CQL Parser Design

This package is dedicated to support the OGC Common Query Language, version 2.0.1, as a query predicate language inside GeoTools. The rest of this document describe the syntax rules and the parser design.

Contents

- BNF
  - Syntax Rules
    - Existence Predicate
    - Comparison Predicate
    - Text Predicate (or Like Predicate)
    - NULL Predicate
    - Between Predicate
    - Temporal Predicate
    - INCLUDE/EXCLUDE Predicate
    - Expression
    - Georutine and Relational Geooperations
    - Geometry Literal
  - Lexical Rules
    - Numeric
    - Delimiter
    - Character String Literal
    - Identifier
- CQL Model
  - CQL Interface
  - CQL Implementation

BNF

The BNF is reproduced bellow to indicate the extensions made, which are highlighted in bold characters. The original BNF was extended thinking in SQL where expression. Moreover, we found some errors in the specification and made some corrections, that will help to use and analysis the CQL language.

Syntax Rules

```
<sequence of search conditions> ::= [1]
    <search condition>
   | <sequence of search conditions>
<semicolon> <search condition>


<search condition> ::= <boolean value
expression>
```

```
<boolean value expression> ::=
    <boolean term>
    | <boolean value expression> OR <boolean
term>

<boolean term> ::=
    <boolean factor>
    | <boolean term> AND <boolean factor>

<boolean factor> ::= [ NOT ] <boolean
primary>

<boolean primary> ::=
    <predicate>
    |<routine invocation>
    | <routine invocation>
    | <left paren> <search condition> <right
paren>
```

```
<predicate> ::=
    <comparison predicate>
    | <text predicate>
    | <null predicate>
    | <temporal predicate>
    | <classification predicate> [1]
    | <existence_predicate>
    | <between predicate>
    | <include exclude predicate>
```

Existence Predicate

```
<existence_predicate> :=
    <attribute_name> EXISTS
    | <attribute_name> DOES-NOT-EXIST
```

Comparison Predicate

```
<comparison predicate> ::=
    <attribute name> <comp op> <literal>
    | <attribute name> <comp op>
<expression>
```

```
<comp op> ::= <equals operator>
    | <not equals operator>
    | <less than operator>
    | <greater than operator>
    | <less than or equals operator>
    | <greater than or equals operator>
```

Text Predicate (or Like Predicate)

```
<text predicate> ::= <attribute name> [ NOT
] LIKE <character pattern>
```

```
<character pattern> ::= <character string
literal>

(* pattern examples :
        attribute like '%contains_this%'
        attribute like 'begins_with_this%'
        attribute like '%ends_with_this'
        attribute like 'd_ve' will match
'dave' or 'dove'
        attribute not like
'%will_not_contain_this%'
        attribute not like
'will_not_begin_with_this%'
        attribute not like
'%will_not_end_with_this' *)
```

NULL Predicate

```
<null predicate> ::= <attribute name> IS [
NOT ] NULL
```

Between Predicate

```
<between predicate> ::= <attribute name> [
NOT ] BETWEEN <literal> AND < literal > [1]
```

Temporal Predicate

```
<temporal predicate> ::=
    <attribute_name> "BEFORE" <date-time
expression>
    | <attribute_name> "BEFORE OR DURING"
<period>
    | <attribute_name> "DURING" <period>
    | <attribute_name> "DURING OR AFTER"
<period>
    | <attribute_name> "AFTER" <date-time
expression>

<date-time expression ::= <date-time> |
<period>
```

```
<date-time> ::= <full-date> "T" <UTC-time>
<full_date> ::= <date-year> "-" <date-month>
"-" <date-day>
<date-year> ::= <digit><digit><digit><digit>
<date-month> ::= <digit><digit>
<date-day> ::= <digit><digit>
<UTC-time> ::= <time-hour> ":" <time-minute>
":" <time-second> "Z"
<time-hour> ::= <digit><digit>
<time-minute> ::= <digit><digit>
<time-second> ::=
<digit><digit>[.<digit>...]
<duration> ::= "P" <dur-date> |
"T"<dur-time>
<dur-date> ::= <dur-day> | <dur-month> |
<dur-year> [<dur-time>]
<dur-day> ::= <digit>... "D"
<dur-month> ::= <digit>... "M" [<dur-day>]
<dur-year> ::= <didit>... "Y" [<dur-month>]
<dur-time> ::= <dur-hour> | <dur-minute> |
<dur-second>
<dur-hour> ::= <digit>... "H" [<dur-minute>]
<dur-minute> ::= <digit>... "M"
[<dur-second>]
<dur-second> ::= <digit>... "S"
<period> ::=
    <date-time> / <date-time>
    | <date-time> / <duration>
    | <duration> / <date-time>
```

