

Stored Procedures

IT703

Database

Design and

Implementation

Advanced SQL-2

Stored Procedures

- Named collection of procedural and SQL statements
- Advantages
 - Reduce network traffic and increase performance
 - Stored at the server (no transmission of individual SQL statements over the network)
 - Reduce code duplication by means of code isolation and code sharing
 - Are called by application programs (minimizing the chance of errors and the cost of application development and maintenance)

Stored Procedures

- Basic Syntax

```
CREATE [OR ALTER] PROCEDURE [OR PROC] procedure_name  
AS  
BEGIN  
PL/SQL instructions;  
END
```

- EXEC *procedure_name*; command ◦ Execute a stored procedure
DROP *procedure_name* command ◦ Delete a stored procedure

Stored Procedures

- Use the sql file from BB (IT703Lec7_SQLSCRIPT)
- Example:
 - Create a stored procedure to assign an additional 5 percent discount for all products when the quantity on hand is more than or equal to twice the minimum quantity.
 - Revise the stored procedure to adjust the discount rate with input values
 - Create a stored procedure to add a customer

Stored Procedures

- Create a stored procedure to add 5 % discount for products ($P_QOH \geq P_MIN * 2$)

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows the SQL script for creating and executing the stored procedure. The bottom-left pane shows the results of the initial query, and the bottom-right pane shows the results after the stored procedure is executed.

SQL Script:

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = 'PRC_PROD_DISCOUNT' AND type = 'P')
DROP PROCEDURE PRC_PROD_DISCOUNT
GO
CREATE PROCEDURE PRC_PROD_DISCOUNT
AS
BEGIN
    UPDATE PRODUCT
    SET P_DISCOUNT = P_DISCOUNT + .05
    WHERE P_QOH >= P_MIN * 2;
    PRINT('* * Update finished * *')
END
```

Initial Query Results:

P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_DISCOUNT
1	11QER/31 Power painter, 15 psi., 3-nozzle	29	5	0
2	13-Q2/P2 7.25-in. pwr. saw blade	32	15	0.05
3	14-Q1/L3 9.00-in. pwr. saw blade	18	12	0
4	1546-QQ2 Hrd. cloth, 1/4-in., 2x50	15	8	0
5	1558-QW1 Hrd. cloth, 1/2-in., 3x50	23	5	0
6	2232-QTY B&D insaw 12-in. blade	8	5	0.05

Execution Command:

```
EXEC PRC_PROD_DISCOUNT;
```

Results After Execution:

P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_DISCOUNT
1	11QER/31 Power painter, 15 psi., 3-nozzle	29	5	0.05
2	13-Q2/P2 7.25-in. pwr. saw blade	32	15	0.1
3	14-Q1/L3 9.00-in. pwr. saw blade	18	12	0
4	1546-QQ2 Hrd. cloth, 1/4-in., 2x50	15	8	0
5	1558-QW1 Hrd. cloth, 1/2-in., 3x50	23	5	0.05
6	2232-QTY B&D insaw 12-in. blade	8	5	0.05

Messages:

```
Command(s) completed successfully.
```

Stored Procedures

- Revise the stored procedure to adjust the rate with inputs ($P_QOH \geq P_MIN * 2$)

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Results' pane shows the output of a query. The query is: `SELECT P_CODE, P_DESCRIPT, P_QOH, P_MIN, P_DISCOUNT FROM PRODUCT`. The results are as follows:

P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_DISCOUNT
11QER/31	Power painter, 15 psi., 3-nozzle	29	5	0.05
13-Q2/P2	7.25-in. pwr. saw blade	32	15	0.1
14-Q1/L3	9.00-in. pwr. saw blade	18	12	0
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15	8	0
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23	5	0.05
2222-QTV	8 1/2-in. blade	0	8	0.05

Below this, the 'Messages' pane shows the execution of the stored procedure: `EXEC PRC_PROD_DISCOUNT 0.5;`. The results of this execution are shown in the 'Results' pane below:

P_CODE	P_DESCRIPT	P_QOH	P_MIN	P_DISCOUNT
11QER/31	Power painter, 15 psi., 3-nozzle	29	5	0.55
13-Q2/P2	7.25-in. pwr. saw blade	32	15	0.6
14-Q1/L3	9.00-in. pwr. saw blade	18	12	0
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15	8	0
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23	5	0.55
2222-QTV	8 1/2-in. blade	0	8	0.05

On the right, the 'Messages' pane shows the command(s) completed successfully.

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = 'PRC_PROD_DISCOUNT' AND type = 'P')
DROP PROCEDURE PRC_PROD_DISCOUNT
GO
CREATE PROCEDURE PRC_PROD_DISCOUNT
@WPI NUMERIC(3,2)
AS
BEGIN
    IF (@WPI <= 0) OR (@WPI >= 1) -- validate WPI parameter
    BEGIN
        PRINT('Error: Value must be greater than 0 and less than 1')
    END
    ELSE
        -- if value is greater than 0 and less than 1
        UPDATE PRODUCT
        SET P_DISCOUNT = P_DISCOUNT + @WPI
        WHERE P_QOH >= P_MIN*2;
        PRINT ('* * Update finished * *')
END
```

