Описание Менеджера эвристик

Пусть дан большой(от 12 таблиц) аналитический запрос с внутренними соединениями. Для его планирования будем выполнять следующие шаги:

1. Представим запрос в виде набора связных компонент, любые две таблицы из разных компонент не имеют условия соединения между собой.

2. Если компонента содержит от 100 таблиц, то сначала строим быстрый жадный план с помощью GOO, затем оптимизируем полученные поддервья размером до k, пока не исчерпается бюджет.

3. Если компонента содержит до 100 таблиц, разобьём на топологии: сначала выделим из компоненты **плотные графы** (с 6+ вершинами), затем циклы(*), звёзды с лучами ограниченной длины, останутся цепи. Звездой назовём группу вершин: центральная и связанные с ней цепи, каждые две цепи не связаны с друг другом в оставшейся топологии. Или если имеется вершина с количеством строк в 10+ раз больше чем у 2+ соседей, то данная вершина станет центральной, а её соседи будут концами цепочек.
(*)Если после выделения плотных графов и циклов, компонента остаётся связной, то для планирования этого дерева используем IKKBZ/linDP (т.к позволяет получить очень близкие к оптимальным планы за разумное время), иначе выделяем звёзды и цепи.

4. Назовём топологию маленькой, если в ней ≤ 12 элементов, иначе топология большая. Спланируем каждую топологию графа:

   (a) Для маленькой цепи будем использовать DPsize. Для большой будем linDP.

   (b) Для маленького цикла будем использовать DPsize. В большом цикле найдем соединение с наибольшей кардинальностью, и разобьём цикл по этом соединению - уберём одну таблицу в этом соединении. Спланируем цепь, и присоединим удалённую таблицу.

   (c) Маленькую звезду спланируем DPsize. Для большой - сначала спланируем лучи как цепи, затем будем последовательно присоединять к центру использую GOO (по кардинальности) с центральной фиксированной вершиной.

   (d) Плотный граф от ≤ 6 вершин будем планировать DPsize. Для больших плотных графов GOO/linDP.

5. Алгоритмом GOO(по кардинальности) соберём все планы одной компоненты в один результат.

6. Объединим планы компоненты запроса алгоритмом GOO(по стоимости) с помощью декартового произведения.