|  |  |
| --- | --- |
| **Ex.No.7** | **CURSOR** |

**AIM**

To implement cursors in DBMS for efficient row-by-row data retrieval and manipulation.

**CREATING A TABLES**

SQL> CREATE TABLE employee (  
  2      id     NUMBER(5),  
  3      name   VARCHAR2(50),  
  4      basic  NUMBER(8,2)  
  5  );  
  
Table created.  
  
SQL> CREATE TABLE customers (  
  2      id      NUMBER(5),  
  3      name    VARCHAR2(50),  
  4      address VARCHAR2(100)  
  5  );  
  
Table created.

**INSERTING VALUES INTO TABLE**  
  
SQL> INSERT INTO employee VALUES (1, 'Job', 3000);  
  
1 row created.  
  
SQL> INSERT INTO employee VALUES (2, 'Max', 4000);  
  
1 row created.  
  
SQL> INSERT INTO employee VALUES (3, 'Sam', 3500);  
  
1 row created.  
  
SQL> INSERT INTO customers VALUES (101, 'Rose', 'New York');  
  
1 row created.

SQL> INSERT INTO customers VALUES (102, 'lisa', 'California');  
  
1 row created.  
  
SQL> INSERT INTO customers VALUES (103, 'jisoo', 'Texas');  
  
1 row created.  
  
SQL> COMMIT;  
  
Commit complete.

**IMPLICIT CURSOR  
EXAMPLE-1**

SQL>DECLARE

total\_rows NUMBER(2);

BEGIN

UPDATE employee SET basic = basic + 500;

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees updated.');

ELSIF SQL%FOUND THEN

total\_rows := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(total\_rows || ' employees updated.');

END IF;

END;

/  
PL/SQL procedure successfully completed.

**EXAMPLE -2**

SQL>SET SERVEROUTPUT ON;

SQL>DECLARE

total\_rows NUMBER(2);

BEGIN

UPDATE employee SET basic = basic + 500;

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees updated.');

ELSIF SQL%FOUND THEN

total\_rows := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(total\_rows || ' employees updated.');

END IF;

END;  
   /  
3 employees updated.  
  
PL/SQL procedure successfully completed.

**EXAMPLE -3**

SQL>DECLARE

total\_deleted NUMBER(2);

BEGIN

DELETE FROM employee WHERE basic < 3500;

IF SQL%NOTFOUND THEN

DBMS\_OUTPUT.PUT\_LINE('No employees deleted.');

ELSE

total\_deleted := SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(total\_deleted || ' employees deleted.');

END IF;

END;

/No employees deleted.  
  
PL/SQL procedure successfully completed.

**EXAMPLE-4**  
DECLARE

total\_inserted NUMBER := 0;

BEGIN

INSERT INTO customers VALUES (104, 'Job', 'Chicago');

total\_inserted := total\_inserted + SQL%ROWCOUNT;

INSERT INTO customers VALUES (105, 'Tom', 'Seattle');

total\_inserted := total\_inserted + SQL%ROWCOUNT;

DBMS\_OUTPUT.PUT\_LINE(total\_inserted || ' customers inserted.');

COMMIT;

END;

/

1 customers inserted.  
  
PL/SQL procedure successfully completed.

**EXPLICIT CURSOR**

**EXAMPLE -1**

SQl> DECLARE

c\_id customers.id%TYPE;

c\_name customers.name%TYPE;

c\_addr customers.address%TYPE;

CURSOR c\_customers IS

SELECT id, name, address FROM customers;

BEGIN

OPEN c\_customers;

LOOP

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

EXIT WHEN c\_customers%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(c\_id || ' ' || c\_name || ' ' || c\_addr);

END LOOP;

CLOSE c\_customers;

END;

 /  
  
PL/SQL procedure successfully completed.  
  
**EXAMPLE -2**  
SQL>DECLARE

c\_id customers.id%TYPE;

c\_name customers.name%TYPE;

c\_addr customers.address%TYPE;

CURSOR c\_customers IS

SELECT id, name, address FROM customers;

BEGIN

OPEN c\_customers;

LOOP

FETCH c\_customers INTO c\_id, c\_name, c\_addr;

EXIT WHEN c\_customers%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(c\_id || ' ' || c\_name || ' ' || c\_addr);

END LOOP;

CLOSE c\_customers;

END;

/

101 rose New York  
102 lisa California  
103 jisoo Texas  
  
PL/SQL procedure successfully completed.  
  
**EXAMPLE -3**

SQL>DECLARE

e\_id employee.id%TYPE;

e\_name employee.name%TYPE;

e\_basic employee.basic%TYPE;

CURSOR emp\_cursor IS

SELECT id, name, basic FROM employee WHERE basic > 3500;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO e\_id, e\_name, e\_basic;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('ID: ' || e\_id || ', Name: ' || e\_name || ', Basic: ' || e\_basic);

END LOOP;

CLOSE emp\_cursor;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END;

/

ID: 1, Name: Job, Basic: 4000

ID: 2, Name: Max, Basic: 5000

ID: 3, Name: Lisa, Basic: 4500

PL/SQL procedure successfully completed.

**EXAMPLE -4**   
  
SQL>DECLARE

DECLARE

c\_id customers.id%TYPE;

c\_name customers.name%TYPE;

c\_addr customers.address%TYPE;

CURSOR texas\_customers IS

SELECT id, name, address FROM customers WHERE address = 'Texas';

BEGIN

OPEN texas\_customers;

LOOP

FETCH texas\_customers INTO c\_id, c\_name, c\_addr;

EXIT WHEN texas\_customers%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Customer: ' || c\_name || ', Address: ' || c\_addr);

END LOOP;

CLOSE texas\_customers;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred: ' || SQLERRM);

END;

/

Customer: Job, Address: Texas

PL/SQL procedure successfully completed.

|  |  |  |
| --- | --- | --- |
| CONTENTS | MARKS ALLOTED | MARKS OBTAINED |
| Aim,Algorithm,SQL,PL/SQL | 30 |  |
| Execution and Result | 20 |  |
| Viva | 10 |  |
| Total | 60 |  |

RESULT

Achieved controlled and optimized data processing using cursors, enabling complex operations with improved precision.