Stored Procedures

What is a stored procedure?

- A set of SQL statements stored and executed on the database server

Instead of sending
multiple SQL statements

What is a stored procedure?

- A set of SQL statements stored and executed on the database server

- A stored procedure is a set of SQL statements that are stored and executed on the database server. Instead of sending multiple SQL statements from the client to server, you encapsulate them in a stored procedure on the server and send one statement from the client to execute them.

What is a stored procedure?

- A set of SQL statements stored and executed on the database server
  - Write in many different languages
  - Accept information in the form of parameters
  - Return results to the client

- You can write stored procedures in many different languages. For example, for DB2 on cloud and DB2, you can write in SQL, PL, PL/SQL, Java, C, or other languages. They can accept information as parameters, perform, create, read, update, and delete (CRUD) operations, and return results to the client application.

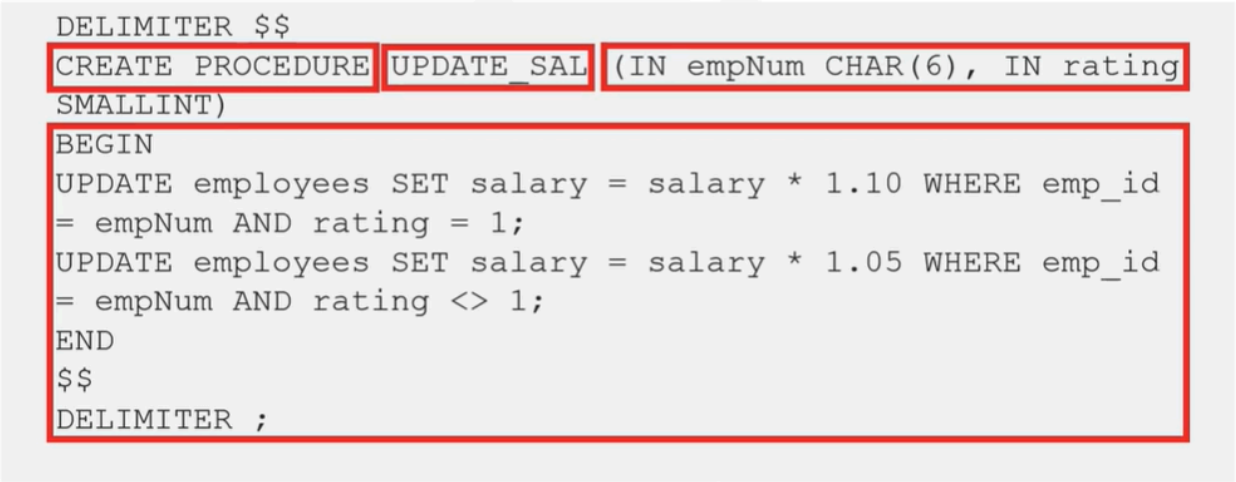
Benefits of stored procedures

- Reduction in network traffic
- Improvement in performance
- Reuse of code
- Increase in security

- The benefits of stored procedures include reduction in network traffic because only one call is needed to execute multiple statements. Improvement in performance because the processing happens on the server where the data is stored with just the final result being passed back to the client. Reuse of code because multiple applications can use the same stored procedure for the same job. Increase in security because A, you do not need to expose all of

your table and column information to client-side developers, and B, you can use server-side logic to validate data before accepting it into the system. Remember though, that SQL is not a fully fledged programming language. You should not try to write all of your business logic in your stored procedures.

CREATE PROCEDURE statement



- Let's look at how to create a stored procedure in SQL. Firstly, you use the create procedure statement, specifying the name of the procedure, and any parameters which it will take. In this example, the update underscore sal procedure will take an employee number and rating which it will use to update the employee's salary by an amount depending on their rating. Then you declare the language you are using. You then enclose your procedural logic and the begin and end statements. In this case, giving employees who have a rating of one, a 10% raise, and all others a 5% raise. Notice that you can use the information passed to the procedure, the parameters, directly in your procedural logic. Also, since the stored procedure is going to use multiple statements, it is prudent to change the delimiter, the character that marks the end of a statement, before we start defining the procedure. Here it has been set to dollar sign dollar sign. Use of this delimiter dollar sign dollar sign in the code will then mark the end of the procedure commands. Finally, we change the delimiter back to semicolon when we are done.

