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`.

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**,** datatype-*constant***) as** newname
aggrop	-->	**min** | **max** | **sum** | **avg** | **extract** | **count**
aggr-clause	-->	aggr | **[**aggr, aggr-list**]**
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**
attr-list	-->	attr | attr**,** attr-list
attrname	-->	id
column	-->	newname **:** datatype
column-list	-->	column | column, column-list
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
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-*constant* | **last** int-*constant* | **ε**
groupattr	-->	attr
groupattr-list	-->	groupattr | groupattr, groupattr-list | **ε**
groupby-clause	-->	**groupby [**groupattr-list**]** | **groupby** groupattr
id	-->	% any valid Prolog constant-identifier without any underscore-

character

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

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*

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

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 --> *****
| result | **[**result-list**]**
| **count(**distinct-clause ***)**
| aggrop**(**ext-attr-expr**)**
| **aggregate(**ext-attr-expr**,** aggrfun**,** datatype**,** datatype-*constant***)**

text --> % any sequence of characters, that completes the optimized query to a valid expression in Secondo executable language

transform --> attrname **=** update-expression

transform-clause --> transform | **[**transform-list**]**

transform-list --> transform | transform**,** transform-list

update-expression --> % a fixed value, or an operation calculating a value

update-query --> **update** relname **set** transform-clause where-clause

var --> id

value --> % an integer, boolean or string value in prolog

value-list --> value | value**,** value-list

where-clause --> **where [**pred-list**]** | **where** pred | **ε**

Unconsidered Query Language Elements

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

- **macros**
- **nonempty** within select-clauses
- subqueries