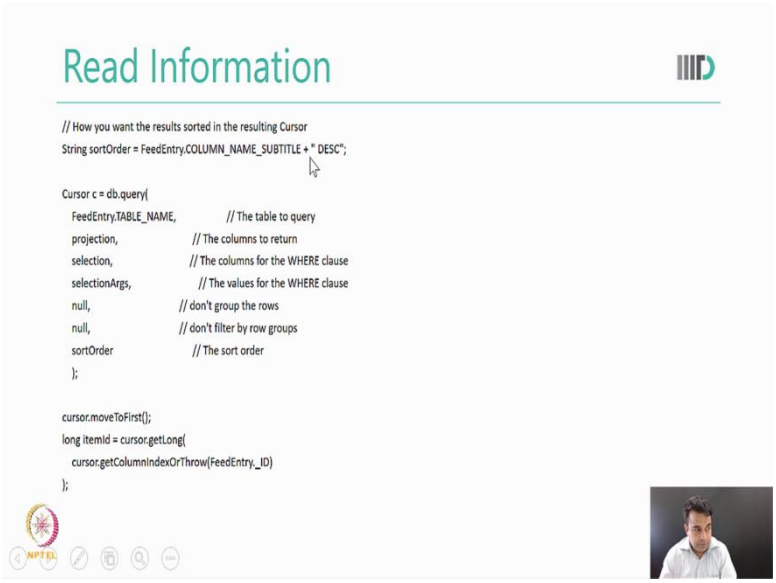
So now we do a simple db.insert in which we give the TABLE_NAME, null means that do nothing if there is if the map is not correct or if the map is not valid and then we insert the map which we created earlier. So essentially what we are doing is that we are inserting a row with values given in the (()) (10:07) map into this TABLE_NAME. Now after this line our database will have a new row with the values entered here. So it is really very simple to create a database and write to it.

(Refer Slide Time: 10:32)



Now suppose we want to read, as you know that in databases whenever we want to read we have to signify that how what we want to read so we have to give criteria of reading. And then we will have to also show that how do we want to see the information that we have just wrote. So for example, to start with we will start with our database we will get a readable version. So last time if you remember we had got the writable, now we are getting a readable. And then we will define a projection that specifies so we define a projection that specifies which column from the database and let us see but this projection will be used only after this query, so what we want we want the TITLE and the SUBTITLE with respect to particular ID.

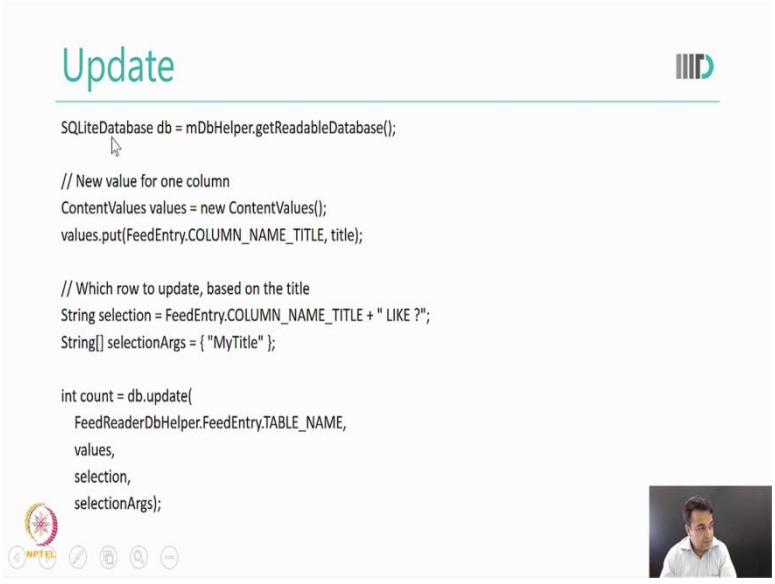
(Refer Slide Time: 11:56)



Now we so our criterion is that we want to get the value of all the rows where the Title value is equivalent to My Title. So that is our WHERE those are few who have done database and I believe that every one of you have done database knows that this is a very simple way to get anything from a SQL database. Then our string selection we simply initialize after that we give an argument where in the next line we define our sorting order and we are saying to return it in the DESCENDING sorting order, then we do the query. So when we do the query we get the result into what we call as a cursor so let us see what is our query. We give the TABLE_NAME that is the table name that we want to query. We get the projection that is the column that we want to get return.

We gives the columns for the WHERE clause and the values for the WHERE clause that is what we described here the WHERE clause, ok, the columns and the values, so here we give the column and here we give the values and then we do not want to group the rows or we do not want to filter by the row groups and then we define the sortOrder that we declared as DESCENDING. Once we get this curser we can start reading from the curser. So now this will contain all the values we have the title is equal to (()) (13:01). So as you can see that it is very easy to create a database read it and write it. Now let us see a very simple way of updating a set SQL SQLite database.

