**Примеры графов и их решение**

Цель: Необходимо создать функционал по обработке графа.

Исходный граф находится в таблице concept\_relationship (рёбра графа). Граф имеет выделенное направление.

1. concept\_relationship.concept\_id\_1 - Родительский, верхний конец ребра.
2. concept\_relationship.concept\_id\_2 - Дочерний нижний конец ребра.

Для обработки выбираются элементы графа (рёбра, узлы) у которых:

1. Установлен признак годности (relationship.defines\_ancestry=1)
2. Установлен признак доступности (concept\_relationship.invalid\_reason is null и concept.invalid\_reason is null)

Вес ребра графа устанавливается в 1 на основании

1. Родительский узел ребра «Стандартный» (concept.concept\_level >= 1)
2. И тип ребра «иерархическое» (dev.relationship.is\_hierarchical = 1)

Иначе у ребра нет веса (0).

Необходимо заполнить таблицу concept\_ancestor значениями: (до заполнения значениями таблица необходимо очистить от записей)

1. ancestor\_concept\_id –Вершины графа (У вершины графа нет Вышестоящих, родительских узлов.)
2. descendant\_concept\_id – Нижние узлы графа (У нижнего узла графа нет нижестоящего, дочернего узла)
3. min\_levels\_of\_separation – наименьшая сумма весов рёбер по графу от указанной вершины до указанного нижнего узла
4. max\_levels\_of\_separation – наибольшая сумма весов рёбер по графу от указанной вершины до указанного нижнего узла.

Если у записи в concept\_ancestor либо вершина, либо нижний узел стандартный (concept.concept\_level >= 1), то такие записи оставляем в таблице. (остальные записи удаляем)

Для повышения скорости работы алгоритма рекомендуется избегать применения DML опереторов (insert …).

Рекомендуется использовать DDL оператор “create table as select”. База данных работает в режиме NOARCHIVELOG и при таком заполнении таблицы не пишутся журнальные файлы, и заполнение таблицы происходи намного быстрее.

Пример

Содержимое таблицы full\_concept\_ancestor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ancestor\_concept\_id | descendant\_concept\_id | Min\_sep | Max\_sep |
|  | 100 | 101 | 1 | 1 |
|  | 101 | 102 | 1 | 1 |
|  | 100 | 103 | 1 | 1 |
|  | 103 | 101 | 1 | 1 |
|  | 103 | 104 | 0 | 0 |
|  | 101 | 104 | 1 | 1 |

level=1

level=1

level=1

level=1

level=1

level=0

Результат выполнения расчётов в таблице concept\_ancestor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ancestor\_concept\_id | descendant\_concept\_id | Min\_sep | Max\_sep |
|  | 100 | 102 | 2 | 3 |
|  | 100 | 104 | 1 | 3 |

**Пример 1**

Содержимое таблицы full\_concept\_ancestor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ancestor\_concept\_id | descendant\_concept\_id | Min\_sep | Max\_sep |
|  | 100 | 101 | 1 | 1 |
|  | 101 | 102 | 1 | 1 |

min\_sep=1  
max\_sep=1

min\_sep=1  
max\_sep=1

Результат выполнения расчётов в таблице concept\_ancestor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ancestor\_concept\_id | descendant\_concept\_id | Min\_sep | Max\_sep |
|  | 100 | 102 | 2 | 2 |

**Пример2**

Содержимое таблицы full\_concept\_ancestor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ancestor\_concept\_id | descendant\_concept\_id | Min\_sep | Max\_sep |
|  | 100 | 101 | 1 | 1 |
|  | 101 | 102 | 1 | 1 |
|  | 100 | 102 | 1 | 1 |

min\_sep=1  
max\_sep=1

min\_sep=1  
max\_sep=1

min\_sep=1  
max\_sep=1

Результат выполнения расчётов в таблице concept\_ancestor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ancestor\_concept\_id | descendant\_concept\_id | Min\_sep | Max\_sep |
|  | 100 | 102 | 1 | 2 |