Enabling Cost-Based Optimizer

This page walks you through the steps to enable the cost-based optimizer.

1. Before you begin

- You have created a query that contains a `SELECT` statement that uses a path pattern in its `FROM` clause with one or more hops.
- You have the privilege to install queries on the graph where you are operating.

2. Procedure

The following is the procedure to enable cost-based optimization when installing a query.

2.1. Identify referenced types and attributes

You need to identify the vertex and edge types for which you need to compute the following statistics:

- Cardinality: the number of instances for each type
- Histogram: the distribution of values for an attribute

2.1.1. Cardinality

You need to compute cardinality data for the following vertex and edge types:

- Any vertex type referenced in the path pattern of a `FROM` clause.
- Any *refined edge type*. A *refined edge type* means an edge type with a specified pair of `from` and `to` vertex types as well as their reverse edge types if they exist.

2.1.2. Histogram

You need to compute histograms for the following attributes:

- Any vertex attribute referenced in a `WHERE` clause in a `SELECT` statement in the query.

If you are not sure which types and attributes you need to compute histogram data for a query, you can try installing that query with the cost-based optimizer enabled first. The optimizer produces a helpful recommendation if it sees any types/attributes that don't have statistics computed for them.

For example, to optimize the following query:

```
CREATE OR REPLACE QUERY example3() FOR GRAPH ldbc_snb SYNTAX V2 {

 vSet = SELECT p
        FROM Tag:t - (<HAS_TAG) - Comment:m - (<LIKES:e) - Person:p
        WHERE p.gender == "female" AND t.id != 100;

}
```

You need the following statistics:

- Cardinality
  - Vertex types
    - `Tag`
    - `Comment`
    - `Person`
  - Refined edge types
    - `HAS_TAG`: from `Comment` to `Tag`
    - `HAS_TAG_REVERSE`: from `Tag` to `Comment`
    - `LIKES`: from `Person` to `Comment`
    - `LIKES_REVERSE`: from `Comment` to `Person`
- Histogram
  - `gender` attribute of type `Person`
  - `id` attribute of type `Tag`

2.2. Compute cardinality data

To compute cardinality data (the count of each type), use the `POST :14240/gsqlserver/gsql/stats/card` endpoint.

For example, to compute the cardinality data of the vertex `Person` in graph `ldbc_snb`, make the following request:

```
curl -s --user tigergraph:tigergraph -X POST
"localhost:14240/gsqlserver/gsql/stats/card?graph=ldbc_snb&vertex=Person"
```

For another example, to compute the cardinality data of the refined edge type `LIKES` from a `Person` vertex to a `Comment` vertex, make the following request:

```
curl -s --user tigergraph:tigergraph -X POST
"localhost:14240/gsqlserver/gsql/stats/card?
graph=ldbc_snb&edge=LIKES&from=Person&to=Comment"
```

For best results, compute the cardinality data for every type you identified in the previous step, each with its own request.

2.3. Compute histogram data

To compute histogram data, use the `POST :14240/gsqlserver/gsql/stats/histogram` endpoint.

For example, to compute the histogram data for attribute `gender` of vertex type `Person` on graph `ldbc_snb`, make the following request:

```
$ curl -s --user tigergraph:tigergraph -X POST
"http://localhost:14240/gsqlserver/gsql/stats/histogram?
graph=ldbc_snb&vertex=Person&attribute=gender&buckets=256"
```

For best results, compute the histogram data for every vertex attribute referenced in a `WHERE` clause of a `SELECT` statement.

2.4. Install query with `-cost` option

Now that the relevant statistics are available to the query optimizer, you can install the query using the cost-based query optimizer. To use the cost-based optimizer during installation, use the `-cost` option when installing the query.

For example, to install the query `example3`, run `INSTALL QUERY example3 -cost`.

Alternatively, set the `cost_opt` session parameter to true by running `set cost_opt = true` in the GSQL shell. This enables the query optimizer for the rest of your GSQL session. You no long need to supply the `-cost` option when installing queries to use the optimizer if the `cost_opt` parameter is set to true.

If you missed any cardinality or histogram data, the optimizer warns you about which types are missing, so you can compute the data for any missing type or attribute.