MyCQL Grammar v1.0

for

MyCQ

Version: 1.0

Author: Park Jae Hong (jhpark@mycqsystems.com)

MyCQ Inc.

2009.3

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
First Release.	2009.3.1		1.0

1. Introduction

1.1 Purpose

This document specifies a continuous query language grammar through which users may request a query on the abstracted data stream models. The design of this grammar recognizes that it needs time based sliding window and count based sliding window concepts to query on the abstracted data stream models.

1.2 Intended Audience

This document is intended for MyCQ users, developers, project manager, testers, and documentation writers.

2. MyCQL Grammar v1.0

```
CQLStatement :=
        CQLWindowStatement (CQLUsingProperty)+ CQLSelectStatement
;
CQLWindowStatement :=
        WINDOW CQLWindowExpr (',' CQLWindowExpr)*
;
CQLWindowExpr :=
        CQLQueueName AS CQLWindowName CQLWindowAttributes
;
CQLQueueName :=
        ID
;
CQLWindowName :=
        ID
;
CQLWindowAttributes :=
            LEFTBRACKET SIZE EQUAL CQLWindowTimeValue ',' SLIDE EQUAL CQLWindowTimeValue RIGHTBRACKET
        |   LEFTBRACKET SIZE EQUAL CQLWindowCountValue ',' SLIDE EQUAL CQLWindowCountValue RIGHTBRACKET
;
CQLWindowTimeValue :=
        (unsigned_integer DAY)+ (unsigned_integer HOUR)+ (unsigned_integer MIN)+ (unsigned_integer SEC)+
;
CQLWindowCountValue :=
        unsigned_integer
;
CQLUsingProperty :=
        USING PROPERTY ID (',' ID)*
;
CQLSelectStatement :=
        SELECT CQLSelectPart FROM CQLFromPart
        (WHERE CQLWhere)+
        (GROUP BY CQLGroupBy)+
        (HAVING CQLHaving)+
        (CQLSetOperation)+
        (ORDER BY CQLOrderBy)+
;
CQLSelectPart :=
        (ALL | DISTINCT)+ [ASTERISK | CQLSelectExpr (',' CQLSelectExpr)*]
;
CQLSelectExpr :=
        [ASTERISK | CQLExpr (AS CQLName)+ | CQLTableAlias '.' ASTERISK]
;
CQLExpr :=
        CQLAndCondition (OR CQLAndCondition)+
;
CQLAndCondition :=
        CQLCondition (AND CQLCondition)+
;
CQLCondition :=
        [CQLOperand (CQLConditionRightHandSide)+ | NOT CQLCondition    | EXISTS '(' CQLSelectStatement ')']
;
```

```
CQLOperand :=
        CQLSummand (CONCAT CQLSummand)+
;
CQLSummand :=
        CQLFactor ((PLUS | MINUS) CQLFactor)+
;
CQLFactor :=
        CQLTerm ((ASTERISK | SLASH) CQLTerm)+
;
CQLTerm :=
        [  CQLValue
         | CQLColumnName
         | CQLFunction
         | (MINUS | PLUS) CQLTerm
         | '(' CQLExpr ')'
         | CQLSelectStatement
         | CQLCase
         | CQLCaseWhen
        ]
;
CQLValue :=
        [
           STRING_LITERAL
         | INTEGER_LITERAL
         | FLOAT_LITERAL
         | DATE STRING_LITERAL
         | TIME STRING_LITERAL
         | TIMESTAMP STRING_LITERAL
         | (TRUE | FALSE)
         | NULL
        ]
;
CQLColumnName :=
        ID '.' ID
;
CQLName :=
            ID
        |   STRING_LITERAL
;

CQLFunction :=
        CQLFunctionName '(' ASTERISK+ DISTINCT+ CQLExpr+ (',' CQLExpr)* ')'
;
CQLFunctionName :=
