fingerTips

Module-11

Stored Procedures in SQL

1) Stored Procedure in SQL

Stored procedures are precompiled code blocks that are stored in a database management system. They are designed to perform specific tasks or operations on the database, and can be called by other applications or scripts to execute those tasks. Stored procedures can be used to perform complex calculations, manipulate data, or perform other actions that would be difficult or time-consuming to implement directly in an application.

In the context of stored procedures, IN, OUT, and INOUT are used to specify the parameters or arguments that are passed to the procedure. Here's what each of these parameters means:

- IN: An IN parameter is used to pass a value to the stored procedure. This parameter is read-only within the procedure, which means that the value cannot be changed by the procedure. The value passed as an IN parameter can be a literal value, a variable, or an expression.
- OUT: An OUT parameter is used to return a value from the stored procedure. This parameter is write-only within the procedure, which means that the procedure can only assign a value to the parameter, but cannot read the value. The value passed as an OUT parameter must be a variable that is declared before the procedure is called.
- INOUT: An INOUT parameter is used to pass a value to the stored procedure and also return a value from the procedure. This parameter is both readable and writable within the procedure. The value passed as an INOUT parameter must be a variable that is declared before the procedure is called.

So, IN parameters are used to pass values into a stored procedure, OUT parameters are used to return values from a stored procedure, and INOUT parameters are used to both pass values into and return values from a stored procedure. These parameters provide a way for the stored procedure to interact with the calling application or script and perform the necessary

tasks on the database.

2) Creating stored procedure in SQL

Syntax:

```
CREATE PROCEDURE `orders_info`()
select * from orders1;
```

Calling Stored Procedure:

```
call orders_info;
```

3) Create stored procedure showing details of all those orders in which restaurants got rating 5.

```
select *
from orders1
where restaurant_rating=5;
```

Create:

```
Create procedure rating_5 ()
select *
from orders1
where restaurant_rating=5;
```

Calling:

```
call rating_5;
```

4) Create stored procedure showing details of all those orders in which restaurants got rating 5/any other.

Now let's say we want to make a parameter:

IN:

Create:

```
Create procedure rating_ (IN rate int)
select *
```

```
from orders1
where orders1.restaurant_rating=rate;
```

Calling:

```
call rating_(4);
call rating_(2);
```

5) Create stored procedure showing count of orders in which restaurants got rating 5.

OUT:

```
select count(*)
from orders1
where restaurant_rating=5;
```

Create:

```
Create procedure total_ratings (out records int)
select count(*) into records
from orders1
where orders1.restaurant_rating=5;
```

Calling:

```
call total_ratings(@records);
select @records as Total_Ratings;
```

6) Create stored procedure showing count of all those orders in which restaurants got rating 4/any other.

INOUT:

```
select count(*)
from orders1
where restaurant_rating=4;
```

Create:

```
Create procedure total_rate (inout records int, in rate int)
select count(*) into records
from orders1
where orders1.restaurant_rating=rate;
```

Calling:

```
call total_rate(@records, 4);
select @records as Total_rate;
```