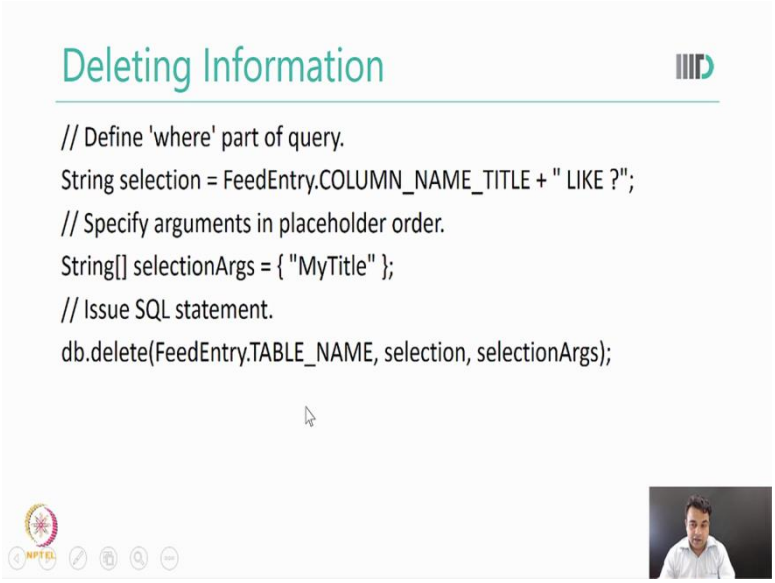
(Refer Slide Time: 13:14)



So updating a SQLite database again is very easy, we get our database instance and we will have to create new values for one column that will be the column that we want to update. So just earlier as in case of writing we are doing to create a value we do a value put and after that we again define the criteria that which column we want to update. So we define a selection

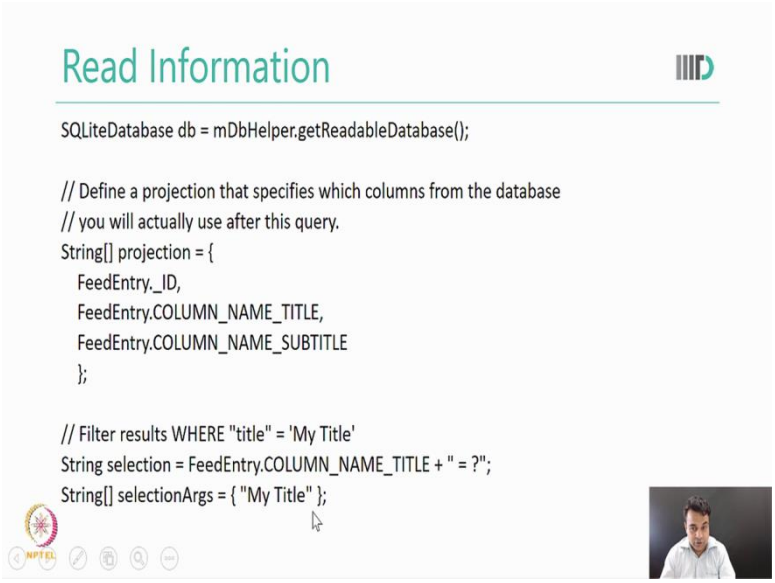
criteria and selection arguments and then we do a db.update where we give the TABLE_NAME, the value that we want to update and the columns that we want to update based on the criteria, so this will update our column.

(Refer Slide Time: 14:04)



If we want to delete this is again very easy we have to just find out the right columns to delete and then we can issue that delete command. Again as you see that the selection, selectionArgs are here given, let us quickly go back in revise how we had been using them.

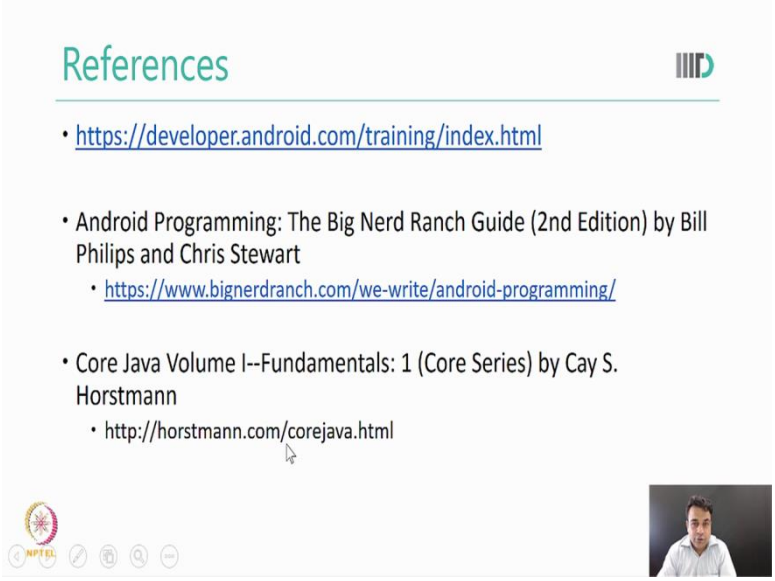
(Refer Slide Time: 14:29)



So we started using them from the read, so the selection was giving us the COLUMN_NAME_TITLE and selection arguments was giving us the values inside that

column. And this is the similar way we want to use it everywhere so when we wanted to read or all the values that are the title was equal to MyTitle we gave it then the read then we wanted to update where title was Mytitle we give it and update and we want to delete where titles MyTitle read of the delete.

(Refer Slide Time: 14:53)



So this was all about using a SQLite database using SQL database or using files is your choice both have their pros and cons, so whether you should read a file or SQLite you should first ask that what kind of data are you going to save. If there is no structure in the data for example, there is no way you can create a table out of your data then it is good to just store a file (()) (15:18). But if there is a structure in the data then it may be good to use the SQLite database, thank you.