

# SQL stored procedure

Server query optimizer

# About project

Instead of sending multiple SQL statements from the client to the server, I encapsulate them in a stored procedure on the server and send one statement from the client end to execute them.

**Benefits:** Stored procedures can be useful if you have an SQL query that you write and execute over and over again. You can save it as a stored procedure, and then just call it to execute it directly in the database server. In stored procedures, you can also pass parameters so that a stored procedure can act based on the passed parameter values.

**Project case study:** I business dealing on pet sales wants to automate the price update of pets on based on the present health status of the pet(good, bad, worse), they could not keep determining this price manually as that would take a lot of time with their database hosted on the cloud(IBM Db2). The task is-How can this business automate the price update of a particular pet given its health status? That is the question this project aims to answer. I created a stored procedure routine named **UPDATE\_SALEPRICE** with parameters **Animal\_ID** and **Animal\_Health**.

- This **UPDATE\_SALEPRICE** routine contains SQL queries to update the sale price of the animals in the PETSale table depending on their health conditions, **BAD** or **WORSE**.
- This procedure routine takes animal ID and health condition as parameters which will be used to update the sale price of animal in the PETSale table by an amount depending on their health condition. Suppose:
- For animal with ID XX having BAD health condition, the sale price will be reduced further by 25%.
- For animal with ID YY having WORSE health condition, the sale price will be reduced further by 50%.
- For animal with ID ZZ having other health condition, the sale price won't change.

# Skills utilised



**Software used: IBM Db2 cloud database**

# Project screenshots and explanations

Next five slides

# Create database

This was what the database looks like in IBM Db2 database instance. The table shown contains list of different pets with their prices and quantity in stock. I retrieved the table with a VIEW call that retrieves all the the data in the original table.

The screenshot shows the IBM Db2 Cloud console interface. The browser address bar displays the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crm%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9...`. The console header indicates "IBM Db2 on Cloud".

On the left sidebar, the "SQL" tab is selected. The "Data objects" section shows a search bar with "Find objects" and a list of objects including "LXQ28194".

The main editor area shows a SQL query in a file named "Untitled ...":

```
1 CALL RETRIEVE_ALL;  
2 CALL UPDATE_SALEPRICE(1, 'BAD'); -- Caller query  
3 CALL RETRIEVE_ALL;
```

Below the editor, the "Results" tab is selected, showing "Result set 1". The results are displayed in a table with 5 columns: ID, ANIMAL, SALEPRICE, SALEDATE, and QUANTITY. The table contains 5 rows of data. A "Filter table" search bar is visible above the table. The total number of rows is indicated as "Total: 5".

ID	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	450.09	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	50.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

# Create Stored Procedure and save in the server

This is screenshot of the stored procedure and the SQL successfully created, ran and stored in the server

IBM Db2 on Cloud

SQL

LXQ28194

REACT...

Untitled - 5

Untitled - 1

Untitled - 2

Untitled ...

1

--#SET TERMINATOR @

2

CREATE PROCEDURE UPDATE\_SALEPRICE (

3

IN Animal\_ID INTEGER, IN Animal\_Health VARCHAR(5) )

4

LANGUAGE SQL

5

MODIFIES SQL DATA

6

BEGIN

7

IF Animal\_Health = 'BAD' THEN

8

UPDATE PETSALE

9

SET SALEPRICE = SALEPRICE - (SALEPRICE \* 0.25)

10

WHERE ID = Animal\_ID;

11

12

ELSEIF Animal\_Health = 'WORSE' THEN

13

UPDATE PETSALE

14

SET SALEPRICE = SALEPRICE - (SALEPRICE \* 0.5)

15

WHERE ID = Animal\_ID;

16

17

ELSE

18

UPDATE PETSALE

19

SET SALEPRICE = SALEPRICE

20

WHERE ID = Animal\_ID;

