Database schema

The constraints provided in the problem description implies a Directed Acyclic Graph structure to the data. This can be modelled in various ways. The chosen method is known as an Adjacency list, which is essentially a join table between the Clusters table and itself.

Tables

Resources

id	int
code	varchar
desc	text
rate	decimal
unit	varchar

Clusters

id	int
name	varchar

ClusterResources

id	int
cluster_id	int
resource_id	int
transform_factor	int?

ClusterEdges

id	int
source_id	int
target_id	int

Cycle Detection

A graph with n vertices and at least n edges contains a cycle.[1]

---

1 https://ocw.mit.edu/high-school/mathematics/combinatorics-the-fine-art-of-counting/lecture-notes/MITHFH_lecturenotes_9.pdf (page 2, corollary 3.1)

Hence, add a trigger on insert to count the vertices and edges in the graph, and roll back the insert if a cycle is detected. The nodes can be counted with a query such as

```
1.1 WITH RECURSIVE pair(s,t) AS (
        SELECT source_id as s, target_id as t FROM arcs where source_id = 1
      UNION
        SELECT source_id, target_id from pair, arcs WHERE source_id = t)
      SELECT COUNT(DISTINCT t) from pair ORDER BY t ;
```

Similarly the edges can be found with

```
1.2 WITH RECURSIVE pair(s,t) AS (
        SELECT source_id as s, target_id as t FROM arcs where source_id = 1
      UNION
        SELECT source_id, target_id from pair, arcs WHERE source_id = t)
      SELECT COUNT(t) from pair;
```

Querying data

The vertex ids, corresponding to the cluster ids can be fetched with the query listed in 1.1, can be joined with the Clusters table, along with the ClusterResources and finally the Resources table, at which point the rate of the top level cluster can be calculated.

In practice it is better to construct the query in the application layer, but execute the queries in the database. Fetching the entire graph from the database, hydrating it into the correct memory structure, and then performing the required queries on the in memory structure is quite inefficient.

Concurrency

Concurrency on the database level is handled through the use of locking on the affected rows. This prevents concurrent writes on the same data by different parties.

In memory, concurrency can be made safer through the use of immutable data structures. Using a different language with better support for persistent data structures would improve efficiency in this regard.