1 Keywords of GQL

Here is the list of keywords of GQL. Note that the keywords are also recognized when a user enters them with upper case characters.

- both_mates
- include
- print
- genome
- select
- from
- mapjoin
- where
- table
- and
- or
- not
- integer
- float
- char
- string
- using
- interval_creation
- create_intervals
- merge_intervals
- intervals

2 The GQL grammar

⟨program⟩→ ⟨table_prototypes⟩ ⟨genome_names⟩ ⟨assigned_selects⟩ ⟨print_statements⟩

⟨table_prototypes⟩→ ⟨table_prototype⟩ ⟨table_prototypes⟩|ϵ

⟨table_prototype⟩→ ⟨table_keyword⟩ **(** ⟨table_args⟩ **) ;**

⟨table_keyword⟩→ **table** ⟨names⟩

⟨names⟩→ **ID**

⟨numbers⟩→ ⟨digit⟩ ⟨numbers⟩| ⟨digit⟩

⟨digit⟩→ **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

⟨table_args⟩→ ⟨table_args⟩ **,** ⟨table_arg⟩

| ⟨table_arg⟩

⟨table_arg⟩→ **integer** ⟨names⟩

| **float** ⟨names⟩

| **char** ⟨names⟩

| **string** ⟨names⟩

⟨genome_names⟩→ ⟨genome_names⟩ ⟨genome_names⟩

| **genome** ⟨names⟩ **;**

| **genome** ⟨names⟩ *** ;**

| **genome * where** ⟨where_args⟩ **;**

⟨assigned_selects⟩→ ⟨assigned_select⟩ ⟨assigned_selects⟩ | ϵ

⟨assigned_select⟩→ ⟨lvalue⟩ **=** ⟨select_statement⟩

⟨lvalue⟩→ ⟨names⟩

⟨select_statement⟩→ **select** ⟨select_args⟩ **from** ⟨compound_from_arg⟩ **where** ⟨where_args⟩

| **select** ⟨select_intrvl_args⟩ **from** ⟨compound_from_arg⟩

| **select** ⟨select_args⟩ **from mapjoin** ⟨from_args⟩

⟨select_intrvl_args⟩→ **create_intervals ( )**

| **merge_intervals (** ⟨names⟩ > ⟨numbers⟩ **)**

| **merge_intervals (** ⟨names⟩ < ⟨numbers⟩ **)**

⟨select_args⟩→ *****| ⟨select_arg_series⟩

⟨select_arg_series⟩→ ⟨names⟩ **,** ⟨select_arg_series⟩

| ⟨names⟩

| ⟨obj_names⟩

⟨from_args⟩→ ⟨compound_from_arg⟩ **,** ⟨from_args⟩

| ⟨compound_from_arg⟩

⟨compound_from_arg⟩→ ⟨from_arg⟩

| ⟨from_arg⟩ **using intervals (** ⟨arith_expr⟩ **,** ⟨arith_expr⟩ **)**

| ⟨from_arg⟩ **using intervals (** ⟨arith_expr⟩ **,** ⟨arith_expr⟩ **, both_mates)**

⟨from_arg⟩→ ⟨names⟩

⟨where_args⟩→ ⟨where_args⟩ **,** ⟨where_args⟩

| ⟨where_args⟩ **and** ⟨where_args⟩

| ⟨where_args⟩ **or** ⟨where_args⟩

| **not** ⟨where_args⟩

| **(** ⟨where_args⟩ **)**

| ⟨lowest_expr⟩

⟨lowest_expr⟩→ ⟨arith_expr⟩ ⟨comparison_op⟩ ⟨arith_expr⟩

⟨arith_expr⟩→ ⟨arith_expr⟩ ⟨arith_op⟩ ⟨arith_expr⟩

| ⟨arith_op⟩ ⟨arith_expr⟩

| ⟨names⟩ **(** ⟨arith_expr⟩ **)**

| ⟨names⟩

| ⟨obj_names⟩

| ⟨numbers⟩

| **const_string**

| **const_char**

⟨obj_names⟩→ ⟨names⟩ **.** ⟨names⟩

⟨comparison_op⟩→ == | >= | <= | <

⟨arith_op⟩→ + | **-** | ***** | **/** | %

⟨print_statements⟩→ ⟨print_statement⟩ ⟨print_statements⟩| ϵ

⟨print_statement⟩→ **print** ⟨names⟩

| **print** ⟨names⟩ **both_mates**

3 THE GQL user guide

3.1 The environment

The installation of GQL requires that the following variables are set.

1. SAMLOC should point to the installation directory of samtools
2. DONOR_DIR should point to the parent directory of donor genomes.
3. BIOSQL_HOME should point to the installation directory of GQL
4. PATH should contain SAMLOC.