        [
            ABS
          | AVG
          | COUNT
          | MAX
          | MIN
          | SUM
          | TOTAL
          | GROUP_CONCAT
          | DATE
          | TIME
```

```
          | IFNULL
          | LIKE
          | LENGTH
          | LOWER
          | LTRIM
          | NULLIF
          | QUOTE
          | RANDOM
          | ROUND
          | RTRIM
          | SOUNDEX
          | SUBSTR
          | TRIM
          | UPPER
          | REPLACE
          | DATETIME
        ]
;
CQLCase :=
        CASE CQLExpr (WHEN CQLExpr THEN CQLExpr)+ (ELSE CQLExpr)+ END
;
CQLCaseWhen :=
        CASE (WHEN CQLExpr THEN CQLExpr)+ (ELSE CQLExpr)+ END
;
CQLConditionRightHandSide :=
        [
            CQLCompare ( (ALL | ANY | SOME) '(' CQLSelectStatement ')' | CQLOperand )
          | IS (NOT)+ NULL
          | BETWEEN CQLOperand AND CQLOperand
          | (NOT)+ IN (CQLSelectStatement | '(' CQLExpr (',' CQLExpr)* ')')
          | (NOT)+ LIKE CQLOperand (ESCAPE STRING_LITERAL)+
          | (NOT)+ REGEXP CQLOperand
        ]
;
CQLCompare :=
        [
            EQUAL
          | LESS
          | GREATER
          | NOTEQUAL
          | LESSEQUAL
          | GREATEREQUAL
          | NOTEQUAL2
        ]
;
CQLFromPart :=
        CQLTableExpr (',' CQLTableExpr)*
;
CQLTableExpr :=
        (CQLTableName | '(' CQLSelectStatement ')') (AS)+ CQLTableAlias
        ( (((LEFT | RIGHT | FULL) (OUTER)+) | INNER)+ JOIN
        (CQLTableName | '(' CQLSelectStatement ')') (AS CQLTableAlias)+ (ON CQLExpr)+ )*
;

CQLTableName :=
```

```
        ID ('.' ID)+
;
CQLTableAlias :=
        ID
;
CQLWhere :=
        CQLExpr
;
CQLGroupBy :=
        CQLExpr (',' CQLExpr)*
;
CQLHaving :=
        CQLExpr
;
CQLOrderBy :=
        CQLOrder (',' CQLOrder)*
;
CQLOrder :=
        [INTEGER_LITERAL | CQLExpr] (ASC | DESC)+ (NULLS (FIRST | LAST))+
;
CQLSetOperation :=
        ((UNION (ALL)+) | EXCEPT | INTERSECT) CQLSelectStatement
;
ID :=
        alphabets (alphabets_and_numerics | '_' | '$' | '#')*
;
STRING_LITERAL :=
        quote (any_characters except quote)+ quote
;
INTEGER_LITERAL :=
        integer
;
FLOAT_LITERAL :=
            (fractional_constant) (exponent_part)+
        |   (digit)+ (exponent_part)
;
alpahabets :=
        ["A"-"Z", "a"-"z"]
;
alphabets_and_numerics :=
        (alphabets | integer)+
;
integer :=
        (["0"-"9"])+
fractional_constant :=
            (digit)* (.) (digit)*
        |   (digit)+ (.)
;
exponent_part :=
        ('e'|'E') ('+' |'-')+ (digit)+
;
quote :=
        '''
;
CONCAT := "||";
```

```
LESS := '<';
LESSEQUAL := "<=";
GREATER := '>';
GREATEREQUAL := ">=";
EQUAL := '=';
NOTEQUAL := "!=";
NOTEQUAL2 := "<>";
ASTERISK := '*';
SLASH := '/';
PLUS := '+';
MINUS := '-';
LEFTBRACKET := '[';
RIGHTBRACKET := ']';
SPACE := " ";
```

3. MyCQL Examples

3.1 Sliding Windows

3.1.1 Simple Time Based Sliding Window

```
Ex1)
window
        queue1 as win1[size=30sec, slide=30sec]
select count(*) from win1
```

