Lesson 12 Transcript: SQL PL Stored Procedures

Slide 1: Cover
Welcome to Lesson 12 of the DB2 on Campus lecture series. Today we are going to talk about SQL PL Stored Procedures. My name is Raul Chong and I'm the DB2 on Campus Program Manager.

Slide 2: Agenda
This is the agenda for today.

Slide 3: Stored procedures overview
And we will start with stored procedures overview.

Slide 4: Stored Procedures
So what is a stored procedure? Well. Let's say you have a client application on the left side, and your database server, your DB2 server, on the right side. Now, you can develop an application where you issue your SQL, let's say, SQL number one to go to the server, DB2 server. It does the processing then it sends the result back to the client. Then SQL number two, go to the server, return. And then the client application issues SQL number three goes to the server and returns. And so on, and so on. Right? But the problem here, as you can see, is the traffic on the network. There is too much traffic on the network using this process.

Now what you could do is you could, instead of putting all of the SQL statements on the client application, you can move them and store them as a stored procedure in the DB2 server. So as you can see, we created a stored procedure called myproc where it has all the SQL statements that I want to execute. Then I called the procedure using the CALL statement from the client application. And then I will return, or I will get back the results. And as you can see, there is a lot less traffic, if we have this method. So, stored procedures are very good for performance, especially when you work on the network. And they are also good because all the logic is centralized on the database server. So that potentially there could be another client application that connects to the same database, and that can use the same stored procedure for that other application.

Slide 5: Stored Procedures
Now stored procedures can be created in several languages. They could be created using SQL PL, C/C++, Java, Cobol, CLR, OLE, etc, etc. And the one we are using in these e-learning sessions or the one covered in this particular presentation is going to be SQL PL. And we chose SQL PL because it's the easiest one to learn, it's very easy to learn. It's very good in terms of performance and especially after version 8.2 of DB2 when SQL PL support had, in DB2 it was this part of engine. So before it wasn't like that. So that made performance improvements of using this language a lot better in DB2.

Slide 6: IBM Data Studio
So now let's move on to the next section in this presentation, which is IBM Data Studio.

Slide 7: IBM Data Studio
Now IBM Data Studio is a replacement of the DB2 Developer Workbench of DB2 version 9. And it is a separate image, so it doesn't come with DB2 9.5, but you have to download it separately from the same website which is ibm.com/DB2/express. And it is based on Eclipse,

so when you start IBM Data Studio, it pretty much looks like Eclipse. And it allows you to develop stored procedures, or user-defined functions. You can also do debugging on those, you can develop SQLJ applications. You can do some alterations and dropping of database objects with impact analysis, so for example, if the tables are related, there is referential integrity between them, and then you can do some analysis on that. You could do some web services, you could do some physical data modeling, so basically you can see the relationship between the different tables and objects, etc, etc. So IBM Data Studio is a very promising tool that will cover not only the development of your database but it will basically cover the whole life cycle and also include things that, like I said before, like when you want to alter something on the database, it will also help you with that as well. Now IBM Data Studio can be, I am going to very quickly show you the installation that IBM Data Studio.

Switch to windows explorer
It is very very simple to install. What I have here, I have downloaded the image of IBM Data Studio, which is the one I have here. And I have unzipped it into this directory. So all you need to do is execute setup.exe. Now there is, I am using the Windows version, you can, if you are using Linux, you can download the Linux version. And there are different versions as well in terms of… there is a community edition which is a free version. And there is another version that is fee-based. If you want the community edition, you may not have all the features available, but some of the core or the basic features will be available.

So I am using the community edition and I am going to use, or I am going to install, actually I have it installed. So what I am going to do is just going to walk you through the installation. So I double click on setup, I click on 'Install IBM Data Studio'. And then the IBM Installation Manager will come up. And as you can see here, it is built based on the Eclipse. And it will take a few seconds to install. Now again I already have it installed, so that's why the 'Status' says 'Installed'.

