Syntax of the SECONDO Optimizer Query Language

In the sequel, we give a short grammar for the SECONDO Optimizer Query Language (SECONDO-SQL). This language can be used directly with the sql/1 predicate of the optimizer (`SecondoPL`), or with a running `OptimizerServer` from the `JavaGUI`.

All attribute and database objects must be stated using lower case characters only. Indexes need to have canonic names, for details consult the more extensive explanations in files `$SECONDO_BUILD_DIR$/Optimizer/optimizer.pl` and `$SECONDO_BUILD_DIR$/Optimizer/database.pl`.

Entry point is <sql-clause>.

<sql-clause>	::=	**let** <objectname> <mquery>**.** | **let(** <objectname> **,** <mquery> **,** <secondo-rest-query> **).** | **sql** <mquery> **.** | **sql(** <mquery> **,** <secondo-rest-query> **).**
<aggr>	::=	<groupattr> | <groupattr> **as** <newname> | <aggr2>
<aggr2>	::=	**count(**<distinct-clause> ***) as** <newname> | <aggrop> **(** <ext-attr-expr> **) as** <newname> | **aggregate(** <ext-attr-expr> **,** <aggrfun> **,** <datatype> **,** <const> **) as** <newname> *% the* <const> *should be the neutral element with respect to the function* <aggrfun>*, that is used to fold the result into a single value and the* <datatype> *it operates on.*
<aggrop>	::=	**min** | **max** | **sum** | **avg** | **extract** | **count**
<aggr-clause>	::=	<aggr> | **[** <aggr> **,** <aggr-list> **]** *% The list must contain at least one* <aggr2> *(calculated value), i.e. it is not allowed for a* <aggr-clause> *to be formed solely by grouping attributes.*
<aggr-fun>	::=	**(*)** | **(+)** | **union_new** | **intersection_new** | ... *% any name* fun *of a binary SECONDO-operator or function object with syntax* fun: T x T --> T *which should be associative and commutative. Infix-operators must be enclosed in round paranthesis.*
<aggr-list>	::=	<aggr> | <aggr> **,** <aggr-list>
<attr>	::=	<attrname> | <var> **:** <attrname> | **rowid** *% the colon "***:***" is used instead of the dot "***.***" in standard SQL, separating a relation name (or variable name) from an attribute name.* **rowid** *adds an attribute named rowid with the row number to each result tuple. It should only be applied to results coming directly from a selection, i.e. no join predicate is allowed in the query's* <where-clause>*. The type of attribute rowid is* `tid` *(tuple identifier).*
<attr-list>	::=	<attr> | <attr> **,** <attr-list>
<attrname>	::=	<id>
<bool-const>	::=	**true** | **false**

<char>	::=	*% Any character. Non-ASCII-characters may cause problems.*
<column>	::=	<newname> **:** <datatype>
<column-list>	::=	<column> | <column> **,** <column-list>
<compop>	::=	**<** | **<=** | **=** | **>=** | **>** | **#** | **<>**
<const>	::=	<bool-const> | <int-const> | <real-const> | <string-const> | <text-const> | <generic-const>
<create-query>	::=	**create table** <newname> **columns [** <column-list> **]**
		| **create index on** <relname> **columns** <index-clause>
<datatype>	::=	**int** | **real** | **bool** | **string** | **line** | **points** | **mpoint** | **uregion** | ...
		% any name of a SECONDO-datatype
<delete-query>	::=	**delete from** <rel-clause> <where-clause>
<digit>	::=	**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
<distinct-clause>	::=	**all** | **distinct** | **ε**
<drop-query>	::=	**drop table** <relname>
		| **drop index** <indexname>
		| **drop index on** <relname> <indexclause>
<ext-attr>	::=	<distinct-clause> <attr>
<ext-attr-expr>	::=	<distinct-clause> <attr-*expr*>
<first-clause>	::=	**first** <int-const> | **last** <int-const> | **ε**
<generic-const>	::=	**[const,** <sec-type-expr>**, value,** <sec-value> **]**
		% Where <sec-type-exp> *is a valid Secondo type-expression in text-syntax, e.g. vector(int), and* <sec-value-expr> *is an according value in nested list format.*
		Can also be used to create undefined <{bool|int|real|string|text}-constant>*s.*
<groupattr>	::=	<attr>
<groupattr-list>	::=	<groupattr> | <groupattr> **,** <groupattr-list> | **ε**
<groupby-clause>	::=	**groupby [** <groupattr-list> **]** | **groupby** <groupattr>
<id>	::=	<letter-lc>$^+$
		% Identifiers have a maximum length of 48 characters. Using Prolog keywords as <id> may result in parse errors!
<indexname>	::=	<id>
<indextype>	::=	**btree** | **rtree** | **hash** | …
		% any name of a logical index type
<index-clause>	::=	<attrname> | <attrname> **indextype** <indextype>
<insert-query>	::=	**insert into** <rel> **values** <value-list> | **insert into** <rel> <query>
<int-const>	::=	{**+** | **-** | **ε**} <digit>$^+$
<letter-lc>	::=	**a** | **b** | **c** | ... | **x** | **y** | **z**
<mquery>	::=	<query>
		| <insert-query>
		| <delete-query>
		| <update-query>
		| <create-query>
		| <drop-query>
		| **union [** <query-list> **]**
		| **intersection [** <query-list> **]**
<newname>	::=	<id>
		% where <id> *is not already defined within the database or the current query*