3.1.2 Complex Time Based Sliding Window

```
Ex1)
window
        queue1 as win1[size=5min slide=5min],
        queue1 as win2[size=1day, slide=1day]
select count(*) from win1, win2

Ex2)
window
        queue1 as win1[size=5min, slide=10sec],
        queue1 as win2[size=5min, slide=10sec],
        queue2 as win3[size=1hour, slide=5min],
        queue3 as win4[size=1hour, slide=5min]
select * from win1, win2, win3, win4
```

3.1.3 Simple Count Based Sliding Window

```
Ex1)
window
        queue1 as win1[size=10, slide=10]
select count(*) from win1
```

3.1.4 Complex Count Based Sliding Window

```
Ex1)
window
        queue1 as win1[size=5 slide=5],
        queue1 as win2[size=100, slide=100]
select count(*) from win1, win2

Ex2)
window
        queue1 as win1[size=5, slide=10],
        queue1 as win2[size=5, slide=10],
        queue2 as win3[size=100, slide=5],
        queue3 as win4[size=100, slide=5]
select * from win1, win2, win3, win4
```

3.2 Time Based Sliding Window

3.2.1 Normal Sliding Window

```
Ex1)
window
        queue1 as win1[size=10sec, slide=10sec]
select count(*) from win1

Ex2)
window
        queue1 as win1[size=10sec, slide=10sec]
        queue2 as win2[size=10sec, slide=10sec]
select count(*) from win1, win2
```

3.2.2 Overlapped Sliding Window

```
Ex1)
window
        queue1 as win1[size=10sec, slide=5sec]
select count(*) from win1

Ex2)
window
        queue1 as win1[size=10sec, slide=5sec]
        queue2 as win2[size=20sec, slide=5sec]
select count(*) from win1, win2
```

3.2.3 Jump Sliding Window

```
Ex1)
window
        queue1 as win1[size=5sec, slide=10sec]
select count(*) from win1

Ex2)
window
        queue1 as win1[size=5sec, slide=10sec]
        queue2 as win2[size=5sec, slide=20sec]
select count(*) from win1, win2
```

3.3 Count Based Sliding Window

3.3.1 Normal Sliding Window

```
Ex1)
window
        queue1 as win1[size=10, slide=10]
select count(*) from win1
```

```
Ex2)
window
        queue1 as win1[size=10, slide=10]
        queue2 as win2[size=10, slide=10]
select count(*) from win1, win2
```

3.3.2 Overlapped Sliding Window

```
Ex1)
window
        queue1 as win1[size=10, slide=5]
select count(*) from win1

Ex2)
window
        queue1 as win1[size=10, slide=5]
        queue2 as win2[size=20, slide=5]
select count(*) from win1, win2
```

3.3.3 Jump Sliding Window

```
Ex1)
window
        queue1 as win1[size=5, slide=10]
select count(*) from win1]

Ex2)
window
        queue1 as win1[size=5, slide=10]
        queue2 as win2[size=5, slide=20]
select count(*) from win1, win2
```

3.4 Continuous Query Examples

3.4.1 select

```
window
 queue1 as window1[size=1sec, slide=1sec]
select col1 from window1
```

3.4.2 distinct

```
window
 queue1 as win1[size=10sec, slide=10sec]
select
 distinct col1 from win1
```

3.4.3 compare

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
 where col1=1000
```

3.4.4 compare

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
 where string like 'hello%' and (col2=1000 or col3=5000)
```

3.4.5 compare

```
window
 queue1 as win1[size=5sec, slide=5sec]
select count(*) from win1
where col1 > 100.01 and col2 < 130
```

3.4.6 compare

```
window
 queue1 as win1[size=1sec, slide=1sec]
select date from win1
where date > '2009-01-01'
```

3.4.7 like

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
 where col1 like 'hello%'
```

3.4.8 like

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
 where col1 like 'hello%' and col2 like 'hi%'
```

3.4.9 like

```
window
 queue1 as win1[size=10sec, slide=10sec]
select col1, col2 from win1
where col1 like '%hello%'
```