But I am going to go through the steps very quickly. So I click 'Next'. Basically you just choose all the values by default. ok? And then after that, I click Next. You have to accept the license, so I accept the license. Then I choose Next. Then here I am going to, again, because I have it installed, and I am going to put a new package group. So I click Next. Here, you could click Next which is the default or if you already have an existing Eclipse, you can extend that Eclipse by choosing this button here. In my case, I don't, I am just going to install it with all the default values. I click Next. You can choose the language. Click Next. And this is like a features summary. I click Next. And then I have all the summary of the things that I would be installing And then I have to click Install, again, because I already have it installed. I am just going to click on Cancel. But as you can see, the installation is fairly easy to do. Just take all the defaults, it should be good enough to work for this particular course.

Slide 8: SQL PL Stored Procedures basics
Ok, so moving on to the next section. We are going to talk about SQL PL Stored Procedures basics.

Slide 9: Creating your first SQL PL Stored Procedure
So you can create a stored procedure from the Command Editor or from IBM Data Studio. Let me show you very quickly how you can create it from the Command Editor. So from here you first have to be connected to a database. 07:19 In this case, I am already connected to the sample database. 07:21 Why you need to be connected? Because a stored procedure is an object that belongs to, or is a part of database, so you must be connected because you want to

create a stored procedure inside this database. So what I am going to do is I am going to 'create procedure p1', I'll say, 'begin end'. So this is basically creating a stored procedure called p1. And it doesn't do anything, ok? And I am going to execute. But it's just to show you that I can successfully create a stored procedure from the Command Editor. Now let me create a stored procedure from the IBM Data Studio. So let me start IBM Data Studio. I go Start > Programs, and then it's called 'IBM Software Development Platform > IBM Data Studio > IBM Data Studio. So I am going to create it from there. So you see how to do it. And let me just go back here to the presentation.

Slide 10: Basic stored procedure structure
While IBM Data Studio is loading up, let me just give you some very basic description of the structure of SQL PL stored procedures. So first you have to start the command with CREATE PROCEDURE, the name of the procedure and some optional parameters.

Slide 11: Optional stored procedure attributes
The parameters are LANGUAGE SQL, so here if you are using SQL PL you have to use LANGUAGE SQL. If you are using C or Java for example, you have to put LANGUAGE C or LANGUAGE Java, etc etc. RESULT SETS—you need to add this clause or this optional attribute if you are returning a cursor or a result set. We will provide an example of this later on. As for SPECIFIC, that means you are giving a stored procedure a unique name, right?

Remember that in DB2 you can overload the stored procedures. So for example, you could do something like this. Let me just go to the end of this example, you could do **create procedure** and then you can say **p1**, and then you can say, **parameter 1** etc etc. Let's say, parameter 1, parameter 2, all the way to parameter one thousand. Ok? And then you can also create another procedure that has the same name, but only takes one parameter. Ok? Now if you want to drop procedure p1 that has one thousand parameters. The only way that DB2 will know that's the one you want to drop is by issuing a **drop procedure**. And then you have to specify p1 and all the parameters, right? So that DB2 knows that's the procedure that you want to drop and not the other one. And again, this is called overloading and it's just computer science concept. And instead of in this case, dropping this procedure, where you have to specify maybe the entire, all the parameters, you could, if you had provided SPECIFIC as part of the name.

So going back to the presentation, if you use this as SPECIFIC as part of the attributes when you define the procedure, that will give that procedure a unique name. And because you are giving this procedure a unique name, when you want to drop it, you can simply DROP SPECIFIC and then the procedure name that you provided. And that's, so basically that's the one use of this SPECIFIC keyword.