<orderattr>	::=	<attrname> | <attrname> **asc** | <attrname> **desc** | **distance(** <id>, <id> **)**
<orderattr-list>	::=	<orderattr> | <orderattr> **,** <orderattr-list>
<orderby-clause>	::=	**orderby [** <orderattr-list> **]** | **orderby** <orderattr> | **ε**
<pred>	::=	<attr-*boolexpr*> | <subquerypred>[1]
<pred-list>	::=	<pred> | <pred> **,** <pred-list>
<query>	::=	**select** <distinct-clause> <sel-clause> **from** <rel-clause> <where-clause> <orderby-clause> <first-clause> | **select** <aggr-clause> **from** <rel-clause> <where-clause> <groupby-clause> <orderby-clause> <first-clause>
<query-list>	::=	<query> | <query> **,** <query-list>
<real-const>	::=	{**+** | **-** | **ε**} <digit>$^{+}$ **.** <digit> { **E** {**+** | - | **ε**} <digit>$^{+}$ | **ε**}
<rel>	::=	<relname> | <relname> **as** <var>
<rel-clause>	::=	<rel> | **[** <rel-list> **]**
<rel-list>	::=	<rel> | <rel> **,** <rel-list>
<relname>	::=	<id>
<result>	::=	<attr> | <attr-expr> **as** <newname>
<result-list>	::=	<result> | <result> **,** <result-list>
<secondo-rest-query>	::=	**'** <text> **'**
<sel-clause>	::=	<sel-clause2> | **nonempty** <sel-clause2>[2]
<sel-clause2>	::=	***** | <result> | **[** <result-list> **]** | **count(** <distinct-clause> ***)** | <aggrop> **(** <ext-attr-expr> **)** | **aggregate(** <ext-attr-expr> **,** <aggrfun> **,** <datatype> **,** <const> **)** *% the types of <const> and the parameter and result types of <aggrfun> must be equal to <dattype>.*
<string-const>	::=	**“** <char>* **”**
<subquerypred>	::=	<attr-*expr*> **in (** <table-subquery> **)** | <attr-*expr*> **not in (** <table-subquery> **)** | **exists (** <query> **)** | **not exists (** <query> **)** | <attr-*expr*> <compop> **any (** <table-subquery> **)** | <attr-*expr*> <compop> **some (** <table-subquery> **)** | <attr-*expr*> <compop> **all (** <table-subquery> **)**
<table-subquery>	::=	*% any nested* <query> *creating a single-column-table. The subquery's* <where-clause> *may refer to attributes from the outer sql query (forming a so-called correlated subquery).*
<text>	::=	*% any sequence of characters, that completes the optimized query to a valid expression in Secondo executable language*
<text-const>	::=	% TODO!
<transform>	::=	<attrname> **=** <update-expression>
<transform-clause>	::=	<transform> | **[** <transform-list**]**
<transform-list>	::=	<transform> | <transform> **,** <transform-list>
<update-expression>	::=	<const> | <const>-expr *% a constant value, or a term calculating a value.*

1 Alternative <subquerypred> requires activation of `optimizerOption(subqueries)`.
2 Using **nonempty** requires optimizerOption(rewriteNonempty) to be active.

<update-query>	::=	**update** <relname> **set** <transform-clause> <where-clause>
<var>	::=	<id>
<value>	::=	*% a valid* `integer`*,* `boolean` *or* `string` *value in Prolog*
<value-list>	::=	<value> | <value> **,** <value-list>
<where-clause>	::=	**where [** <pred-list> **]** | **where** <pred> | **ε**

Notes

<N-*expr*> means any expression formed by operators, constants (<const>, and N-elements.

<N-*boolexpr*> means any expression formed by operators, constants, and N-elements. returning a value of type `bool`.

ε means the empty word (i.e. the defined element may be omitted).

Terminal symbols are printed in **bold face** font.

Alternatives are separated by the vertical bar “ | ”.

Unconsidered Query Language Elements

The grammar given above does still not consider the following extensions to the Secondo Optimzer:

- **macros**