TCS1011 Data Structures and Algorithms

Trimester 3, Session 2008/2009
Faculty of Information Technology
Multimedia University

ASSIGNMENT

ADT for Relational Database Table

1. Title

Implement an ADT for a relational database table using C++.

2. Deadline

To be submitted via email to workjudge@gmail.com by the milestone deadlines set in lecture plan.

3. Grouping

To be done individually.

4. Objective

The objective of this assignment is to test the skill of students in creating a table ADT using the C++ programming language. The table ADT must be able to support the use of SQL-like commands to create tables, modify data and query data. The use of STL is not allowed for this assignment.

5. Milestone 1

a. Introduction

For this milestone, create a C++ program which can read and execute INSERT and SELECT commands from a text file for a table called Menu. The relation schema of Menu is

Menu(id, name, description, price)

as shown in the table below. The Menu table can contain up to a maximum of 100 tuples. The domain of each attribute is either character string of 1 to 40 characters or number in the range from -999999.99 to 999999.99.

Attribute	Domain
id	string (4)
name	string (15)
description	string (40)
price	number

Note:
string(n) means a character string with a maximum of n characters.
number means a number from -999999.99 to 999999.99.

Using an example to illustrate the requirements, let's assume the text file “menu.sql” contains the following commands and your C++ program has the name “database.exe”.

```
// menu.sql

INSERT INTO menu (id, name, description, price)
VALUES (“P001”, “Dinner Plate”, “Side order and 3 pieces of chicken”, 9.20);

INSERT INTO menu (id, name, description, price)
VALUES (“P002”, “Lunch Plate”, “Side order and 2 pieces of chicken”, 7.80);

INSERT INTO menu (id, name, description, price)
VALUES (“S003”, “Potato Snack”, “100g of potato slices”, 3.00);

INSERT INTO menu (id, name, description, price)
VALUES (“B004”, “Family Bucket”, “Drinks, side order and 9 pcs of chicken”,
25.00);

SELECT *
FROM menu;
```

Executing your program with the command

```
prompt> database menu.sql
```

your program should display the following result on screen.

```
MENU
-------------------------------------------------------------------------------
| ID   | NAME           | DESCRIPTION                          | PRICE      |
-------------------------------------------------------------------------------
| P001 | Dinner Plate    | Side order and 3 pieces of chicken       |       9.20 |
| P002 | Lunch Plate     | Side order and 2 pieces of chicken       |       7.80 |
| S003 | Potato Snack    | 100g of potato slices                    |       3.00 |
| B004 | Family Bucket   | Drinks, side order and 9 pcs of chicken  |      25.00 |
-------------------------------------------------------------------------------
```

```
4 rows selected.
```

The width of each column displayed is the maximum width specified for each corresponding attribute. In the case of Menu, the widths are 4+2 characters, 15+2 characters, 40+2 characters and 10+2 characters for id, name, description and price respectively.

If there are no data in the table Menu, your program should display

```
MENU

0 rows selected.
```

The sequence of INSERT statements and SELECT statements can be of any order. For example, the sequence of statements can start with four INSERT statements followed by two SELECT statements, and then followed by another 2 INSERT statements Your program should support any combination of INSERT and SELECT statements.

b. INSERT Statement

In addition to the example your program must be able to support NULL data for the INSERT statement except for the first attribute (id) which is the primary key. The following statement is a valid INSERT statement for your program.

```
INSERT INTO menu (id, name, description, price)
VALUES ("P010", "Breakfast Meal", NULL, NULL);
```

*Note: NULL's should be displayed as blanks when a SELECT statement is executed.

You can assume that the INSERT statement always has the four attributes (id, name, description and price) in a correct sequence. It consist of two lines starting with INSERT and VALUES respectively with no "return" characters in each line.

c. SELECT Statement

The SELECT statement for this milestone is a greatly simplified version of standard SQL. Your program only needs to support the * wild card or a sequence of attribute names for indicating the attributes to display, . For example,

```
SELECT id, price, name
FROM menu;

SELECT name
FROM menu;
```

are valid statements to be supported while

```
SELECT id, price + 3
FROM menu;

SELECT name, name, id
FROM menu;
```

are invalid statements which should be rejected.

For the WHERE clause, your program only needs to support one condition which contains only one operator, the = (equal) operator. For example,

```
SELECT price, name
FROM menu
WHERE name = "Potato Snack";

SELECT name, id
FROM menu
WHERE price = 5.00;
```

are valid statements to be supported while

```
SELECT id, price
FROM menu
WHERE price > 5.00;

SELECT name, id
FROM menu
WHERE ( name = "Potato Snack" ) AND ( id = "S003" );
```

are invalid statements which should be rejected.

The syntax of the SELECT statement is summarized in the table below.

Part of the SELECT statement	**Supported command structure**
SELECT	* or a sequence of attribute names
FROM	this is always `menu` as there is only one table
WHERE	supports only the = operator for only one condition.

Assume the SELECT statement consists of either two lines, if the WHERE clause is not included, or three lines starting with SELECT, FROM and WHERE respectively. In each line, assume also there are no "return" characters.

6. Milestone 2

a. Introduction

For this milestone, expand the support of SQL of your ADT table to CREATE TABLE, DROP TABLE and DELETE statements. Assume that your program can support only one table at any time.

To enable efficient processing of DELETE statements, use pointer-based linked list as your primary storage of data in the table. This avoids shifting of other tuples when a set of tuples is deleted.

b. CREATE TABLE Statement

The CREATE TABLE statement should support the creation of one relational table of any table name, and with a schema of up to six attributes of any data type of either STRING(n) or NUMBER. STRING (n) means a character string with a maximum of n characters, where n is a value between 1 to 40. NUMBER means a number from -999999.99 to 999999.99.

