Graph Model for Cassandra Schema Design

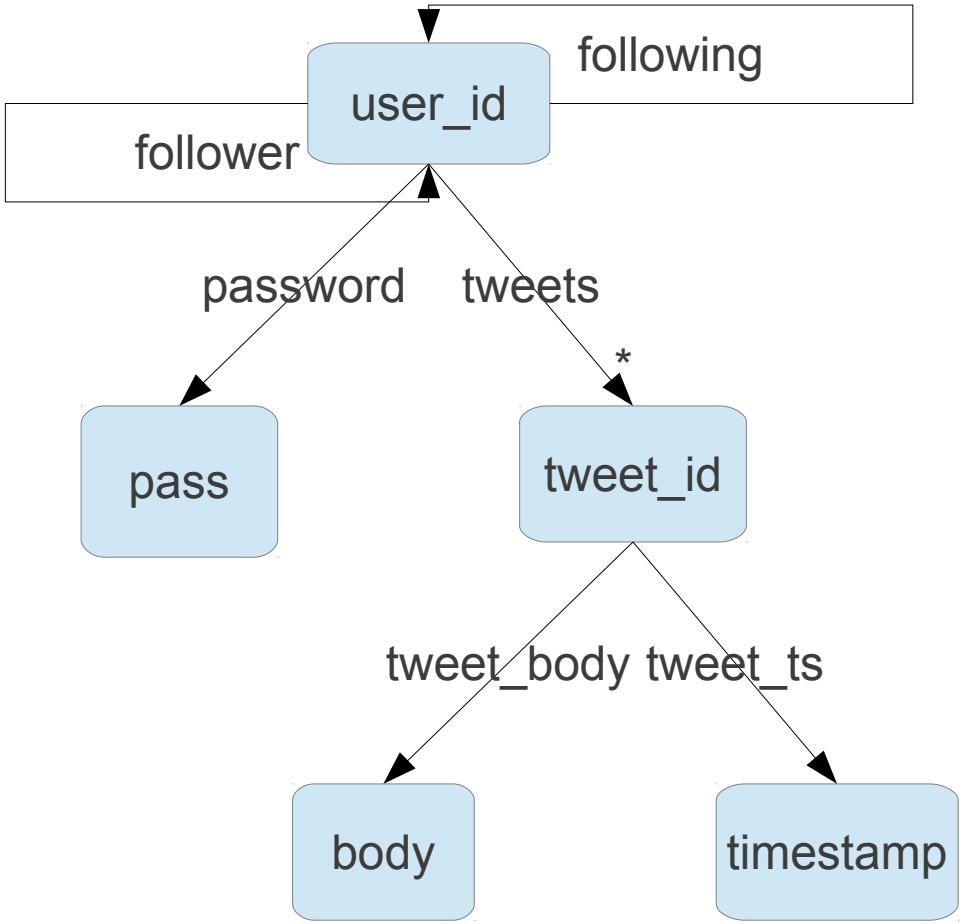
Graph model overview

- Why use graph model
  - We do not have a query optimizer, so we need to generate the query plan by ourselves
  - The queries cannot be expressed in SQL, because we do not have fixed schema or join.
- Solution space will be a closure (think about the Warshall's transitive closure) of graphs
- *Query* will be represented as (*source*, *sink*), or a path on the initial graph.
- A *query plan* is a path from *source* to *sink*, associated with a cost of the path

