Аннотация

Работа посвящена разработке новых эвристических методов планирования запросов в реляционных СУБД. В литературе известно достаточно много алгоритмов планирования, но в индустриальных применениях известны лишь базовые подходы, опирающиеся на динамическое программирование, жадные и генетические алгоритмы и их простейшие эвристические комбинации, позволяющие выбирать тот или иной алгоритм в зависимости от числа таблиц в запросе. Данная работа преследует цель получить глубокое понимание того, как разные алгоритмы возможно комбинировать в ходе планирования запроса, делая переключение между алгоритмами в зависимости от топологии соединений и выделенного временного бюджета на планирование. Ожидаемым результатом работы является решение оптимизационной задачи планирования с помощью новых эвристик, реализация решения в виде модификации планировщика в ядре реляционной СУБД с открытым кодом и экспериментальная оценка на известных индустриальных бенчмарках.

Введение

Современные СУБД работают с большим объёмом информации и транзакций, используя для ввода запросов язык SQL. С ростом количества данных увеличивается и время, необходимое для выполнения запросов. Сокращение этого времени выполнения запросов становится важным фактором для удобства и эффективности СУБД.

В процессе трансляции запрос превращается сначала в логическое представление в виде дерева, затем с помощью оптимизатора СУБД в физический план исполнения. Производительность СУБД напрямую зависит от качества построенного физического плана. Для создания "хорошего" плана нужно решить или приблизить решение NP-трудной задачи выбора оптимального порядка соединений таблиц, так как оно требует сложных вычислений и значительных затрат ресурсов операционной системы. В процессе работы оптимизатора, решается вопрос какой тип соединения использовать, как и в каком порядке соединить таблицы. Запрос преобразуется в набор планов.

Соединение таблиц – это операция, которая позволяет объединять данные из двух и более таблиц по определённому условию. Использование соединений необходимо, когда информация распределена между несколькими таблицами.

Отношение – либо единичная таблица, либо результат соединения нескольких таблиц.

Каждый план можно представить в виде дерева, вершинами в котором являются отношения. Рёбра — условия соединения отношений. При этом операция соединения не ассоциативна по стоимости, то есть $(R1 \bowtie R2) \bowtie R3 \neq R1 \bowtie (R2 \bowtie R3)$.

Выбор, в какой последовательности нужно соединить таблицы, является задачей выбора порядка соединений. Различные планы исполнения для одного и того же запроса возвращают одинаковый результат, но время и ресурсы (CPU, память, I/O обращения, возможно сетевые ресурсы), необходимые для выполнения запроса, сильно различаются. Поэтому выбор оптимального плана позволяет сократить время отклика, минимизировать потребление ресурсов и эффективно обрабатывать большие массивы данных, значительно повышая удобство работы пользователя.

Комбинаторная природа задачи видна через простую оценку. Если соединения выполнять бинарно, форма плана задаётся двоичным деревом с n листьями(таблицами). Таких структур C_{n-1}, где C_k — число Каталана; асимптотически $C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$. Даже без учёта перестановок самих отношений это уже экспоненциальный рост количества возможных планов.

В работе запрос рассматривается как (гипер)граф соединений. Вершины — отношения (таблицы), рёбра — условия соединений. Рёбра, затрагивающие более двух таблиц – гипер-рёбра. Ситуацию осложняют внешние соединения (`OUTER JOIN`): в общем случае они семантически не ассоциативны и не коммутативны, следовательно, ограничивают допустимые переупорядочивания.

Можно заметить, что гиперграф запроса можно разбить на различные топологии: цепи, циклы, звёзды, плотные графы. Количество возможных планов у двух топологий (с обычными рёбрами и без внешних соединений) от одинакового числа таблиц может отличаться в несколько раз, при увеличении количества таблиц разница может вырасти до нескольких порядков.

На практике индустриальные СУБД комбинируют подходы. Для малого числа таблиц применяют динамическое программирование (перебор подмножеств с запоминанием лучших частичных планов). При росте количества таблиц переключаются на эвристики: жадные стратегии, локальные улучшения, генетические алгоритмы и их простые сочетания. У такого подхода есть существенный недостаток: количество возможных планов коррелирует с числом таблиц, но линейная зависимость отсутствует. Также в индустрии есть спрос на тарифицируемое планирование и исполнение запросов: хотелось бы иметь возможность задать ограниченный бюджет на обработку запроса.

В данной работе предлагаются методы адаптивного планирования запросов, которые включают в себя:

1. Различные эвристики для планирования запроса.
2. Критерии для выбора эвристик для планирования запроса.
3. Критерии для динамического переключения между эвристиками во время планирования.
4. Реализацию динамическое переключение между эвристиками в зависимости от критериев.