INCLUDE/EXCLUDE Predicate

```
<include exclude predicate> ::= "INCLUDE" |
"EXCLUDE"
```

Expression

```
<expression> ::= <term> { <addition
operator>  <term> }
<addition operator>::= <plus sign> | <minus
sign>

<term> ::=  <factor> { <multiplication
operator> <factor> }
<multiplication operator> ::= <asterisk> |
<solidus>

<factor> ::=
   <function>
 | <literal>
 | <attribute>
 | ( <expression> )
 | [ <expression> ]
```

```
<function> ::= <identifier> "(" ( <function
arguments> ("," <function arguments> )* )?
")"

<function arguments> ::= <literal> |
<attribute> | <expression>
```

```
<literal> ::= <signed numeric literal>|
<general literal>

<signed numeric literal> ::= [ <sign> ]
<unsigned numeric literal>

<general literal> ::=
    <character string literal>
    | <boolean literal>
    | <geography literal

<boolean literal> ::= TRUE | FALSE | UNKNOWN
```

Georutine and Relational Geooperations

```
<routine invocation> ::=
     <geoop name><georoutine argument list>
    | <relgeoop name><relgeoop argument
list>
    | BBOX <bbox argument list>

<routine name> ::= < attribute name>
```

```
<geoop name> ::=   EQUAL | DISJOINT |
INTERSECT | TOUCH | CROSS | WITHIN |
CONTAINS |OVERLAP | RELATE [2]

Since 2.6.1

<geoop name> ::= EQUALS | DISJOINT |
INTERSECTS | TOUCHES | CROSSES | WITHIN |
CONTAINS | OVERLAPS | RELATE
```

```
<bbox argument list>::= "(" <attribute>
","<min X> ","<min Y> ","<max X> ","<max
Y>["," <crs>] ")"  [1]

<min X> ::= <signed numerical literal>
<min Y> ::= <signed numerical literal>
<max X> ::= <signed numerical literal>
<max Y> ::= <signed numerical literal>
<crs> ::=  ... (* default: EPSG:4326. *)
```

```
<relgeoop name> ::= DWITHIN | BEYOND

<argument list> ::= <left paren> [
<positional arguments> ] <right paren>

<positional arguments> ::= <argument> [ \{
<comma> <argument> \} ...]

<argument> ::= <literal> | <attribute name>

<georoutine argument list> ::=  <left
paren><attribute name><comma><geometry
literal><right paren>

<relgeoop argument list> ::= <left
paren><attribute name><comma><geometry
literal><comma><tolerance><right paren>

<tolerance> ::=<unsigned numeric
literal><comma><distance units>

<distance units> ::= = "feet" | "meters" |
"statute miles" | "nautical miles" |
"kilometers"
```