Notions of graph model

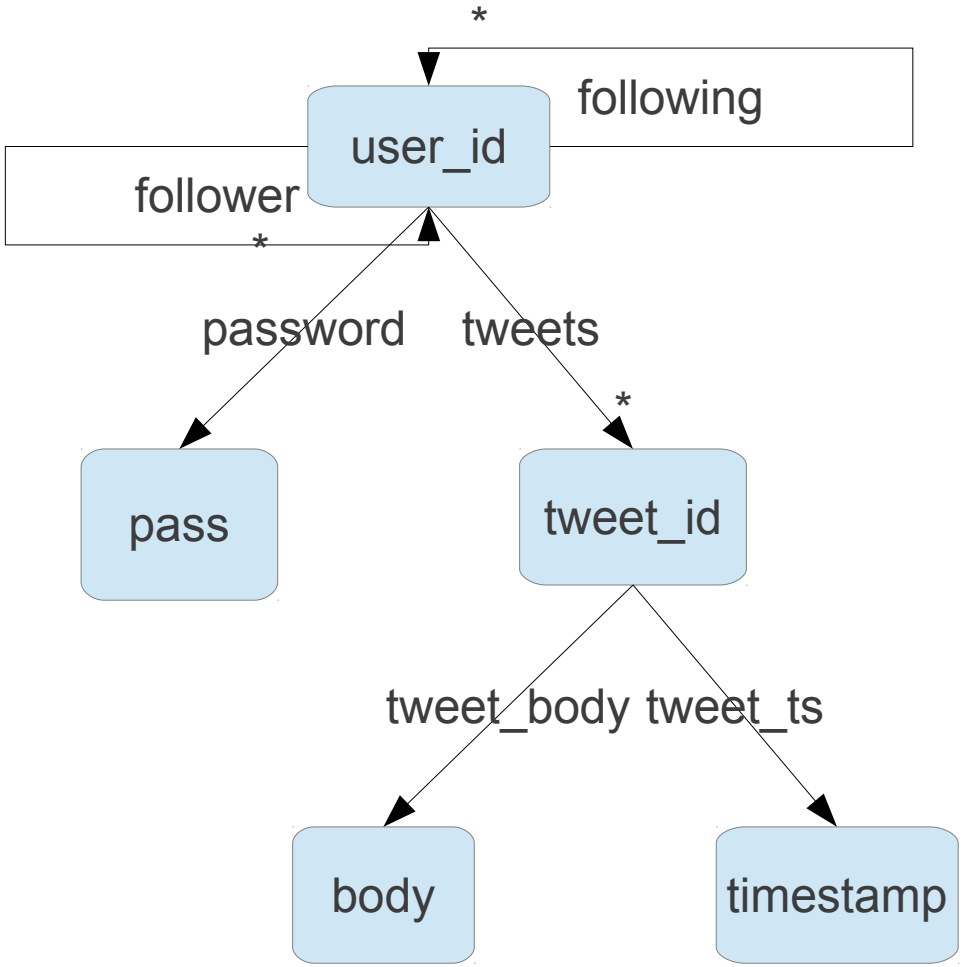
- An edge represents a column family called “tweets”
- *user_id* is row key
- *tweet_id* is in column name or value
- By looking up CF tweets with *user_id*, we can get *tweet_ids*
- Cardinality and cost are associated with the edge.

Initial graph



- There are six column families: follower, following, password, tweets, tweet_body, tweet_ts
- This graph is easily derivable from UML or ER chart