Slide 12: Parameters
Ok, before moving on to parameters, let me just show you now how to work with the stored procedures from IBM Data Studio. So here let's, let me just drop this right now because I want to use the same name. I want to delete this project. Ok, so here we have is IBM Data Studio. Again, this looks very much like Eclipse. Over here, make sure that you have the 'Data' perspective selected. Ok, because there are other perspectives, for example, when I click on this button, you can choose Java perspective. But you need to be on the data perspective to work with stored procedures. Now what we are going to do first is going to create a project, so we go File > New > Project. And we are going to choose 'Data Development Project'. If you don't see it here, you can also choose within Data, choose 'Data

Development Project'. It's the same thing. Click Next. We are going to give a name to this project. And I am going to call it 'DB2onCampus'. Then I click, let's call it 'DB2onCampus2'. Then I am going to click Next. Then in my case, I already have a connection, so I can choose an existing connection. Probably in your case, you may not have a connection, so just click on 'Create a new connection'. Or that may be the only option that you have. Then click Next. Then from here, you will have to choose, already this one is selected, 'DB2 for Linux, UNIX and Windows', All Versions. And then on this side, you can specify the driver that you want to use. Ok? The JDBC driver, this is the database that I want to access so that I want to associate to my project. It's a SAMPLE database. It's using a JDBC connection to the SAMPLE database. So that's why it's asking me for the Host and the Port. In this case, it's, I am using, or my database is on my same server, so I choose localhost. And the port number is 50000 for the instance which is the default value for the port for my instance. And well we talked about this when we we're talking about connectivity in the previous lesson: client connectivity, so you may already know what this means.

Ok, so now that we have this information, I am going to test the connection just in case. And then I am going to put my password for my User ID rfchong, in your case, it maybe another user id, probably will be 'DB2admin' or whatever user you use to install DB2. Then click 'Test Connection'. And I should get a successful connection as it shows in this message. So 'Connection to DB2 for Linux, UNIX and Windows is successful'. Click ok. I continue, I click Next. Then normally what I do here, I just click Finish. That's enough.

And I will have my DB2oncampus2 project created. So I can expand this tree. And then I can choose 'Stored Procedures' because that's what we are talking about in this particular lesson. So I can right-click, I can choose New > Stored procedure, I am going to create a new stored procedure within this project. I can call it 'myproc'. Then I go 'Languages', there is a SQL PL and Java, ok? Whenever it says SQL, it means SQL PL procedure language. And as you can see, you can also create a Java procedure, stored procedure. Now if you recall, you can create stored procedures in many languages, like C/C++, COBOL etc etc. But if you are using that IBM Data Studio tool, you can only create using SQL PL language and Java.
So we are going to use SQL PL. So we select SQL here because, as I said before, this is the easiest to learn and very good in performance. And then the tool, that is IBM Data Studio, is automatically giving me some SQL statements as a template or part of a SQL template. We are going to use this instead of creating my own SQL. And you'll see where this will appear. Ok? So I, you can either click Next here. And work with that exception etc. etc. because this is basically a wizard and it will start building a template for you. I normally preferred to click on Finish. And then I get the editor part where I can start modifying things.

So for example, here you can see, this is the SQL that I mentioned or that was part of the Wizard, right? But normally here is where I preferred to just click Finish because from here I can run my own code. So for example I can declare another variable. I am just adding things so that you can see I can do new things, set x=1. etc. etc. These are just useless variables, but so you see I can make changes. Then what this particular code was doing is it's declaring cursor and it's opening the cursor so that means it's going to return the cursor back to whoever called this procedure.

And we will talk a little bit more about cursors in the next few slides. So now I am going to right-click and choose Save. So I am going to save this procedure. Then I am here, on the left side, I choose a procedure, right-click and then I choose 'Deploy'. Ok? Deploy means that you are basically issuing that "create procedure" statement, so here it's asking me for what

schema I want to use, because if I, for all the objects in the procedure, if I don't provide the schema, then this is the schema, ARFCHONG in this example would be the schema that would used for any of the objects that don't have a schema. Ok? Then I click Finish. And then the Deploy should start. And as you can see from the bottom right side corner, the Deploy was successful. Ok, the Deploy was successful. Now let's say, oh by the way, so this is a tool where you can write your stored procedure, you can do some debugging. And again just think of it as an editor. It's a lot more than an editor, if you think of it as an editor that means the procedure when you write it here, is not residing on the database. The procedure will reside on the database when you issue the Deploy. That means execute this create procedure statement, and then the procedure will be stored on the database that is associated to this particular project.