Hint: Use data dictionary to record attribute names and attribute data types and refer to it when processing SQL statements. This provides a higher abstract level of processing from native data types provided by C++.

An Example of a CREATE TABLE statement is shown below

```
CREATE TABLE staff
(
   sno       STRING( 5 )  PRIMARY KEY,
   name      STRING( 40 ),
   salary    NUMBER,
   position  STRING ( 10 ),
   telephone STRING ( 15 ),
   gender    STRING ( 1 )
);
```

Assume that the first attribute is always the primary key and must be unique and not contain a NULL value.

Your program should display

```
TABLE staff created.
```

after processing the statement.

c. DROP TABLE Statement

The DROP TABLE statement should drop a table from data dictionary and clean up memory used the table.

An Example of a DROP TABLE statement is shown below.

```
DROP TABLE staff;
```

Your program should display

```
TABLE staff dropped.
```

after processing the statement.

d. DELETE Statement

The DELETE statement to support consists of two lines as shown below.

```
DELETE FROM staff
WHERE sno = "A1234";
```

The features to support in WHERE clause is similar to the WHERE clause of a SELECT statement. There is always only one condition which uses the = operator.

Your program should display the number of tuples deleted, such as,

```
5 tuples deleted.
```

after processing the statement.

7. Milestone 3

a. Introduction

Add support for the UPDATE statement and the ORDER BY clause in the SELECT statement for this milestone.

b. UPDATE Statement

The UPDATE statement to support consists of three lines as shown below.

```
UPDATE staff
SET SALARY = 5000
WHERE position = "MANAGER";
```

The SET clause consist of only one attribute name on the left hand side and a constant value on the right hand side of the = (assignment) operator. The WHERE clause is similar to the WHERE clause of a SELECT statement.

Your program should display the number of tuples updated, such as,

```
2 tuples updated.
```

after processing the statement.

c. ORDER BY Clause for the SELECT Statement

A SELECT statement with an ORDER BY clause consists of three lines or four lines depending on whether the WHERE clause is specified.

```
SELECT name
FROM staff
WHERE position = "Programmer"
ORDER BY salary ASC;

SELECT sno, name
FROM staff
ORDER by sno DESC;
```

The ORDER BY clause only needs to handle one attribute. After the attribute name, either ASC or DESC is specified to indicate the order of display in either ascending order or descending order.

Based on the example above, your program should display the following (the salary of Alan < the salary of Stephen < the salary of Cathy < the salary of Bishop)

```
STAFF
-------------------------------------------
| NAME                                    |
-------------------------------------------
| Alan                                    |
| Stephen                                 |
| Cathy                                   |
| Bishop                                  |
-------------------------------------------
4 rows selected.
```

```
STAFF
-----------------------------------------------------------
| SNO        | NAME                                       |
-----------------------------------------------------------
| Z9999      | Alan                                       |
| K0034      | David                                      |
| J3421      | Stephen                                    |
| A1234      | Bishop                                     |
| A0723      | Cathy                                      |
-----------------------------------------------------------
5 rows selected.
```

after processing the statements.

7. Evaluation Criteria

Milestone 2 Mark Sheet

	Max	Marks
1. Features		
a. SELECT statement and INSERT statement are supported	1	
b. DROP TABLE statement is supported	1	
c. CREATE TABLE statement is supported	1	
d. DELETE statement is supported	1	
Max 2 marks only if pointer based linked list is not used.		
TOTAL	**4**	

Milestone 3 Mark Sheet

	Max	Marks
1. Features		
a. SELECT statement is supported	3	
a. INSERT statement is supported	3	
b. DROP TABLE statement is supported	2	
c. CREATE TABLE statement is supported	3	
d. DELETE statement is supported	3	
e. UPDATE statement is supported (Extra)	2	

f. ORDER BY clause is supported (Extra)	2	
Each feature will be evaluated based on fulfilment of requirements, basic error checking, quality of comments and good coding format and style.		
-5 marks if pointer based linked list is not used.		
TOTAL	**14**	

8. Submission Instruction

(a) Create a folder in the following format:

LECTURESECTION_MILESTONE_FULLNAME

For example, if your name is *Wan Nor Atikah*, you come from *TC101* lecture section, and you are submitting *Milestone 1*, then your folder name should be "**TC101_M1_WAN_NOR_ATIKAH**" without the double quotes as illustrated in the figure below.

For milestone 2 or 3 submission, just change M1 to M2 or to M3 respectively.

(b) Place **ONLY** your files of formats .cpp and .h in the folder. DO NOT place your **.exe** file in the folder.

(c) Zip your folder to create a zip (TC101_M1_WAN_NOR_ATIKAH.zip) archive .

(d) Send the zipped file to **workjudge@gmail.com** before the deadline. The email title should be

TCS1011 LECTURESECTION MILESTONE Submission

For example, if you come from *TC102* lecture section, and you are submitting *Milestone 2*, you email title should be "**TCS1011 TC102 M2 Submission**" without the double quotes. Take not that **the words are separated by space, not underscore.**

Late submission will be detected through the system clock.

9. Additional Information

1. **2 marks will be deducted if the submission instructions are not followed.**
2. You are expected to make sure that your code is easily readable. Pay attention to indentation.
3. In your .cpp and .h files, insert appropriate comments to help others to understand your code.
4. For ALL your .cpp and .h files, insert the following info at the header:

```
/************************************
Program: YOUR_PROGRAM_NAME.cpp
Course:  Data Structures and Algorithms
Year:    2008/09 Trimester 3
Name:    WAN NOR ATIKAH
ID:      10710045678
Lecture: TC101
Lab:     TC201
Email:   abc123@zmail.com
Phone:   018-1234567
************************************/
```