Geometry Literal

```
<geometry literal> :=
    <Point Tagged Text>
    | <LineString Tagged Text>
    | <Polygon Tagged Text>
    | <MultiPoint Tagged Text>
    | <MultiLineString Tagged Text>
    | <MultiPolygon Tagged Text>
    | <GeometryCollection Tagged Text>
    | <Envelope Tagged Text>

<Point Tagged Text> ::= POINT <Point Text>

<LineString Tagged Text> ::= LINESTRING
<LineString Text>

<Polygon Tagged Text> ::= POLYGON <Polygon
Text>

<MultiPoint Tagged Text> ::= MULTIPOINT
<Multipoint Text>

<MultiLineString Tagged Text> ::=
MULTILINESTRING <MultiLineString Text>

<MultiPolygon Tagged Text> ::= MULTIPOLYGON
<MultiPolygon Text>

<GeometryCollection Tagged Text>
::=GEOMETRYCOLLECTION <GeometryCollection
Text>
```

```
<Point Text> := EMPTY | <left paren> <Point>
<right paren>
<Point> := <x><space><y>
<x> := numeric literal
<y> := numeric literal
<LineString Text> := EMPTY | <left paren>
<Point> \{<comma><Point >\}...<right paren>
<Polygon Text> := EMPTY | <left
paren><LineString Text>\{<comma><LineString
Text> \}...<right paren>
<Multipoint Text> := EMPTY | <left
paren><Point Text>\{<comma><Point Text
>\}...<right paren>
<MultiLineString Text> := EMPTY | <left
paren><LineString Text>\{<comma><LineString
Text>\}...<right paren>
<MultiPolygon Text> := EMPTY | <left
paren><Polygon Text>\{<comma><Polygon
Text>\}...<right paren>
<GeometryCollection Text> := EMPTY | <left
paren><Geometry Tagged
Text>\{<comma><Geometry Tagged
Text>\}...<right paren>
<Envelope Tagged Text> ::= ENVELOPE
<Envelope Text>
```

```
<Envelope Text> ::= EMPTY |
    <left paren><WestBoundLongitude><comma>
        <EastBoundLongitude><comma>
        <NorthBoundLatitude><comma>
        <SouthBoundLatitude>< <right paren>
```

```
<WestBoundLongitude> ::= numeric literal
<EastBoundLongitude> ::= numeric literal
<NorthBoundLatitude> ::= numeric literal
<SouthBoundLatitude> ::= numeric literal
```

Sym	Note	Reference
ℹ	extensions	1
⚠	warning	2

Lexical Rules

```
<SQL terminal character> ::= <SQL language
character>
<SQL language character> ::= <simple Latin
letter>
| <digit>
| <SQL special character>
<simple Latin letter> ::= <simple Latin
upper case letter>
```

```
                                | <simple Latin
lower case letter>
<simple Latin upper case letter> ::=
       A | B | C | D | E | F | G | H | I | J
| K | L | M | N | O
       | P | Q | R | S | T | U | V | W | X |
Y | Z
<simple Latin lower case letter> ::=
       a | b | c | d | e | f | g | h | i | j
| k | l | m | n | o
       | p | q | r | s | t | u | v | w | x |
y | z
<digit> ::=
       0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<SQL special character> ::= <space>
                                | <double quote>
                                | <percent>
                                | <ampersand>
                                | <quote>
                                | <left paren>
                                | <right paren>
                                | <asterisk>
                                | <plus sign>
                                | <comma>
                                | <minus sign>
                                | <period>
                                | <solidus>
                                | <colon>
                                | <semicolon>
                                | <less than
operator>
```

```
                                | <equals
operator>
                                | <greater than
operator>
                                | <question mark>
                                | <left bracket>
                                | <right bracket>
                                | <circumflex>
                                | <underscore>
                                | <vertical bar>
                                | <left brace>
                                | <right brace>
<space> ::= /*space character in character
set in use
              In ASCII it would be 40*/
<double quote> ::= "
<percent> ::= %
<ampersand> ::= &
<quote> ::= '
<left paren> ::= (
<right paren> ::= )
<asterisk> ::= *
<plus sign> ::= +
<comma> ::= ,
<minus sign> ::= -
<period> ::= .
<solidus> ::= /
<colon> ::= :
<semicolon> ::= ;
<less than operator> ::= <
<equals operator> ::= =
```

```
<greater than operator> ::= >
<question mark> ::= ?
<left bracket> ::= [
<right bracket> ::= ]
<circumflex> ::= ^
<underscore> ::= _
<vertical bar> ::= |
<left brace> ::={
<right brace> ::=}
<separator> ::= { <comment> | <space> |
<newline> }...
<token> ::= <nondelimiter token>
            | <delimiter token>
<nondelimiter token> ::= <regular
identifier>
                         | <key word>
```

```
                                | <unsigned numeric
literal>
```

```
<key word> ::= <reserved word>
<reserved word> ::= NOT | AND | OR | LIKE |
                    IS | NULL |
                    EXISTS | DOES-NOT-EXIST
|
                    DURING | AFTER | BEFORE
                    INCLUDE | EXCLUDE |
                    TRUE | FALSE |
                    EQUALS | DISJOINT |
INTERSECTS | TOUCHES | CROSSES | WITHIN |
CONTAINS| OVERLAPS | RELATE | DWITHIN |
BEYOND |
                    POINT | LINESTRING |
POLYGON |
                    MULTIPOINT |
MULTILINESTRING | MULTIPOLYGON |
GEOMETRYCOLLECTION
```