Ok, now that it was deployed. So now it's on the database. I am going to run this procedure, so I can right-click. Then I choose Run. Ok? When I choose run, what happens? I get the result back and the results are two columns which are basically this: SELECT PROCSCHEMA, PROCNAME. So, basically, I created a cursor and then when I open the cursor (that means return back the cursor to the caller). In this case, the tool itself is the caller. And also make sure that you have DYNANMIC RESULT SETS equals 1, this means I am returning one cursor, which is this one. If I had two cursors, I'll have to put result sets 2. So this shows you how I was able to create stored procedure from IBM Data Studio and deploy it, and then run it. And just, since we are here, I am also going to run this myproc procedure from the Command Editor.

So to call a procedure, we will say **call myproc**. It doesn't take any parameters and I execute. It's connected to the sample database and I get the two same columns back. The first column is a wide column, so that's why you can't see the other one, but that's why I am scrolling to the right and to the left. So it's the same result, also you can do the same thing from command window. But first I have to connect to the sample database, because that's where the database, sorry, that's where the stored procedure is residing, and from here, I do a **DB2 call myproc**, and it also gives me the same result. Ok.

Switch to ppt
Continuing with some basics with respect to stored procedures, if you want to put parameters, you can put or there are three types of input and output parameters. There is an input, which is specified with IN keyword. There is an output parameter specified with OUT. And there is an input and output parameter specified with INOUT. Ok. And all these stored procedures are called using the call statement.

Slide 13: Comments in SQL PL Stored Procedures
Now if you want put some comments on the stored procedure, SQL stored procedure, you can use two dashes as shown in this first example or you can use the slash asterisk, asterisk slash (/* */) which is similar to the C language.

Slide 14: Compound Statement
For statements that are part of the logic of the stored procedure if you put the BEGIN and END, that would be a compound statement. And you can optionally put ATOMIC, which means everything would be treated as one single statement. So that means if for one particular thing fails within this BEGIN ATOMIC, then everything fails. And you have to follow this order, so you have to follow this; it's like a procedural language. So you have to specify all the declarations at the beginning and then the logic.

Slide 15: Variable declaration
To declare a variable is very simple, you just use that declare statement, and you just put declare, the variable name, data type and you can put a default value. If you don't put a default value, the default value is null. So here are some examples, declare the variable name 'temp1', and this is SMALLINT with a default value of zero.

Slide 16: Assignment statements
To assign a value to a variable, you use SET. Like here, SET total equals 100. Or you can also use VALUES(100) INTO the variable total. Here, for example, you can do SET total equals to a select statement. But you have to make sure that this select statement returns only one value because total it's a variable that accepts only one value. If you were to return many many variables, or many many columns or rows; then you probably have to define a cursor. And here is another example where we just setting the variable sch to CURRENT SCHEMA. This is a particular or special register. And there are different special registers in DB2, like current time, current timestamp, current date, etc, etc.

Slide 17: Quicklab: stored procedure with parameters
So here if you like, you can pause this presentation. Look in the book, in the *Getting Started with DB2 Express-C* book, this particular stored procedure, copy paste the stored procedure into either the Command Editor or IBM Data Studio. And execute it and see what it does, right? So you can pause and at the same time, I will show you how this can be done.

Switch to notepad
So I have that same procedure here. And again, you could do it from here, from the Command Editor. And clear the result. And if you are doing from the Command Editor, make sure to change the terminator as I did here. And I explained this in the lessons related to tools where I talked about the Command Editor. As you can see, the create procedure statement is all of these. And the @ symbol means that, the @ symbol is a terminator character for all of these statements. Ok? Now each of these lines or statements within the stored procedure are terminator, are finishing with a semicolon, that's part of the syntax of the create procedure statement. So if I were using a semi-colon here to say that it's the end of my statement, of my entire create procedure statement, there will be a conflict.

