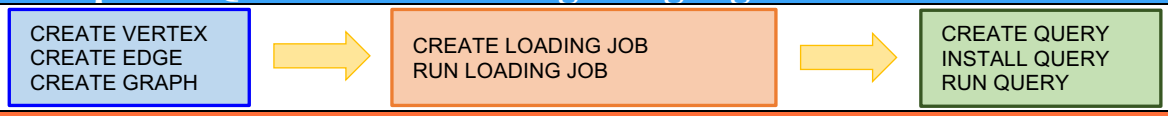
TigerGraph GSQL DDL & Loading Language v2.2 Reference Card

Workflow



Define a Schema

```
DROP ALL      # erases all graph and job definitions, and clears graph store
CREATE VERTEX vname (PRIMARY_ID id type [, attribute_name type [DEFAULT default_value] ]* )
  [WITH STATS="none"|"outdegree_by_edgetype"]
CREATE UNDIRECTED EDGE ename (FROM vname1, TO vname2 [, attribute_name type [DEFAULT default_value]]* )
CREATE DIRECTED EDGE name (FROM vname1, TO vname2 [, attribute_name type [DEFAULT default_value]]* )
  [WITH REVERSE_EDGE="rname"]
CREATE GRAPH gname (*)

USE GRAPH gname       # set gname to be the active graph
USE GLOBAL
DROP GRAPH gname
```

Attribute Types

type: INT | UINT | FLOAT | DOUBLE | BOOL | STRING | STRING COMPRESS | FIXED_BINARY(n) | DATETIME | UDT | LIST<elementType> | SET<elementType> | MAP<keyType, valueType>

UDT: TYPEDEF TUPLE<f1 INT(b), f2 UINT, f4 STRING(n)> tupleName

LIST|SET element and MAP value type: INT, DOUBLE, STRING, STRING COMPRESS, DATETIME, UDT

MAP keyType: INT, STRING, STRING COMPRESS, DATETIME

Schema Change – Modify Local Vertex/Edge Types

```
CREATE SCHEMA_CHANGE JOB job_name FOR GRAPH gname {
  [sequence of LOCAL DROP, ALTER, and ADD statements, each line ending with a semicolon]
}
RUN JOB job_name;
```

```
  ADD VERTEX vname (PRIMARY_ID id type …)  // same syntax as CREATE VERTEX
  ADD UNDIRECTED EDGE ename (FROM vname1…) // same syntax as CREATE UNDIRECTED EDGE
  ADD DIRECTED EDGE ename (FROM vname1… )  // same syntax as CREATE DIRECTED EDGE
  ALTER VERTEX|EDGE name ADD (attribute_name type DEFAULT default_value]
                             [, attribute_name type [DEFAULT default_value]]* );
  ALTER VERTEX|EDGE name DROP (attribute_name [, attribute_name]* );
  DROP VERTEX vname [, vname]*;
  DROP EDGE ename [, ename]*;
```

Schema Change – Modify or Assign Global Vertex/Edge Types

```
CREATE GLOBAL SCHEMA_CHANGE JOB job_name {
  [sequence of GLOBAL DROP, ALTER, and ADD statements, each line ending with a semicolon]
}
RUN JOB job_name;
```

```
  ADD VERTEX vname TO GRAPH gname // assigns an existing global vertex type to a graph
  ADD EDGE ename   TO GRAPH gname //
  ALTER VERTEX|EDGE name ADD (attribute_name type DEFAULT default_value]
                             [, attribute_name type [DEFAULT default_value]]* );
  ALTER VERTEX|EDGE name DROP (attribute_name [, attribute_name]* );
  DROP VERTEX vname FROM GRAPH gname;  // removes a global vertex type from a graph
  DROP EDGE ename   FROM GRAPH gname;
```


DCN GDLRC220-190102

Create a LOADING JOB block

```
CREATE LOADING JOB job_name FOR GRAPH gname {
  [zero or more DEFINE statements]
  [zero or more LOAD statements] | [zero or more DELETE statements]
}
```

DEFINE statements:

```
  DEFINE FILENAME fileVar [= filePath];
    filePath = (path | "all:"path | "any:"path | mach_aliases":"path ["," mach_aliases":"path]* )
    mach_aliases = list of machine aliases, e.g,, m1,m3

  DEFINE HEADER header_name = "column_name"[,"column_name"]*;
  DEFINE INPUT_LINE_FILTER filter_name = boolean_expression_using_column_variables;
```

LOAD statements:

```
  LOAD (fileVar|filepath_string|TEMP_TABLE tname) Destination_Clause [,Destination_Clause]*
  [USING Parsing_Conditions];
```

DELETE statement:

```
  DELETE VERTEX vname (PRIMARY_ID id_expr) FROM (fileVar|filePath) [WHERE condition];
  DELETE EDGE ename (FROM id_expr [, TO id_expr]) FROM (fileVar|filePath) [WHERE condition];
  DELETE EDGE * (FROM id_expr vname) FROM (fileVar|filePath) [WHERE condition];
```

Destination_Clause: TO VERTEX|EDGE name VALUES (*id_expr* [,*attr_expr*]*)[WHERE *conditions*]
TO TEMP_TABLE name (id_name [,attr_name]*) VALUES (*id_expr* [,*attr_expr*]*)[WHERE *conditions*]

Parsing_Conditions: parameter=value [parameter=value]*

```
  SEPARATOR=sChar                    HEADER="true"|"false"
  EOL=eChar
  QUOTE="single"|"double"              USER_DEFINED_HEADER="true"|"false"
  REJECT_LINE_RULE=filter_name         JSON_FILE="true"|"false"
```

id_expr: attr_expr|REDUCE(*reducer_func_name*(*attr_expr*))
attr_expr: $1|$"column_name"|*token_func_name*(*attr_expr*[, *attr_expr*]*)
attr_expr for UDT: TupleName($1, $2, …)
attr_expr for LIST|SET: $1 | SPLIT($1, ",")
attr_expr for MAP: $1 -> $2 | SPLIT($1, ",") | SPLIT($1, ",", ":")
token_func_name: see Language Reference "Built-in Loader Token Functions"
reducer_func_name: max, min, add, and, or
WHERE condition:
Operators: +, -, *, /, <, >, ==, !=, <=, >=, AND, OR, NOT, IS NUMERIC, IS EMPTY, IN, BETWEEN..AND

Load Data and Manage Loading Jobs

```
CLEAR GRAPH STORE [-HARD]  # erases all graph data. Note: DROP GRAPH & DROP ALL do this automatically.

RUN LOADING JOB [loading_options] job_name [USING fileVar[=filePath] [, fileVar[=filePath]]* ]
```

loading_options

```
  -n [firsLineNum,] lastLineNum
  -dryrun
  -noprint
```

```
SHOW LOADING STATUS jobId|ALL

ABORT LOADING JOB jobId|ALL

RESUME LOADING JOB jobId
```


DCN GDLRC220-190102