LMDQL EBNF

<MDX_statement> ::= <select_statement>
| <create_formula_statement>
| <drop_formula_statement>
<select_statement> ::= **[$VARIABLE <variable_specification>]**
[WITH <formula_specification>]
SELECT [<axis_specification>
[, <axis_specification>...]]
FROM [<cube_specification>]
[WHERE [<slicer_specification>]]
[<cell_props>]
<variable_specification> ::= <member>.<identifier>
[<or> <member>.<identifier>…]
<or> ::= |
<formula_specification> ::= <single_formula_specification> [<single_formula_specification>...]
<single_formula_specification> ::= <member_specification> | <set_specification>
<member_specification> ::= MEMBER <member_name> AS <value_expression>
[, <solve_order_specification>] [, <member_property_definition>...]
<member_name> ::= <member>.<identifier> | <cube_name>.<member>.<identifier>
<solve_order_specification> ::= SOLVE_ORDER = <unsigned_integer>
<member_property_definition> ::= <identifier> = <value_expression>
<set_specification> ::= SET <set_name> AS <set>
<set_name> ::= <identifier> | <cube_name>.<identifier>
<axis_specification> ::= [NON EMPTY] <set> [<dim_props>] ON <axis_name>
<axis_name> ::= COLUMNS
| ROWS
| PAGES
| CHAPTERS
| SECTIONS
| AXIS(<index>)
| LEVEL(<index>)
<dim_props> ::= [DIMENSION] PROPERTIES <property> [, <property>...]
cube_specification> ::= [<cube_name> [,<cube_name>...]]
<slicer_specification> ::= {<set> | <tuple>}
<cell_props> ::= [CELL] PROPERTIES <cell_property> [, <cell_property>...]
<cell_property> ::= <mandatory_cell_property>
| <optional_cell_property>
| <provider_specific_cell_property>
<mandatory_cell_property> ::= CELL_ORDINAL | VALUE | FORMATTED_VALUE
<optional_cell_property> ::= FORMAT_STRING
| FORE_COLOR
| BACK_COLOR
| FONT_NAME
| FONT_SIZE
| FONT_FLAGS
<provider_specific_cell_property> ::= <identifier>
<create_formula_statement> ::= CREATE [<scope>]<formula_specification>
<drop_formula_statement> ::= <drop_member_statement>
| <drop_set_statement>
<drop_member_statement> ::= DROP MEMBER <member_name>
[, <member_name>...]
<drop_set_statement> ::= DROP SET <set_name> [, <set_name>...]
<scope> := GLOBAL | SESSION
<identifier> ::= <regular_identifier> | <delimited_identifier>
<regular_identifier> ::= <alpha_char> [{<alpha_char> | <digit>
| <underscore>}...]
<delimited_identifier> ::=
<start_delimiter>{<double_end_delimiter> | <nondelimit_end_symbol>}
[{<double_end_delimiter> | <nondelimit_end_symbol> }...]

<end_delimiter>
<start_delimiter> ::= <open_bracket>
<end_delimiter> ::= <close_bracket>
<double_end_delimiter> ::= <end_delimiter> <end_delimiter>
<nondelimit_end_symbol> ::= !! Any character except <end_delimiter>
<cube_name> ::= [[[<data_source>.] <catalog_name>.] [<schema_name>.]
<identifier>
<data_source> ::= <identifier>
<catalog_name> ::= <identifier>
<schema_name> ::= <identifier>
<dim_hier> ::= [<cube_name>.]<dimension_name>
| [[<cube_name>.]< dimension_name>.]<hierarchy_name>
<dimension_name> ::= <identifier>
| <member>.DIMENSION
| <level>.DIMENSION
| <hierarchy>.DIMENSION
<dimension> ::= <dimension_name>
<hierarchy> ::= <hierarchy_name>
<hierarchy_name> ::= <identifier>
| < member>.HIERARCHY
| <level>.HIERARCHY
<level> ::= [<dim_hier>.]< identifier>
| <dim_hier>.LEVELS(<index>)
| <member>.LEVEL
<member> ::= [<level>.]<identifier>
| <dim_hier>.<identifier>
| <member>.<identifier>
| <member_value_expression>
<property> ::= <mandatory_property> | <user_defined_property>
<mandatory_property> ::= CATALOG_NAME
| SCHEMA_NAME
| CUBE_NAME
| DIMENSION_UNIQUE_NAME
| HIERARCHY_UNIQUE_NAME
| LEVEL_UNIQUE_NAME
| LEVEL_NUMBER
| MEMBER_UNIQUE_NAME
| MEMBER_NAME
| MEMBER_TYPE
| MEMBER_GUID
| MEMBER_CAPTION
| MEMBER_ORDINAL
| CHILDREN_CARDINALITY
| PARENT_LEVEL
| PARENT_UNIQUE_NAME
| PARENT_COUNT
| DESCRIPTION
<user_defined_property> ::= <dim_hier>.<identifier>
| <level>.<identifier>
| <member>.<identifier>
<tuple> ::= <member>
| (<member> [, <member>...])
| <tuple_value_expression>
<set> ::= <member>:<member>
| <set_value_expression>
| <open_brace>[<set>|<tuple> [, <set>|<tuple>...]]<close_brace>
| (<set>)
<open_brace> ::= {
<close_brace> ::= }
<open_bracket> ::= [