Probably DB2 will not know where to end, it will probably end on the first occurrence of these characters. So that's why we need to change this to an '@' symbol. And this would be the same for user-defined functions and also for triggers, and also for inline SQL PL. Ok so we can create a procedure like this if I create, press this button then the procedure will be executed. Ok, now if I want to invoke this procedure, I can say 'call P2', right? And it's taking 3 parameters. So I can say 5 for the first parameter, 7 for the second parameter, and for the third parameter, it is a null parameter. So I have to put a question mark. ok? And I am going to put a question mark and I execute. And again I get that this is my result back. So the first parameter now has a value of twelve. The third parameter now has a value of five. And the first one was an input parameter. And this is just the Return Status for this particular call.

Ok. Let me do the same procedure, but this time I am going to do it from the IBM Data Studio. So I am going to close this one. And I am going to go to the stored procedures, right-click New > Stored Procedure. I am going to, even though it says myproc1, I am going to just finish it here. Ok. Because since I have the code, I can just copy/paste. So here I just copy/paste on top of that. Now because I already created P2 from the Command Editor, I am going to change this to, let's say, P3. Ok, let's change it to P3 as well. Now from here, I can

right-click and choose Save. So now I have my procedure P3 created. And it shows here, my P3, ok? Now, this needs to be refreshed, but this is actually P3. I can right-click and choose deploy. This is just, I am going to just going to click Finish because I want to use that on my schema. I click Finish and now you can see my procedure is correctly deployed. And now if I want to run it, I can right-click and I can choose Run. And here I will be putting the values. I put five, I put six and I press ok. And then well there are no results because what I am doing is actually changing parameter values. When I click on the Parameters tab, I can see these results. Five, six, eleven and then it's five, ok? Now this particular name will be changed if you refresh this. Let's see, you can close the IBM Data Studio and open it again, and this name will be changed to P3. But from here you can see it says P3 already. Ok.

Slide 18: Cursors
Moving on to cursors, I already showed you how to work with cursors. Cursors are basically a result set. Basically it's, you know, when you are working with not only one value, but maybe a bunch of rows and columns, then you would be basically declaring first a cursor, like with this syntax, you say **DECLARE cursor name CURSOR WITH RETURN** to whatever, I'll explain this in a few minutes. And then cursor is based on a Select statement. Ok, then what you do, once you declare the cursor, normally you will OPEN the cursor. Then normally you would create a loop and you would FETCH each row from that cursor within that loop and process the information until you run out of rows. And then in the end, normally you will want to CLOSE a cursor because you want to basically release memory. Now "WITH" is here, WITH RETURN, what that means is you can put there either WITH RETURN to CLIENT or WITH RETURN to CALLER. So what that means is, the cursor can return directly to the client or the caller.

Let me give you an example of that. Let's say you have your client application here. And your client application is calling stored procedure number one. Stored procedure number one is calling stored procedure number two, which is calling stored procedure number three. Ok, now let's say, stored procedure number three is the one that has the cursor, and then if it says **with return to client**, then it means it will return all the way to the client application. If it says **with return to caller**, then it means it just return to SP2 which was the stored procedure that called SP3. So that's basically what client and caller mean.

Slide 19: Quicklab: stored procedure returning result sets
So next you have another quick lab for you to try where you have this other stored procedure and actually I am not going to show you this because it's the same idea. It's a stored procedure that declares a cursor. In this case, we return to client if you are using IBM Data Studio, then the tool IBM Data Studio will be the client itself. And then in this cursor, I am **select**ing these three columns **from staff WHERE salary > 20,000**. And at the end I open the cursor, so that's always the case. When you want to… when you define a cursor and you want to return the cursor to the caller or to the client. At the end of the stored procedure, you will open the cursor and that would mean you return the cursor to the caller or the client. And as I said before, you need to specify DYNAMIC RESULT SETS and the number of cursors or result sets which is pretty much the same thing; in this example that you are returning back. So don't forget it. You have two cursors and you opened the two cursors. You have to say **Result Sets 2**.