Workload (query) representation



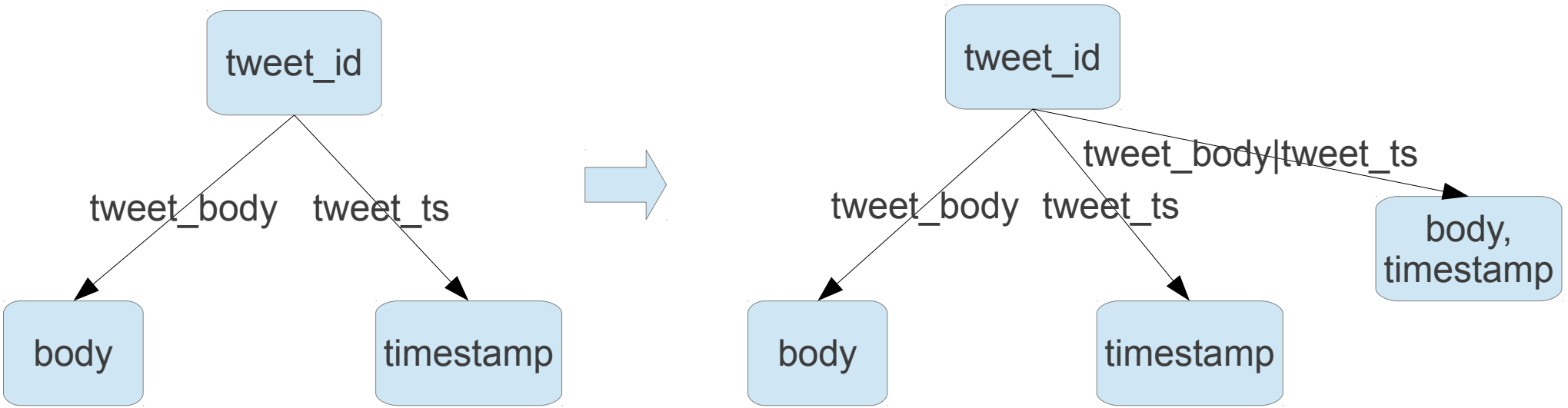
- Timeline query:
  - User_id -following-> user_id -tweets-> tweet_id -tweet_body-> body
  - User_id -following-> user_id -tweets-> tweet_id -tweet_ts-> timestamp
  - Sort body by timestamp

Enumerating the solution space

- Three operations, ops, to generate new edges and vertices: merge, in line, pull up
- G_0 is the initial graph.
- $G_{i+1} = G_i \cup \{op(v1,v2...) | v1, v2,.. \in \text{vertices of } G_i\}$
- Until there is no $cost_q$ can be further improved

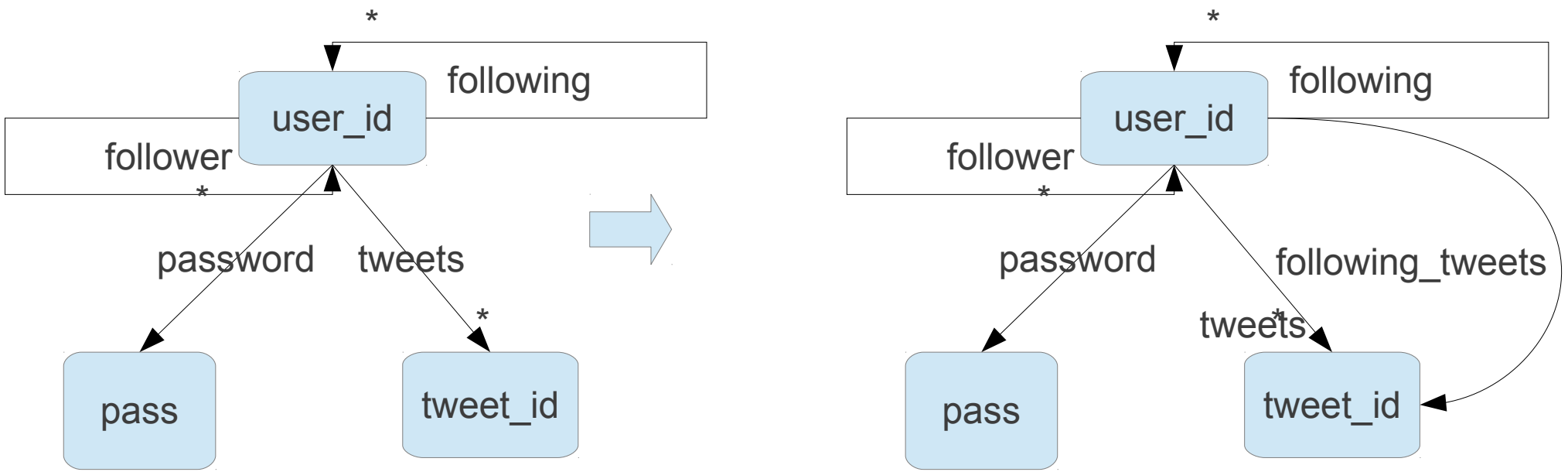
Merge

- Merge two sibling nodes, *a* and *b*, as a new node containing *a* and *b*
- Add a new edge from the parent to the new node. The new edge is named “edgeNameA|edgeNameB”