History

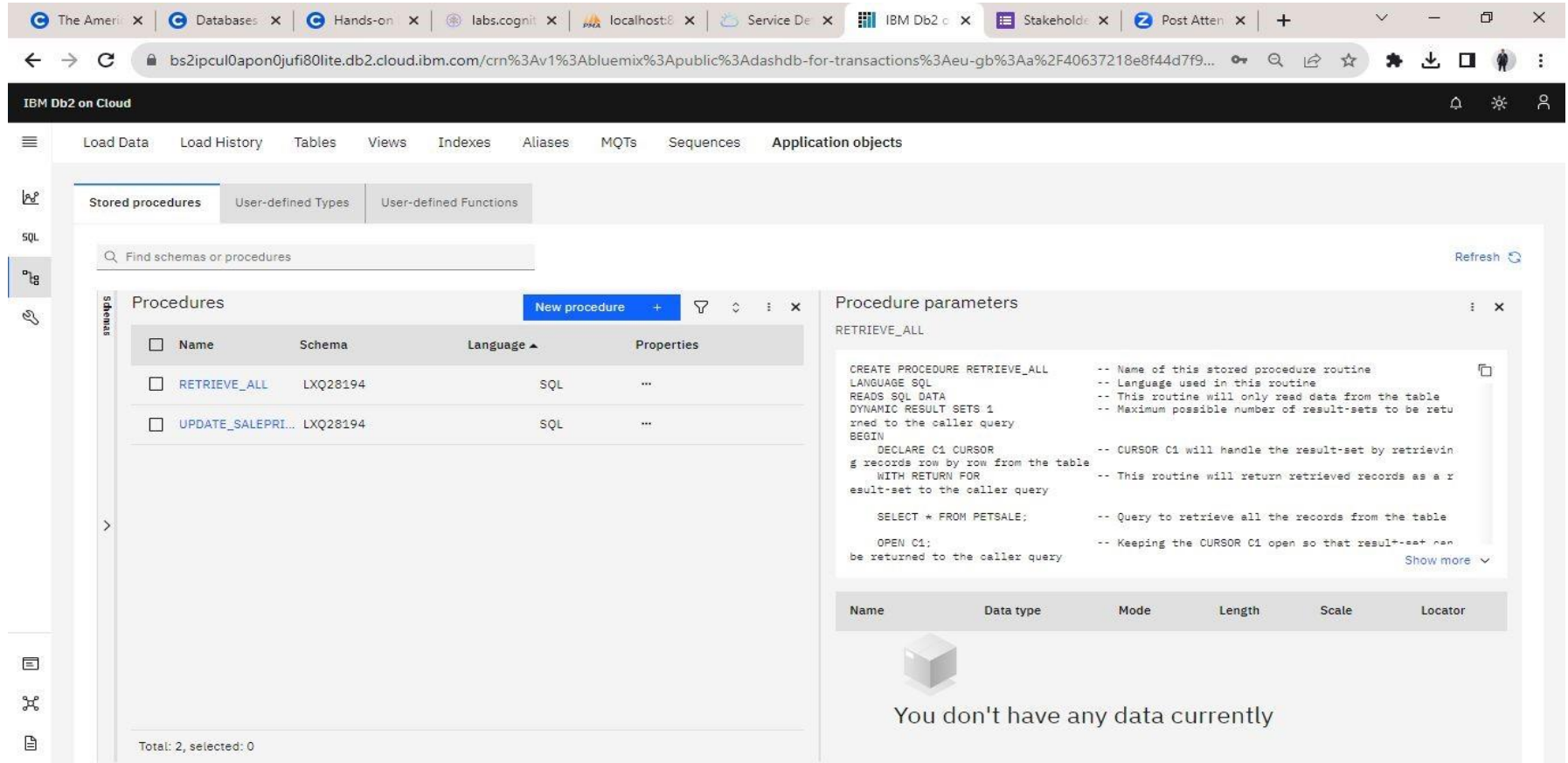
Results

Find by statement or status

Script	Date	Status	Runtime
Untitled - 6	Aug 18, 2023 2:03:40 PM	1	0.071 s
CREATE PROCEDURE UPDATE_SALEPRICE ( IN Animal_ID INTEGER, IN Animal_Health VARCHAR(5) )		1	0.071 s
Untitled - 1	Aug 18, 2023 1:32:32 PM	1	0.189 s
CALL RETRIEVE_ALL		1	0.189 s
Untitled - 5	Aug 18, 2023 1:30:26 PM	1	0.050 s

# Confirm the stored procedure is saved on the server

This screenshot shows on the left hand side that the UPDATE\_SALESPRICE Procedure now exists in the database server and ready to run when called



The screenshot displays the IBM Db2 on Cloud console interface. The top navigation bar includes tabs for 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The 'Stored procedures' tab is selected, showing a list of procedures in the 'LXQ28194' schema. The 'Procedures' table lists two procedures: 'RETRIEVE\_ALL' and 'UPDATE\_SALESPRI...'. The 'Procedure parameters' panel on the right shows the SQL code for the 'RETRIEVE\_ALL' procedure, which is a stored procedure routine that reads SQL data from the 'PETSale' table and returns the results as a result-set.