Slide 20: SQLCODE and SQLSTATE
Ok moving on. We have now the concepts of SQLCODE and SQLSTATE. These are two special variables. And you need to, even though they are special, you still need to declare them whin an SQL PL stored procedure. And you have to declare as show here SQLSTATE as CHAR(5), SQLCODE is an integer. And every time you perform an SQL operation, select, insert, update, etc etc, the SQLCODE variable would be set. And it would be set to zero if it's a successful operation. If it's greater than zero, means that it's successful with warning; and less than zero and it is unsuccessful; and 100 if there is no data found. Right? So for example, 100 is what you will use, when you working with a cursor. So you are working on a cursor, with a cursor and then you are looping. You are looping on that cursor and you are fetching the rows from that cursor and doing some processing and that loop in order to end that loop, what you could do is you can put a condition where you set that SQLCODE is equal to 100. So you say when SQLCODE equals 100, then the loop ends. Ok, so that's a typical example of when SQLCODE equals 100 will be used.

Slide 21: SQLSTATE and conditions
SQLSTATE provides similar information as SQLCODE, but it's not as particular as SQLCODE. SQLSTATE is normally more a standard across different database vendors, so Oracle SQL may have the same SQL state, but it provides a more generic message, while SQLCODE is more specific, more specific to DB2 in this case.

Now you can also have something called *general conditions*, for example, we have SQLWARNING, SQLEXCEPTION, and NOT FOUND. These are basically… these map to a given SQLSTATE. But rather than using the SQLSTATE, we provide these conditions. Ok? And these are built-in conditions. You will see when I use these conditions in the next… in a few minutes. And then we have a given specific condition which is SQLSTATE with this value, which, this SQLSTATE means there was a truncation, ok? So I could also do the following, I could declare a specific condition for that SQLSTATE. I can say **DECLARE trunk**, it's any name that I want to put **CONDITION FOR SQLSTATE 01004**. So basically a condition is used, basically it helps you when you are doing some SQL PL stored procedure coding, so you are writing your code and rather than testing based on a given SQLSTATE, you can put the condition in there and that will be easier to read.

Slide 22: Condition handling
Let's take a look at condition handling. So what we have here at the bottom is a sample structure of a program. So you have a BEGIN ATOMIC … END, so this is your procedure. And then the 2nd and 3rd line is where you are working with your condition handling. So here you declare your handler. And then on the 3rd line, is where you have your logic to perform that handling operation. Then you have statement one, two and three. That's just your logic for your given stored procedure. So let's say you are working on the statement number one, it was executed, no problem. Then the program continues with statement number two. And then this statement raised an exception. Ok? Because it raised an exception, what's going to happen is the exception will be caught by the handler, which was defined the 2nd line. And it will proceed with the 3rd line which will do whatever logic you put for the handling action. And once it's done, the control will be returned to a different place and it depends on how or what type of handler you define.

If you define a CONTINUE handler, so if you said BEGIN ATOMIC, DECLARE CONTINUE HANDLER FOR etc etc., then after the handling condition , the statement that would follow will be a statement number three. If you define an UNDO handler here, if you

say DECLARE UNDO HANDLER FOR etc. etc., then after the <handler-action>, it will go to END; after the END statement, but it would also undo anything that it performed. And if you use EXIT, so if you say DECLARE EXIT HANDLER FOR etc. etc. Then the same thing, after doing the <handler-action>, it will do END, but it will not undo anything and it will just exit from there, ok? And then you also have here, the DECLARE CONTINUE UNDO or EXIT HANDLER FOR. and here you put the conditions, so you can put for SQLWARNING, for SQLEXCEPTION, for trunc, etc. etc.