Stored Procedures

- Create a stored procedure to add a customer

The screenshot displays the SQL Server Enterprise Manager interface with two query windows and a messages window.

Top Query Window: Contains the SQL command `SELECT * FROM CUSTOMER;`. The results pane shows a table with 4 rows and 6 columns: CUS_CODE, CUS_LNAME, CUS_FNAME, CUS_IN..., CUS_AR..., and CUS_PHONE.

	CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_IN...	CUS_AR...	CUS_PHONE
7	10016	Brown	James	G	615	297-1228
8	10017	Williams	George		615	290-2556
9	10018	Farriss	Anne	G	713	382-7185
10	10019	Smith	Olette	K	615	297-3809

Bottom Query Window: Contains the SQL command `EXEC PRC_CUS_ADD 10023, 'Walker', 'Johnie', NULL, '615';` followed by `SELECT * FROM CUSTOMER;`. The results pane shows the same table as above, but with an additional row (11) for the newly added customer.

	CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_IN...	CUS_AR...	CUS_PHONE
8	10017	Williams	George		615	290-2556
9	10018	Farriss	Anne	G	713	382-7185
10	10019	Smith	Olette	K	615	297-3809
11	10023	Walker	Johnie	NULL	615	

Right Query Window: Contains the SQL code for creating the stored procedure `PRC_CUS_ADD`.

```
IF EXISTS (SELECT name FROM sysobjects
WHERE name = 'PRC_CUS_ADD' AND type = 'P')
DROP PROCEDURE PRC_CUS_ADD
GO
CREATE PROCEDURE PRC_CUS_ADD
@W_CODE INTEGER,
@W_LN VARCHAR(15),
@W_FN VARCHAR(15),
@W_INIT CHAR(1),
@W_AC CHAR(3),
@W_PH CHAR(8)
AS
BEGIN
-- attribute names are required when not giving values for all
INSERT INTO CUSTOMER(CUS_CODE, CUS_LNAME, CUS_FNAME, CUS_IN
VALUES (@W_CODE, @W_LN, @W_FN, @W_INIT, @W_AC, @W_P
PRINT ('Customer ' + @W_LN + ', ' + @W_FN + ' added.');
```

Messages Window: Displays the message "Command(s) completed successfully."

Embedded SQL

- SQL statements contained within an application programming language
- Host language: Any language that contains embedded SQL statements
- Differences between SQL and procedural languages
 - Run-time mismatch
 - SQL is executed one instruction at a time, executed at the server side
 - Host language runs at client side in its own memory space

Embedded SQL

- Processing mismatch
 - Conventional programming languages process one data element at a time
 - Newer programming environments (Visual Studio, .Net) manipulate data sets in a cohesive manner

-
- Data type mismatch
 - Data types provided by SQL might not match data types used in different host languages
 - To bridge the differences, the Embedded SQL standard defines a framework to integrate SQL within several programming languages

Embedded SQL

- Embedded SQL framework defines:
 - Standard syntax to identify embedded SQL code within the host language (EXEC SQL / END-EXEC)
 - Standard syntax to identify host variables which are preceded by a colon (:)
 - Communication area used to exchange status and error information between SQL and host language

END OF SESSION