Numeric

```
<unsigned numeric literal> ::= <exact
numeric literal>
                                | <approximate
numeric literal>
<exact numeric literal> ::=
                <unsigned integer>
[<period>[<unsigned integer> ] ]
                | <period> <unsigned
integer>
<unsigned integer> ::= <digit>...
<approximate numeric literal> ::= <mantissa>
E <exponent>
<mantissa> ::= <exact numeric literal>
<exponent> ::= <signed integer>
<signed integer> ::= [ <sign> ] <unsigned
integer>
<sign> ::= <plus sign> | <minus sign>
```

Delimiter

```
<delimiter token> ::= <character string
literal>
                      | <SQL special
character>
                      | <not equals operator>
                      | <greater than or
equals operator>
                      | <less than or equals
operator>
                      | <concatenation
operator>
                      | <double greater than
operator>
                      | <right arrow>
                      | <left bracket>
                      | <right bracket>
<character string literal> ::=
                <quote> [ <character
representation>... ] <quote>
<character representation> ::= <nonquote
character> | <quote symbol>
<quote symbol> ::= <quote><quote>
```

```
<not equals operator> ::= <>
<greater than or equals operator> ::= >=
<less than or equals operator> ::= <=
```

Character String Literal

```
<character string literal> ::=
      <quote> [ <character
representation>... ] <quote>
<character representation> ::= <nonquote
character> | <quote symbol>

<quote symbol> ::= <quote><quote>
```

Identifier

The following section is intended to give context for identifier and namespaces. It assumes that the default namespace is specified in the query request and does not allow any overrides of the namepace

```
<regular identifier> ::= <identifier body>
<identifier body> ::=
<identifier start> [ { <underscore> |
<identifier part> }... ]
<identifier start> ::= <simple latin letter>
<identifier part> ::= <identifier start>
                    | <digit>
```

```
<identifier> ::= <identifier start> [ {
<colon> | <identifier part> }... ]
<identifier start> ::= <simple Latin letter>
<identifier part> ::= <simple Latin letter>
| <digit>
<attribute name> ::= <simple attribute name>
| <compound attribute name>
```

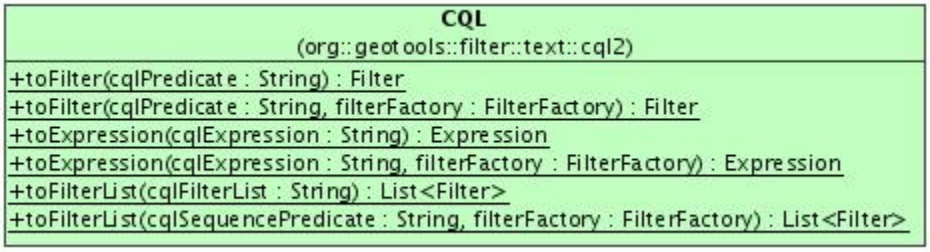
```
<simple attribute name> ::= <identifier>
<compound attribute name> ::=
<identifier><period>

[{<identifier><period>}...]
                                <simple
attribute name>
```

CQL Model

CQL Interface

This diagram presents the package interface. In parser's protocol methods performs the parsing of CQL and builds the the filter.



CQL Implementation

The figure shows the principal class in the parser and build process. CQLParser does a top down analysis of the

input string and makes the parsing tree. Each time CQLParser builds a node, it makes a call to CQLCompiler, that implements the semantic actions related and builds the product or subproduct required to make the Filter at the end of the parsing process