SWITCH to Slide – SQLSTATE and conditions
That's conditions we mentioned here, right? SQLWARNING, SQLEXCEPTION, NOT FOUND. And you can put trunc, which is the one that you define yourself for example.

Slide 23: Flow control statements
Ok, moving on to the next slide. Here is a list of flow control statements. I want to talk about these statements because they are very very similar to any procedural language. You have CASE, IF, FOR, WHILE, ITERATE, etc etc. so it's the same thing as other procedural languages and SQL PL is very easy language to learn. I think if you have knowledge of any other procedure language, you can learn this probably in one day or maximum one week.

Slide 24: Dynamic SQL
Ok so moving on with the agenda. We are now going to talk about dynamic SQL.

Slide 25: Dynamic SQL
So what is dynamic SQL? What is the difference between dynamic SQL and static SQL? Well in dynamic SQL, this is for example, here, on the 3rd line where it says **SELECT col1 FROM tabname**. This is an example of dynamic SQL. Because col1, in this example, is a variable, and tabname, as well, is a variable. So basically, what you have here, you have a select statement, but beforehand you don't know what is the column you want to select, and you don't know the table that you want to select from, this will be provided to you at run time. So this is why this is dynamic SQL. So if you have these dynamic SQL in your procedure or your program, DB2 will not be able to determine the access plan ahead of time because it just doesn't know what column you want to get and doesn't know what table you want to get. So this would be done at run time. At run time, DB2 will have to get the access plan that is the best for this particular query and then it will execute the query. So normally you would say that static SQL is, ah, provides a better performance than dynamic SQL because dynamic SQL, the access plan is obtained at run time, while static SQL is done ahead of time.

For example, if this is static SQL, instead of col1, let's say the column first name from the table is 'employee'. So select first name from employee and that was part of my program, then DB2 immediately knows that's the SQL you want to execute, therefore you can do the compilation or binding where the access plan can be obtained ahead of time. And then at run time, you just execute what the access plan, the SQL using that access plan that you already stored on a package. And we talked about this a little bit when we were talking about maintenance and about the rebind command.

Ok, when you use, or when you are with dynamic SQL, normally you have to build a string, like in this example. And you have to use the keywords EXCECUTE IMMEDIATE or PREPARE + EXECUTE. And I will explain this in a few minutes, but you can also use these keywords or these commands with a static SQL. So even though you have a static SQL,

where you know everything on the the column names, you know the table names. You can treat a static SQL as well as dynamic SQL if you prepare them or use these keywords with that query. So that means that particular query, which is static SQL, you are treating as dynamic SQL and the access plan will be obtained at run time instead of binding time or compilation time.

Slide 26: Quicklab: Dynamic SQL
Ok. so what we have here is an example of a dynamic SQL or of a stored procedure using dynamic SQL. And I would suggest you to pause at this time and with this dynamic SQL statement, you first need to create this table T2 as shown. Again, this information is in the book called *Getting Started with DB2 Express-C,* which is a free book that you can download from ibm.com/DB2/express. And it was one of the requisites before starting this e-learning course.

Anyway, from here, you can just copy/paste from the book, which is in PDF format, and then either run it from the Command Editor or from IBM Data Studio. And then you can test it and you will see it is starting to insert some values. Ok, so basically that's dynamic SQL. And you need, like what you are seeing in red, is what is typical from dynamic SQL. What you need to use is a specific or special data type which is STATEMENT. So you DECLARE st STATEMENT, you build your statement. The variable is 'stmt', like this, and then you 'PREPARE st FROM' statement. And then you execute.

So the difference between PREPARE, EXECUTE and EXECUTE IMMEDIATE is EXECUTE IMMEDIATE is in one command you put PREPARE EXECUTE, so EXECUTE IMMEDIATE will prepare, and execute right away, ok? While PREPARE and EXECUTE is prepare, prepare statement first. And then it executes and then you can execute several times. PREPARE means is basically like compiling the statement one time, and then you can execute it several times. EXECUTE IMMEDIATE means you are going to prepare it one time and immediately you are going to execute. Ok, so you will use PREPARE EXECUTE when you are going to be executing the same statement with different values over and over. While use EXECUTE IMMEDIATE, you are going to execute something only one time.