Working with stored procedures

Call from:

- External applications
- Dynamic SQL statements

```
CALL UPDATE_SAL ('E1001', 1)
```

- You can call stored procedures from your external applications or from dynamic SQL statements. To call the update _SAL stored procedure that we just created, you use the call statement with the name of the stored procedure and pass the required parameters. In this case, the employee ID and the rating for that employee.

Summary

In this video, you learned that:

- Stored procedures are a set of SQL statements that execute on the server.
- Stored procedures offer many benefits over sending SQL statements to the server.
- You can use stored procedures in dynamic SQL statements and external applications.

Stored Procedures: A Simple Explanation

In this video, we learned about stored procedures, which are like a recipe for the database. Imagine you have a favorite dish that requires several steps to prepare. Instead of explaining each step every time you want to make it, you write down the entire recipe once. Similarly, a stored procedure is a set of SQL commands saved on the database server that can be executed with a single call. This means you can perform multiple actions in the database without sending each command separately, making it more efficient.

For example, think of a stored procedure as a magic button in a restaurant. When you press it, the chef knows exactly what to do: chop vegetables, cook the meat, and serve the dish. This saves time and reduces confusion. Stored procedures also help keep your data secure and allow different applications to use the same commands without exposing sensitive information.

Is there any other concept within the content that you need an explanation for?

What are the advantages of using stored procedures in applications?

Here are the **advantages of using stored procedures** in applications:

- **Reduced Network Traffic**: Only one call is needed to execute multiple SQL statements, minimizing the amount of data sent over the network.
- **Improved Performance**: Processing occurs on the server where the data is stored, which can lead to faster execution and reduced load on the client.
- **Code Reusability**: Multiple applications can use the same stored procedure, promoting consistency and reducing redundancy in code.

- **Increased Security**: Stored procedures can hide the underlying table structures from client-side developers, and they can include server-side logic to validate data before it is processed.
- **Easier Maintenance**: Changes to the logic can be made in one place (the stored procedure) rather than in multiple application codes, simplifying updates and maintenance.

How can you create a stored procedure in SQL?

To create a stored procedure in SQL, you can follow these steps:

1. **Use the CREATE PROCEDURE Statement**: Start with the `CREATE PROCEDURE` command, followed by the name of the procedure.
2. **Define Parameters**: Specify any parameters the procedure will accept. For example, you might want to pass an employee ID and a rating.
3. **Declare the Language**: Indicate the programming language you are using (e.g., SQL, PL/SQL).
4. **Write the Procedural Logic**: Enclose your SQL statements within `BEGIN` and `END` keywords. This is where you define what the procedure will do.
5. **Change the Delimiter**: If your procedure contains multiple statements, change the delimiter (e.g., to `$$`) to avoid confusion with the default statement terminator.
6. **Call the Procedure**: After creating it, you can call the procedure using the `CALL` statement, passing the required parameters.

Example:

```
DELIMITER $$

CREATE PROCEDURE update_sal(IN emp_id INT, IN emp_rating INT)
BEGIN
```

```
    IF emp_rating = 1 THEN
        UPDATE employees SET salary = salary * 1.10 WHERE id = emp_id;
    ELSE
        UPDATE employees SET salary = salary * 1.05 WHERE id = emp_id;
    END IF;
END $$

DELIMITER ;
```

In this example:

- The procedure `update_sal` takes two parameters: `emp_id` and `emp_rating`.
- It updates the salary based on the employee's rating.

How can you call a stored procedure after creating it?

To call a stored procedure after creating it, you use the `CALL` statement followed by the name of the procedure and the required parameters. Here's how you can do it:

Syntax:

```
CALL procedure_name(parameter1, parameter2, ...);
```

Example:

If you have created a stored procedure named `update_sal` that takes two parameters (employee ID and rating), you would call it like this:

```
CALL update_sal(101, 1);
```

In this example:

- `101` is the employee ID.

- `1` is the rating for that employee.

This command will execute the logic defined in the `update_sal` procedure, updating the salary for the employee with ID `101` based on their rating.