3.2 GQL projects

GQL organizes experiments into projects. All donors that participate in a project are assumed to have been aligned against the same reference and their chromosome naming follows the same conventions. When a user creates a project with name P, src_tables_dir S and destination directory d, they need to move all donors of the project under under $DONOR_DIR/P, all text tables under S/P and all products of the queries that use this project will be found under d/P.

3.3 Text tables

Text tables are tab delimited text files that should be placed under S/P for project P with src_tables S. A table with name T should have a file name T.txt. The first line of a Text table file should always contain the following content: *#table T (type var1, type var2...);* where T is the table name, type should be any of {integer, string, char, float} and var should be any string of characters. Note that at least a field is expected to contain chromosome information. **ATTENTION** for proper functionality, the table name of the first line should match the file name without the .txt suffix. The rest lines of a text table file can be either comment lines that begin with '#' or data.

3.4 Reads Table

The *Reads* table is embedded for every genome and in the current prototype a user can access the following fields.

- integer location –the mapping coordinate of the read
- integer length – the read length
- char strand – the mapping strand
- integer mate_loc – the mapping location of the other end

- mate_strand – the mapping strand of the other end.

Note that the user does not have access to the chromosome field because a query runs over all chromosomes iteratively.

3.5 program structure.

Each GQL program consists of 4 distinct parts:

1. The *include* line
2. The donor selection lines
3. A set of select statement assignments
4. A set of print statements.

The include line. Currently the first line of each program should be "include <tables.txt>" which loads the table definitions.

Donor selection. The user selects on which genomes the query should run on. The choice can either be explicit, e.g *genome NA18507, NA18506;* or it can be an expression that contains a wild-card, such as: *genome NA185*;* or an SQL like statement such as *genome * where sex=="male" and population=="YRI"*. Note that for the latter to be functional a user needs to upload a text table with name *genome_info* that should contain an entry for each donor genome with fields that depend on the information that is available for each donor. The only restriction is that the field that corresponds to a donor name should be labeled as *name*.

Select statement assignments. Each statement should be of the form *variable = select statement*. After the execution of a select statement that left variable is a table of the same type as the return of the select statement and it can be used as input in subsequent statements. There are four types of possible select statements.

- A selection statement of the form:

  *select * from Input-table where where_expr*

  where Input-table can be any table within the scope of the query and where_expr can be any boolean function of the attributes of Input-table. The statement returns a table of the same type and the same arguments as the Input-table.

- An interval creation of the form:

  select create_intervals() from Input-table using intervals (expr1, expr2,[both_mates]).

  This returns a table of Intervals with attributes *begin, end, chromosome* that is created on Input-table. expr1 and expr2 can be integer expressions on any field of Input-table and they specify the begin and end fields respectively. If Input-table is of type *Reads* a user can add the *both_mates* flag which specifies that both ends of a pair end participate in each interval.

- An interval merging of the form:

  select merge_intervals(interval count $\leqslant$ *constant) from Input-table.*

  The input has to be of type *Intervals* and the output is also of type *Intervals*.

- A MAPJOIN operation of the form:

  *select * from mapjoin Input-tableA using intervals(expr1, expr2, [both_mates]), Input-tableB using intervals(expr3, expr4, [both_mates]).*

  Both input tables can be of any type and the resulting table contains the attributes of Input-tableA followed by attributes of Input-tableB. In case there are naming conflicts we append suffix 1 to the resulting entry that originates from table A and suffix 2 to the resulting table from table B.

Print Statement A print statement consists of the keyword *print* followed by a table name. If the table name is of type *Reads* it can be followed by the *both_mates* keyword to let GQL know that reads need to be printed as pair ends. Below we summarize the output of GQL depending on the type of table.

- Reads. The output consists of a .bam file that contains all reads that a table contains, a .bam.short file that is a text file that contains only the starting, ending mapping coordinate and the orientation of each read, a .hist file which is used by our GUI to plot histogram related information and a .bam.links.txt which displays the correspondence of the current output with the initial bam file. The latter contains pairs of integers where the first integer corresponds to the location of a read in the output bam file and the other integer corresponds to the location of the same read in the original bam file. In case keyword *both_mates* is present the output .bam.short file is organized such that a line contains the starting and ending mapping coordinates of both reads.

- Intervals. The output contains a .interval text file where each row represents an interval. It also contains a .interval.links.txt file with the same functionality as in Reads.

- Text table. The output contains a .txt file which is a subset of the initial input and a .links.txt which connects the output entries with the original ones of the source.
- Joined table, which is any table that has been the result of one or more MAPJOIN operations. The output products consist of individual entrees of tables of the above basic types and also a .links.txt file which shows what entries of the leftmost entry of the MAPJOIN pairs with the rightmost ones.