Now here you also have what is called *parameter markers*. Parameter markers are used either in dynamic SQL or you can use them in static SQL. For the following purpose, if you have for example, let me give you an example using this notebook.

Switch to notepad
Here you have a, something similar to the query that you have here: **insert into t2 values** and you put 1 comma 2. Ok, then you have the same statement, but you put 3 comma 5. Etc. etc. ok? When DB2 sees these two statements, because the values are different, then DB2 will assume that these are different statements, completely different statements. So that means that DB2 will take a look at this statement, it will calculate the access plan. And basically it's like compiling this SQL and then next time, then it will continue the processing, it will see this other SQL statement and it will say, ok, this is another SQL statement, so I am going to again compile it or obtain the access plan and so on and so on; but actually, this is the same statement, with different values, and the access plan is probably is going to be the same regardless of the values. So a parameter marker, if I can put a parameter marker, and remove this, basically allows me to or allows DB2 to just calculate the access plan once. And then I can pass different, it will basically use the same access plan no matter what value I am going to be passing to this statement. So that's the purpose of a parameter marker.

Back to ppt

Slide 27: Agenda
Ok, so moving on to the last section of this presentation. How do you call stored procedures? Well I already show you how you can call stored procedures. For example, from the command line or Command Editor, you can use the call statement. Right? And you can think about the command Editor as a Java application. Now if you want to call a stored procedure from your application?

Slide 28: Calling from CLI
It depends on the syntax of that, the programming language which you are using, for example, if you are using CLI, CLI stands for Call Level Interface and it's basically the IBM version of ODBC. Anyway, if you are using CLI, you have to use CLI syntax which is very similar to ODBC. And here, for example, you have to put CALL, the CALL statement. And in this example, MEDIAN_RESULT_SET, that's the name of the stored procedure. And then you have to assign it to this stmt variable and you have to prepare the variable etc etc. Bind the parameter etc etc. So depending on the language you are using, you have to use different syntax. And you can look at more details about this particular example from this directory: `sqllib/samples/sqlproc/resultset.c`. This, I will show you later on where this directory is, but if you install DB2, it will be in this path.

Slide 29: Calling Stored Procedures from a VB.NET Application
If it's your Visual Basic .NET application, then the same thing, you have to declare the name of the procedure As a String. This is the name of the procedure in this example. And then you have to follow basically the syntax of Visual Basic .NET for calling this procedure.

Slide 30: Calling Stored Procedures from a Java Application
The same thing with Java; you know, you have to prepare the call, you have to register the out parameters, so these are out parameters, what data type they have, etc. etc.

Slide 31: Recommended Book
If you would like to learn more about SQL PL stored procedures, either for Linux, UNIX, Windows, i5/OS or z/OS, I suggest you to take a look at this book. The book was written for DB2 version 8.2, but most of the things still apply for DB2 version 9.5. DB2 9.5 the addition has happened to SQL PL is the support for the XML data type.

Slide 32: Sample Applications
And this is what I was talking about before. If you go, if you install DB2, there should be a directory in Windows, but it would be similar in Linux. You go all the way to `SQLLIB/samples`, in this case I am choosing Java. But, as you can see from here, you know, there is .NET examples, C, CLI, COBOL, etc etc. right? So go to this directory and take a look at these files or these programs if you want to learn more about how to code with DB2, not only just stored procedures.

Slide 33: What's Next?
Ok, so with this, you have reached the end of this lesson. So congratulations because you completed Lesson 12 on SQL PL stored procedures. And asking what is next, it will be Lesson 13: User-defined Functions. And also covers inline SQL PL. So thank you very much for listening. Hopefully you will move on to the next lesson. Have a good day.