Object-Oriented Query Language

Zhongling Li

February 26, 2006

Chapter 3
Language Reference Manual

3.1 Lexical Conventions

3.1.1 Comments

The characters /* introduce a comment, which can expand multiple lines and terminates with the characters */. The characters // starts a single-line comment.

3.1.2 Identifiers

An identifier is a sequence of letters and digits and must start with a letter. Character underscore _ counts as a letter. Identifiers are case sensitive, upper case and lower case are considered different.

3.1.3 Keywords

The following identifiers are reserved for use as keywords, and cannot be used otherwise:

```
entity    criteria       order
int       string         boolean
true      false          null
and       or             not
while     break          continue
if        else
```

3.1.4 Strings

A String is a sequence of characters enclosed by double quotes. Two consecutive double quotes represent a double quote.

3.1.5 Other Tokens

```
{     }     (     )     [     ]
+     -     *     /     =
==    >     >=    <     <=    !=
“     ,     ;
```

3.1.6 White Space

White space is defined as space character, horizontal tab, and line terminators.

3.2 Types

3.2.1 Primitive Types

Three primitive data types are supported at current stage. They are static, immutable. Every variable should be declared with a type tag and checked at compilation time.

- int 32-bit signed integers, default to 0
- string sequence of characters, default to null
- boolean only have two possible values: true or false, default to false

3.2.2 Reference Types

There are four kinds of reference types: entity, array, criteria and order.

Entity Type

Entity type maps to a database table. Entity type must be defined before use.

Each entity type should have a unique name and a list of fields of primitive type (int, string or boolean). One of the field must be an "int id" field which is the primary key of the mapping table (normally a surrogate key, like Oracle sequence or SQL Server identity type).

```
entity Address {
      int id,
      string street,
      string city,
      string state,
      boolean valid
}
```

Standard database CRUD (create, retrieve, update and delete) and query operations are provided at entity level, i.e. each entity type has create(), update(), delete(), get() and query() methods.

Objects

An object is an instance of an entity. It maps to a row in the database table. Object can only be created through entity's create() method, or loaded through get()/query() methods. Object does not need explicit type definition when it is first referenced. For example, both of the following two statements defines variable "a" which reference to an object of entity Address type:

```
a = Address.create("123 Main Street", "New York", "NY", "100020");
Address a = Address.get(100);
```

Entity's field value can only be accessed through its objects. Modification to the field will only be persisted to the database when entity's update() method is explicitly called.

```
print(a.state);   // will print string "NY"
a.state = "CA";   // will set state value to be "CA"
Address.update(a);      // persist the update to database
```

Array

Array is a collection of objects of the same entity type. It is declared with entity name followed by a pair of brackets "[]". Array can only be created at runtime through entity's query() method, which will return a collection of objects which satisfy the query criteria.

Array has index value is an int type start from 0 to the length of array, which is returned by array's length attribute. Object in the array can be referenced by its index value. Null array does not exist in OQL, an empty array (length=0) will be returned from query() if no matching data is found. For example:

```
Address[] aList = Address.query(criteria); // query to return array
print(aList.length);    // print the size of array
Address a = aList[0];   // reference the first object in the array
```

Criteria

Criteria define the basic query conditions. It specifies which entity field to apply the constraint to, and the value and type of the constraint. Criteria can also be grouped together with logical and / or. For example:

```
criteria c1 = criteria.ne("city", "New York");
criteria c2 = criteria.eq("valid", true);
criteria c3 = criteria.or(c1, c2);
```

Order

Order defines the ordering of the query result. Its reference variable is constructed by specifying which of the entity's field name to be sorted and in what order.

```
order asc = order.asc("city");
order desc = order.desc("state");
```

Variables

A variable is a storage location and has an associated type, which is a primitive type or one of the reference types. A variable of reference type can hold either a null reference or reference to an instance of that reference type.

3.3 Expressions

When an expression is evaluated (executed), the result denotes one of the following: a variable, a value or void.

3.3.1 Identifier

An Identifier is a left-value expression. It will be evaluated to some values bounded to this identifier.

3.3.2 Access array elements

Array expression is left-value. It contains an array identifier, followed by an int index value enclosed by a pair of "[" and "]".

3.3.3 Arithmetic expression

Arithmetic expression is applicable only to int type values (an expression or variable). It only supports binary operators like "+", "-", "*", "/" which indicates addition, subtraction, multiplication and division.

Arithmetic expressions are grouped left to right. "*" and "/" are of higher precedence than "+" and "-".

3.3.4 String expression

Operator "+" is applicable to string values to indicate a string concatenation as expression result.

```
s = "abc" + "123";      // s has value "abc123"
```

3.3.5 Relational expression

Binary relational operators "==", "<=", ">=", "<", ">" and "!=" evaluates whether the first operand is equal to, less than or equal to, greater than or equal to, less than, greater than or not equal to the second operand. The expression result is a boolean value (true or false).

For primitive type (int, string or boolean), since they are immutable, relation expression compares the value of two operands.

For reference types, only "==" and "!=" applies which compare the reference itself instead of target's value.

3.3.6 Logical expression

Logical operator “and”, “or”, “not” indicates logical and, or and negation of relational expressions. Operator “or” has the highest precedence, followed by “and” then “or”.

3.4 Statement

A sequence of statements will be executed sequentially unless the flow is altered by flow-control statement. A group of zero or more statements can be surrounded by a pair of “{“ and “}” to be treated as a single statement.

3.4.1 Assignment statement

Assignment statement has the form of two expressions on each side of assignment operator “=”, and terminated by “;”.

```
<left expression> = <right expression>
```

The right expression will be evaluated, and the result value will be assigned to the left expression.

3.4.2 Conditional statement

Conditional statement takes a logical expression and executed one statement based on the evaluation result of the expression.

```
if  ( <logical expression> )
      <statement>
```

or

```
if ( <logical expression> )
      <statement>
else
      <statement>
```

If the logical expression is true, first statement will be executed, otherwise the optional second statement is executed.

Each “else” is matched with the closest previous unmatched “if” statement.

3.4.3 Loop statement

A loop statement is in the form of:

```
while (<logical expression>)
      <statement>
```

The statement wrapped inside will be executed continuously if the logical expression evaluates to boolean value true.

A break statement in the form of keyword “break” followed by “;” will break the innermost loop.

A continue statement in the form of keyword “continue” followed by “;” will end the current iteration of innermost loop and process to the next iteration.

3.5 Built-in function

3.5.1 print(): Console output

Function print() takes only one argument and print its value to the console. For primitive type argument, its value will be directly printed. For reference type, all the values of the target object will be printed (all the fields of an entity type, or every entity of an array).

3.5.2 connect(): establish JDBC connection

Function connect() takes four arguments of string type: user, password, JDBC URL and JDBC driver class name. It should appear once as the first statement of a OQL program to prepare connection for execution. The connection will be closed automatically when the program finishes execution.