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

Реляционные СУБД перед исполнением SQL-запроса строят *план* — определяют порядок соединения таблиц (*join order*), выбирают алгоритмы и точки использования индексов. От качества плана зависит время выполнения и расход ресурсов (CPU, память, дисковый и сетевой ввод-вывод). Ключевая подзадача — выбор порядка соединений — комбинаторно взрывоопасна и в общем виде NP-трудна: исчерпывающий перебор становится непрактичным уже при умеренном числе таблиц.

Комбинаторная природа задачи видна через простую оценку. Если соединения выполнять бинарно, форма плана задаётся бинарным деревом с n листьями. 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 пространства планов. Дополнительно реальные запросы ограничивают допустимые перестановки (по графу соединений и по семантике соединений), что ещё больше усложняет задачу.

Удобная модель — *(гипер)граф соединений*. Вершины — отношения (таблицы), рёбра — предикаты соединений. При предикатах, затрагивающих более двух таблиц, естественен гиперграф. Топология здесь существенна: цепочки, «звёзды», плотные циклы или наличие гиперрёбер радикально меняют «стоимость» поиска хорошего порядка. Для ациклических гиперграфов известны полиномиальные многошаговые процедуры исполнения (в духе алгоритма Яннакакиса), а для общих структур — оценки сложности через параметры, наподобие ширины декомпозиции. Тем не менее промышленным оптимизаторам чаще приходится строить бинарные планы и подбирать порядок соединений под стоимостную модель.

Ситуацию осложняют внешние соединения (`OUTER JOIN`): они нарушают полную ассоциативность и коммутативность и жёстко ограничивают допустимые переупорядочивания без изменения смысла. В результате оптимизатор должен одновременно учитывать (i) топологию (гипер)графа, (ii) семантические ограничения порядка, (iii) неточную стоимостную модель и (iv) ограниченный *временной бюджет* на планирование.

На практике индустриальные СУБД комбинируют подходы. Для малого числа таблиц применяют динамическое программирование (перебор подмножеств с запоминанием лучших частичных планов). При росте n переключаются на эвристики и стохастические процедуры: жадные стратегии, локальные улучшения, генетические алгоритмы и их

простые сочетания. Типичное правило — «по числу таблиц»: до порога используется DP, выше — эвристики. У такого правила два недостатка: оно игнорирует форму (гипер)графа и не учитывает бюджет времени на поиск, хотя именно эти факторы критичны.

Идея работы — методы *адаптивного планирования*: выбирать и комбинировать алгоритмы *в зависимости от топологии соединений и выделенного бюджета времени* на оптимизацию. Практически это означает:

1. ввести быструю метрику «топологической сложности» подзадачи (по графу/гиперграфу соединений и ограничениям `OUTER JOIN`);
2. на её основе задать *бюджет перебора* (сколько допустимо генераций и оценок планов);
3. динамически *переключаться* между стратегиями (DP, жадные, стохастические, гибридные), в том числе *на лету* при переходе к подзадачам иной структуры;
4. гарантировать «безопасное» завершение: при исчерпании бюджета возвращать план предсказуемого качества вместо незавершённого поиска.

Научная значимость состоит в том, что вместо статического «разделения труда по n» предлагается топология- и бюджет-осознанный контроль над поиском: алгоритм выбора порядка соединений становится процедурой управления вычислительным бюджетом на графе, а не просто выбором одной фиксированной техники. Это позволяет систематизировать уже известные эвристики через единый интерфейс бюджета и сравнивать их «на равных» на классах топологий (цепи, звёзды, циклы, смешанные гиперрёбра).

Практическая часть — реализация прототипа в открытом планировщике (ядро СУБД с открытым кодом) с добавлением: (а) оценки топологической сложности (например, через плотность рёбер/гиперрёбер, наличие «жёстких» компонент из-за `OUTER JOIN`, ориентировочную ширину декомпозиции); (б) бюджетируемых операторов поиска (ограничение на число вызовов генерации/оценки частичных планов); (в) переключателя стратегий на границах подзадач (цепные фрагменты, почти-звёздные центры, малые циклы и т. п.).

Оценка — на бенчмарках TPC-H и TPC-DS: сравнение времени планирования, качества планов (время выполнения, пиковая память, объём I/O) и устойчивости результатов по классам топологий. Ожидаемый эффект — более стабильное качество планов при контролируемом времени оптимизации, особенно на «неудобных» структурах графа и при наличии внешних соединений.

Таким образом, работа соединяет математически понятную комбинаторику (числа Каталана, ацикличность/циклы, гиперрёбра и ограничения на перестановки) с инженерной задачей управления вычислительным бюджетом в оптимизаторе запросов. Результат — набор адаптивных эвристик и правил переключения, воспроизводимая реализация и экспериментальные данные, пригодные для дальнейшего развития в промышленных СУБД.