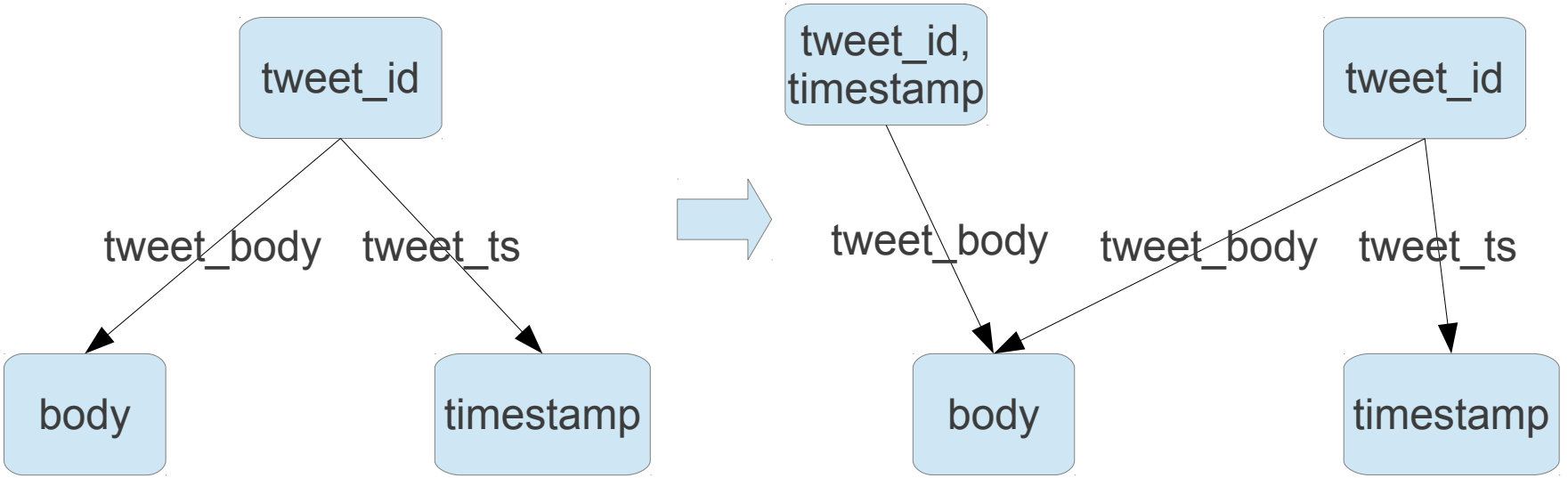
In line

- For any edge(a,b) and edge(b,c) in E (edge set), add a new edge (a,c) into E
- The new edge is named as concatenation of the the name of edge(a,b) and edge(b,c)



Pull up

- For any *edge(a,b)*, create a new node *(a+b)* as the sibling of a
- Duplicate all the edges of a to *(a+b)* except the *edge(a,b)*



Query plan and cost

- Find all paths in the graph, can calculate the path costs
- $cost_q$ is $cost(p_1 U\ p_2 U\ ...)$
- $cost_q(p_1\ U\ p_2) = cost(p_1) + cost(p_2) – cost\ (pre)$
- Where p_1,p_2 are two paths for query q;

  pre is longest common prefix of p_1 and p_2

Vertex cost

- For nodes with more than one attribute, there are configurations, of which attribute or combination of attributes are column names, are enumerated.
- The minimum cost for the query is returned
- If data is not sorted, the sort cost will be added on the the lowest common ancestress node, e.g. tweet_id