<close_bracket> ::=]
<open_parenthesis> ::= (
<close_parenthesis> ::=)
<underscore> ::= _
<alpha_char> ::= a | b | c | ...| z | A | B | C | ... | Z
<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<value_expression> ::= <numeric_value_expression>
| <string_value_expression>
<numeric_value_expression> ::= <term>
| <numeric_value_expression> {<plus | <minus>} <term>
<term> ::= <factor> | <term> {<asterisk> | <solidus> | **<exponentiation>** | **<root>**} <factor>
<factor> ::= [<sign>] <numeric_primary>
<sign> ::= + | -
<plus> ::= +
<minus> ::= -
<asterisk>::= *
<solidus> ::= /
<exponentiation> ::= ^
<root> ::= root
<numeric_primary> ::= <value_expression_primary>
| <numeric_value_function>
<value_expression_primary> ::= <unsigned_numeric_literal>
| (<value_expression>)
| <character_string_literal>
| [<cube_name>.]<tuple>[.VALUE]
| <property>[.VALUE]
| <conditional_expression>
<conditional_expression> ::= <if_expression> | <case_expression>
<if_expression> ::= IIF(<search_condition>, <true_part>, <false_part>)
<true_part> ::= <value_expression>
<false_part> ::= <value_expression>
<case_expression> ::= <simple_case> | <searched_case> | <coalesce_empty>
<simple_case> ::= CASE <case_operand>
<simple_when_clause>...
[<else_clause>]
END
<searched_case> ::= CASE
<searched_when_clause>...
[<else_clause>]
END
<simple_when_clause> ::= WHEN <when_operand> THEN <result>
<searched_when_clause> ::= WHEN <search_condition> THEN <result>
<else_clause> ::= ELSE <value_expression>
<case_operand> ::= <value_expression>
<when_operand> ::= <value_expression>
<result> ::= <value_expression>
<coalesce_empty> ::= COALESCEEMPTY (<value_expression> ,
<value_expression>
[, <value_expression>]...)
<unsigned_numeric_literal> ::= <exact_numeric_literal>
| <approximate_numeric_literal>
<exact_numeric_literal> ::= <unsigned_integer>[.<unsigned_integer>]
| <unsigned_integer>.
| .<unsigned_integer>
<unsigned_integer> ::= {<digit>}...
<approximate_numeric_literal> ::= <mantissa>E<exponent>
<mantissa> ::= < exact_numeric_literal>
<exponent> ::= [<sign>]<unsigned_integer>
<string_value_expression> ::= <value_expression_primary>
| <string_value_expression>

<concatenation_operator>
<value_expression_primary>
<character_string_literal>::= <quote>[<character_representation>...]
<quote>
<character_representation> ::= <nonquote_character> | <quote_symbol>
<nonquote_character> ::= !!
Any character in the character set other than <quote>
<quote_symbol> ::= <quote> <quote>

<quote> ::= '
<concatenation_operator> ::= ||
<index> ::= <numeric_value_expression>
<percentage> ::= <numeric_value_expression>
<set_value_expression> ::= <dim_hier>.MEMBERS
| <level>.MEMBERS
| <member>.CHILDREN
| BOTTOMCOUNT(<set>, <index>
[, <numeric_value_expression>])
| BOTTOMPERCENT(<set>, <percentage>,
<numeric_value_expression>)
| BOTTOMSUM(<set>, <numeric_value_expression>,
<numeric_value_expression>)
| CROSSJOIN(<set>, <set>)
| DESCENDANTS(<member>, <level> [,<desc_flags>])
| DISTINCT(<set>)
| DRILLDOWNLEVEL(<set> [, <level>])
| DRILLDOWNLEVELBOTTOM(<set>, <index>
[,[<level>], <numeric_value_expression>])
| DRILLDOWNLEVELTOP(<set>, <index>[, [<level>]
, <numeric_value_expression>])
| DRILLDOWNMEMBER(<set>, <set>[, RECURSIVE])
| DRILLDOWNMEMBERBOTTOM(<set>, <set>, <index>
[, <numeric_value_expression>], RECURSIVE])
| DRILLDOWNMEMBERTOP(<set>, <set>, <index>
[, [<numeric_value_expression>], RECURSIVE])
| DRILLUPLEVEL(<set>[, <level>])
| DRILLUPMEMBER(<set>, <set>)
| EXCEPT(<set>, <set> [, [ALL]])
| EXTRACT(<set>, <dim_hier>[, <dim_hier>...])
| FILTER(<set>, <search_condition>)
| GENERATE(<set>, <set> [, [ALL]])
| HIERARCHIZE(<set>)
| INTERSECT(<set>, <set> [, [ALL]])
| LASTPERIODS(<index> [, <member>])
| MTD([<member>])
| ORDER(<set>, <value_expression>
[, ASC | DESC | BASC | BDESC])
| PERIODSTODATE([<level>[, <member>]])
| QTD([<member>])
| TOGGLEDRILLSTATE(<set1>, <set2>[, RECURSIVE])
| TOPCOUNT(<set>, <index> [, <numeric_value_expression>])
| TOPPERCENT(<set>, <percentage>, <numeric_value_expression>)
| TOPSUM(<set>, <numeric_value_expression>, <numeric_value_expression>)
| UNION(<set>, <set> [, [ALL]])
| WTD([<member>])
| YTD(<member>)
| OPERATORDEFINITION (<string_value_expression>,
<string_value_expression>
[,PARAM <open_parenthesis>
<string_value_expression>...