3.4.10 not like

```
window
 queue1 as win1[size=10sec, slide=10sec]
select * from win1
where col1 not like '%hello%'
```

3.4.11 not like

```
window
 queue1 as win1[size=10sec, slide=10sec]
select * from win1
 where string not like '%hello_%'
```

3.4.12 order by

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
order by col1
```

3.4.13 order by asc

```
window
 queue1 as win1[size=10sec, slide=10sec]
select * from win1
order by int asc
```

3.4.14 order by desc

```
window
 queue1 as win1[size=10sec, slide=10sec]
select * from win1
order by int desc
```

3.4.15 where in

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
where col1 in (1000, 50)
```

3.4.16 where between

```
window
 queue1 as win1[size=5sec, slide=5sec]
```

```
select * from win1
where col1 between 100 and 250
```

3.4.17 as

```
window
 queue1 as win1[size=5sec, slide=5sec]
select A.col1, A.col2 from win1 as A
where A.col1 = 128
```

3.4.18 inner join

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=5sec, slide=5sec]
select win1.col1, win1.col2 from win1 inner join win2 on win1.col1 = win2.col1
order by win1.col1
```

3.4.19 left join

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=5sec, slide=5sec]
select win1.col1, win1.col2 from win1 left join win2 on win1.col1 = win2.col2
order by win1.col1
```

3.4.20 inner join and aggregation

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=5sec, slide=5sec]
select avg(win1.col1), max(win1.col1), min(win1.col1) from win1 inner join win2 on win1.col1 = win2.col1
order by win1.col1
```

3.4.21 calculation

```
window
 queue1 as win1[size=5sec, slide=5sec]
select (col1*2), (col2+50) from win1
```

3.4.22 where exists

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=1sec, slide=1sec]
select col1, col2 from win1
where exists
```

```
 (
  select * from win2
  where col1 = 128
 )
```

3.4.23 group by, having

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=1sec, slide=1sec]
select col1, col2 from win1
group by col1
having max(col1) >=  (select avg(col1) from win2 where win1.col2 = win2.col2 )
```

3.4.24 group by

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
group by col1
order by col1
```

3.4.25 group by, having

```
window
 queue1 as win1[size=5sec, slide=5sec]
select * from win1
group by col1
having sum(col1) > 1000 and avg(col1) > 100
order by col1
```

3.4.26 not in, union

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=5sec, slide=5sec]
select * from win1
where col1 not in(2000,300)
group by col1
union
select * from win2
order by col1
```

3.4.27 union

```
window
 queue1 as win1[size=5sec, slide=5sec],
 queue2 as win2[size=5sec, slide=5sec]
select col1, col2 from win1
```

```
union
select col1, col2 from win2
```

3.4.28 union

```
window
 queue1 as win1[size=1sec, slide=1sec],
 queue1 as win2[size=5sec, slide=5sec],
 queue2 as win3[size=5sec, slide=5sec]
select col1, col2 from win1
union
select col1, col2 from win2
union
select col1, col2 from win3
```

3.4.29 union all

```
window
 queue1 as win1[size=1sec, slide=1sec],
 queue1 as win2[size=5sec, slide=5sec],
 queue2 as win3[size=5sec, slide=5sec]
select col1, col2 from win1
union all
select col1, col2 from win2
union all
select col1, col2 from win3
```

3.4.30 count()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select count(distinct col1) from win1
```

3.4.31 avg()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select avg(col1) from win1
```

3.4.32 max()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select max(col1) from win1
```

3.4.33 min()

```
window
```

```
 queue1 as win1[size=1sec, slide=1sec]
select min(col1) from win1
```

3.4.34 sum()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select sum(col1) from win1
```

3.4.35 round()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select round(col1) from win1
```

3.4.36 date()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select date('now') from win1
```

3.4.37 time()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select time('now') from win1
```

3.4.38 datetime()

```
window
 queue1 as win1[size=1sec, slide=1sec]
select datetime('now') from win1
```