Name	Schema	Language	Properties
RETRIEVE_ALL	LXQ28194	SQL	...
UPDATE_SALESPRI...	LXQ28194	SQL	...

```
CREATE PROCEDURE RETRIEVE_ALL
LANGUAGE SQL
READS SQL DATA
DYNAMIC RESULT SETS 1
-- Name of this stored procedure routine
-- Language used in this routine
-- This routine will only read data from the table
-- Maximum possible number of result-sets to be returned to the caller query
BEGIN
  DECLARE C1 CURSOR
  -- CURSOR C1 will handle the result-set by retrieving records row by row from the table
  WITH RETURN FOR
  -- This routine will return retrieved records as a result-set to the caller query
  SELECT * FROM PETSale;
  -- Query to retrieve all the records from the table
  OPEN C1;
  -- Keeping the CURSOR C1 open so that result-set can be returned to the caller query
```

Total: 2, selected: 0

# Call stored Procedure for PET ID '1' and health condition 'BAD'

I called the stored procedure with input **parameters 1 and BAD** and the price of **Cat** changed from 450 to 337, returning a 25% decrease in price as stipulated earlier

The screenshot shows the IBM Db2 on Cloud web interface. The browser address bar displays the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9...`. The interface includes a sidebar with navigation icons and a main workspace. The workspace has a top bar with tabs for 'Data objects' and 'Saved objects'. Below this is a search bar labeled 'Find objects' and a list of objects, including 'LXQ28194'. The main workspace is divided into two sections: a top section for writing SQL queries and a bottom section for viewing results. The SQL editor contains the following code:

```
1 CALL RETRIEVE_ALL;  
2 CALL UPDATE_SALEPRICE(1, 'BAD'); -- Caller query  
3 CALL RETRIEVE_ALL;
```

The bottom section shows the 'Results' tab with 'Result set 1' displayed. The results are shown in a table with 5 columns: ID, ANIMAL, SALEPRICE, SALEDATE, and QUANTITY. The table contains 5 rows of data. The total number of rows is 5.

ID	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	337.56	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	50.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24



# Call stored Procedure for PET ID '3' and health condition 'WORSE'

I called the stored procedure with input **parameters 3 and WORSE** and the price of **Parrot** changed from 50 to 25, returning a 50% decrease in price as stipulated.

The screenshot shows the IBM Db2 Cloud web interface. The browser address bar displays the URL: `bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F40637218e8f44d7f9...`. The interface includes a sidebar with navigation icons and a main workspace. The workspace has a top bar with tabs for 'Data objects' and 'Saved objects'. Below this is a search bar labeled 'Find objects'. The main area is divided into two panes. The left pane shows a list of objects, including 'LXQ28194'. The right pane is the SQL editor, showing a query with three lines: `1 CALL RETRIEVE_ALL;`, `2 CALL UPDATE_SALEPRICE(3, 'WORSE'); -- Caller query`, and `3 CALL RETRIEVE_ALL;`. Below the editor is a 'Results' tab, which is active. It shows a table with 5 rows and 5 columns: ID, ANIMAL, SALEPRICE, SALEDATE, and QUANTITY. The table data is as follows:

ID	ANIMAL	SALEPRICE	SALEDATE	QUANTITY
1	Cat	337.56	2018-05-29	9
2	Dog	666.66	2018-06-01	3
3	Parrot	25.00	2018-06-04	2
4	Hamster	60.60	2018-06-11	6
5	Goldfish	48.48	2018-06-14	24

The 'Results' tab also includes a 'Filter table' input, a 'Total: 5' indicator, and icons for filtering, saving, and refreshing the results.

# Conclusion

**Stored procedure is a key server query optimisation method for increased transaction speed and database performance, enabling a high degree of security and automation and reducing query redundancy. With this, businesses can automate inventory updates faster.**



# Contact

---

**Christian Nzeanorue**

[christian.nzeanorue@gwu.edu](mailto:christian.nzeanorue@gwu.edu)

+1 (240)-337-3535

