

Course ISE 326: Database Systems Engineering

Recitations 11–12: User Defined Types in MS SQL Server

Michael J. May

25 May – 1 June 2010

In this recitation we will use the Microsoft SQL Server to develop custom data types which we can then use in our code. The complex data types are written in C# and then deployed to the SQL Server for use. We will discuss in class more about custom data types in databases. This recitation is meant to introduce the concepts and get you started using them in a simple example.

1 Complex Numbers

A complex number consists of two parts - a real part and an imaginary part. Normally a complex number is written in the following manner:

$$a + bi$$

where $i^2 = -1$. Complex numbers are useful for many engineering applications. The following are the important properties of complex numbers:

Addition $(a + bi) + (c + di) = (a + c) + (b + d)i$

Subtraction $(a + bi) - (c + di) = (a - c) + (b - d)i$

Multiplication $(a + bi) \times (c + di) = ac + bci + adi + bdi^2 = (ac - bd) + (bc + ad)i$

Absolute Value $|a + bi| = \sqrt{a^2 + b^2}$

In the following recitation you will put together an example of code which defines a Complex Number type in SQL Server.

2 Defining the New Type

We will define the new data type in C# using MS Visual Studio. Please use MS Visual Studio 2005 since the Express editions do not let you connect to the MS SQL Server to run queries.

1. Create a new solution of type **Visual C#** → **Database** → **SQL Server Project**
2. Add a database reference to the Hermon server under your user name
3. Select 'No' to not enable SQL/CLR debugging
4. Select **Project** → **Add User-Defined Type**. Call the type “xxComplexNumber” where “xx” are the first letters of your first and last names
5. MS Visual Studio has done most of the work for you now, setting up a framework for the new type. You will need to add some code to flesh out the custom properties of the type.

6. Define two variables - x and y as well as corresponding properties X and Y
7. You will need to update the following functions and properties
 - Constructor - You can't declare an explicit empty constructor, but you should define at least one based on atomic parameters
 - IsNull - checks whether current instance has a null value or not
 - Null - gets a null value for the ADT
 - Parse - to create the Complex Number object from a string
 - To String - to convert the object for representation on the screen
8. Define functions to operate on the Complex Number ADT:
 - Addition(ComplexNumber) – (Calling it Add doesn't work)
 - Subtract(ComplexNumber)
 - Multiply(ComplexNumber)
 - AbsoluteValue()

3 Deploying and Using the ADT

Once you have successfully written the code for your ADT you need to compile and deploy it to your database. You can do so by selecting “Build Solution” or “Deploy Solution” from the “Build” menu on top. Doing so will export your new type to the Hermon database for use.

Once you have successfully deployed the code, create a table with the following signature:

`CTable1(id:int, cnumber:ComplexNumber)`

Remember to replace the ComplexNumer type above with the name of your custom ADT.

Once you have done so, populate the table with the following values:

```
create table CTable1 ( id int primary key, cnumber ComplexNumber);
INSERT INTO CTable1(id, cnumber) VALUES (1, '1+2i');
INSERT INTO CTable1(id, cnumber) VALUES (2, '4+82i');
INSERT INTO CTable1(id, cnumber) VALUES (3, '83+13i');
INSERT INTO CTable1(id, cnumber) VALUES (4, '7+373i');
INSERT INTO CTable1(id, cnumber) VALUES (5, '20+3i');
INSERT INTO CTable1(id, cnumber) VALUES (6, '48+85i');
INSERT INTO CTable1(id, cnumber) VALUES (7, '220+779i');
INSERT INTO CTable1(id, cnumber) VALUES (8, '99+96i');
INSERT INTO CTable1(id, cnumber) VALUES (9, '67+86i');
INSERT INTO CTable1(id, cnumber) VALUES (10, '1+2i');
INSERT INTO CTable1(id, cnumber) VALUES (11, '201+615i');
INSERT INTO CTable1(id, cnumber) VALUES (12, '1+996i');
INSERT INTO CTable1(id, cnumber) VALUES (13, '4+111i');
INSERT INTO CTable1(id, cnumber) VALUES (14, '46+531i');
INSERT INTO CTable1(id, cnumber) VALUES (15, '66+82i');
INSERT INTO CTable1(id, cnumber) VALUES (16, '4+82i');
```

4 Queries

You can use the functions you wrote in your queries. For the addition, subtraction, and multiplication functions, you will may need to construct ComplexNumbers on the fly. You can do so using the CONVERT function: `CONVERT(ComplexNumber, '1+1i')` creates a ComplexNumber which you can use for comparison or passing as a function parameter.

Once you have inserted the data into the tables, preform the following queries:

1. Select all of the complex numbers
2. Select the complex numbers whose id's are greater than 5
3. Select the id which have duplicate complex numbers
4. Select the absolute values for all of the complex numbers
5. Select the complex number with the largest absolute value
6. Select the id whose complex number has the largest absolute value
7. Add $10+10i$ to all of the complex numbers (This requires putting a function in the select statement. Don't forget to use the `.ToString()` method at the end of the function)
8. Select the ids whose complex numbers are less than $100+100i$ when added to $10+10i$.
9. Select the ids whose complex numbers pairwise add to $68+88i$
10. Try sorting by the complex number field (`order by cnumber`). What happens? How does MS SQL Server sort the complex numbers?
11. Try sorting by the absolute value of the complex number, its X value, and its Y value
12. You can project a single field from the complex number and use it as a regular integer. Use projection (dot operator) to compute the average X and Y values for all of the complex numbers
13. Find the count of the number of ids which have the complex value $4+82i$