<close_parenthesis>])
| HANALYSIS (<set>, <set> <set>[<set>] [,<unsigned_numeric_literal>...])
| VANALYSIS (<member>, <set>)
| CROSS (<member>[, <string_value_expression>...])
| NNEARESTVALUES (<member>, <unsigned_integer> [, ASC | DESC])
| NNEARESTVALUESPERCENTUAL (<member>, <unsigned_numeric_literal> [, ASC | DESC])

<desc_flags> ::= SELF
| AFTER
| BEFORE
| BEFORE_AND_AFTER
| SELF_AND_AFTER
| SELF_AND_BEFORE
| SELF_BEFORE_AFTER

<member_value_expression> ::= <member>.{PARENT | FIRSTCHILD | LASTCHILD
| PREVMEMBER | NEXTMEMBER}
| <member>.LEAD(<index>)
| <member>.LAG(<index>) | <member>.{FIRSTSIBLING | LASTSIBLING}
| <dimension>[.CURRENTMEMBER]
| <dimension>.DEFAULTMEMBER
| <hierarchy>.DEFAULTMEMBER
| ANCESTOR(<member>, <level>)
| CLOSINGPERIOD(<level>[, <member>])
| COUSIN(<member>, <member>)
| OPENINGPERIOD(<level>[, <member>])
| PARALLELPERIOD([<level>[, <index> [, <member>]]])

<tuple_value_expression> ::= <set>.CURRENTMEMBER
| <set>[.ITEM]({<string_value_expression>
[, <string_value_expression>...]}
| <index>)

<boolean_primary> ::= <value_expression> <comp_op> <value_expression>

alter_statement ::= <create_statement> | <remove_statement> | <move_statement> |
<update_statement>

<create_statement> ::= CREATE DIMENSION MEMBER <member_spec>,
KEY='<key_value>' [[, <property_name>='<value>'] [, <property_name>='<value>']...]

<remove_statement> ::= DROP DIMENSION MEMBER <member_spec> [WITH DESCENDANTS]

<move_statement> ::= MOVE DIMENSION MEMBER <member_spec>
[WITH DESCENDANTS]
UNDER <member_spec>

<update_statement> ::= UPDATE DIMENSION MEMBER <member_spec>
[AS '<mdx_expression>',] |
<property_name>='<value>' [[, <property_name>='<value>']...]

<numeric_value_function> ::=
AGGREGATE(<set> [, <numeric_value_expression>])
AVG(<set>[, <numeric_value_expression>])
CORRELATION(<set> , <numeric_value_expression> [, <numeric_value_expression>])
COVARIANCE(<set>, <numeric_value_expression> [, <numeric_value_expression>])
COUNT(<set>[, INCLUDEEMPTY])
LINREGINTERCEPT(<set>, <numeric_value_expression> [, <numeric_value_expression>])
LINREGPOINT(<numeric_value_expression>, <set>, <numeric_value_expression>
[,<numeric_value_expression>])
LINREGR2(<set>, <numeric_value_expression> [, <numeric_value_expression>])
LINREGSLOPE(<set>, <numeric_value_expression> [, <numeric_value_expression>])
LINREGVARIANCE(<set>, <numeric_value_expression> [, <numeric_value_expression>])
MAX(<set>[, <numeric_value_expression>])
MEDIAN(<set>[, <numeric_value_expression>])
MIN(<set>[, <numeric_value_expression>])
RANK(<tuple>, <set>)
STDEV(<set>[, <numeric_value_expression>])
SUM(<set>[, <numeric_value_expression>])

VAR(<set>[, <numeric_value_expression>])

SEPARATRIX (<set>, <member>, <unsigned_integer>)

<search_condition> ::= <boolean_term> | <search_condition> {OR | XOR} <boolean_term>

<boolean_term> ::= <boolean_factor> | <boolean_term> AND <boolean_factor>

<boolean_factor> ::= [NOT] <boolean_primary>

<boolean_primary> ::= <value_expression> <comp_op> <value_expression>
| ISEMPTY(<value_expression>)
| (<search_condition>)

<comp_op> ::= <equals_operator>
| <not_equals_operator>
| <less_than_operator>
| <greater_than_operator>
| <less_than_or_equals_operator>
| <greater_than_or_equals_operator>

<equals_operator> ::= =

<not_equals_operator> ::= <>

<greater_than_operator> ::= >

<less_than_operator> ::= <

<greater_than_or_equals_operator> ::= >=

<less_than_or_equals_operator> ::= <=