Chuong Vu

HW4

2/2

5.13

a) We need to know the number of attributes and the names of the attributes of r to decide the number and names of columns in the table.

b) WE can use the JDBC methods getColumnCount() and the getColumnsName(int) to get the information's required.

c)
```
static void printTable(String r)
{
	try {
		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection.conn = DriverManager.getConnection(
			"jdbc:oracle:thin:star/X@//edgar.cse.lehigh.edu:1521/XE");
		Statement stmt = conn.createStatement);
		ResultSet rs = stmt.ExecuteQuery(r);
		ResultSetMetaData rsmd = rs.getMetaData();
		int count = rsmd.getColumnCount();
		System.out.println("<tr>");

		for(int i = 1; i <= count; i++){
			System.out.println("<td>" + rsmd.getColumnName(i) + "</td>");
		}

		System.out.println("</tr>");

		while(rs.next()){
		    System.out.println("<tr>");
		    for(int i = 1; i <= count; i++){
		        System.out.println("<td>" + rsmd.getString(i) + "</td>");
		    }
		    System.out.println("</tr>");
		}
		stmt.close();
		conn.close();

	} catch (SQLException sqle) {

		System.out.println("SQLException: " + sqle);
	}
}
```

5.14

a) Same as JDBC, we need to know the number of attributes and the names of the attributes of r to decide the number and names of columns in the table.

b) The function SQLNumResultCols(hstmt, &numColumn) can be used to find the number of columns in a statement, while the function SQLDescribeCol() can be used to find a column's name, data type, precision, scale, and nullability.

c)

```
typedef static struct {
    SQLCHAR name[33]; /* column name */
    void *value; /* column value */
    SQLSMALLINT type; /* column type */
    SQLINTEGER len; /* result value length */
} COL_RESULT;

void printTable(char *r) {
        RETCODE error;
        HENV env; /* environment */
        HDBC conn; /* database connection */
        SQLAllocEnv(&env);
        SQLAllocConnect(env, &conn);
        SQLConnect(conn, "db.yale.edu", SQL NTS, "avi", SQL NTS, "avipasswd", SQL NTS);
        {
                HSTMT stmt;
                char * sqlquery = "select * from " + r;
                SQLSMALLINT tot_cols;
                COL_RESULT *cols;
                long row;

                SQLAllocStmt(conn, &stmt);
                error = SQLExecDirect(stmt, sqlquery, SQL_NTS);
                if (error == SQL_SUCCESS) {
                        /* Allocate column results container */
                        SQLNumResultCols(stmt, &tot_cols);
                        cols = (COL_RESULT *)calloc(tot_cols, sizeof(COL_RESULT));

                        /* Fetch column names and bind column results */
                        for ( i = 0; i < tot_cols; ++i ) {
                                SQLDescribeCol(stmt, i+1, cols[i].name, 33, NULL, &cols[i].type,
                                                &size, NULL, NULL);
                                cols[i].value = malloc(size+6);
                                SQLBindCol(stmt, i+1, SQL_C_CHAR, cols[i].value,
                                                size+1, &cols[i].len);
                        }



                        /* Print all rows in record-oriented format */
                        for (row = 1; SQLFetch(stmt) == SQL_SUCCESS; ++row ) {
```

```
                        printf("**** row %ld:\n", row);
                        for (i = 0; i < tot_cols; ++i) {
                                printf(" %32.32s: %s\n", cols[i].name,
                        }
                        cols[i].len == SQL_NULL_DATA ? "NULL" : cols[i].value);
                }

                /* Drop statement handle and free allocated memory */
                SQLFreeHandle(SQL_HANDLE_STMT, stmt);
                for (i = 0; i < tot_cols; ++i) {
                        free(cols[i].value);
                }
                free((void *)cols);
        }
        SQLFreeStmt(stmt, SQL_DROP);
    }
    SQLDisconnect(conn);
    SQLFreeConnect(conn);
    SQLFreeEnv(env);
}
```

5.15

a) Using SQL function as appropriate.

```
create function avg_salary(cname varchar(15))
returns integer
declare result integer
        select avg(salary) into result
        from works W
        where W.company_name = cname
return result;
end
select company_name
form works
where avg_salary(company_name) > avg_salary("First Bank Corporation")
```

b) Without using SQL functions.

```
select company_name
from works w
group by company_name
having avg(salary) > (select avg(salary)
                      from works
                      where company_name="First Bank Corporation")
```