Лекция 6

Методы оптимизации запросов

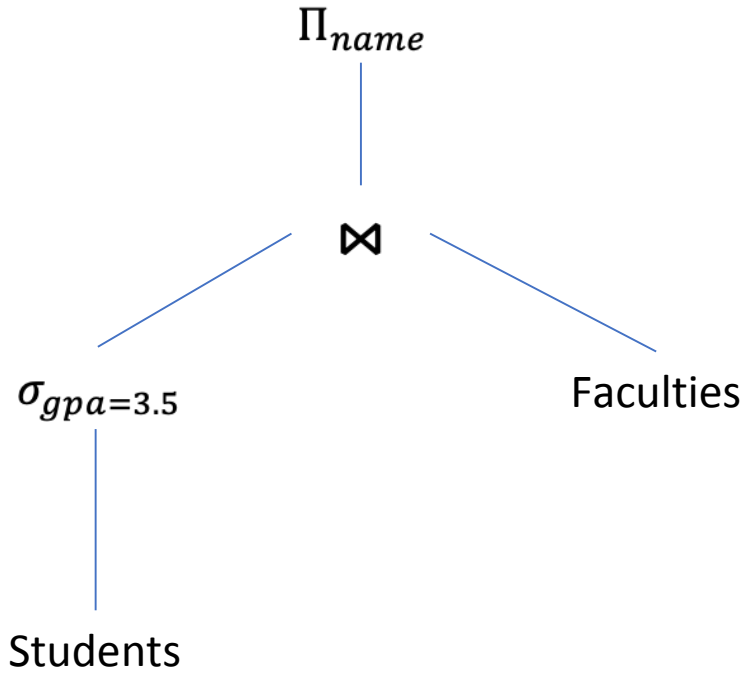
Основные шаги обработки запросов

Вычисление выражений

- Для вычисления выражений, содержащих несколько реляционных операций, существует несколько вариантов вычисления.
- Если каждая операция вычисляется отдельно и результатом такого шага является временное отношение, то такая ситуация называется *материализацией.*
- Альтернативным способом является организация *конвейера (pipeline)* выполнения параллельных операций

Логический план запроса

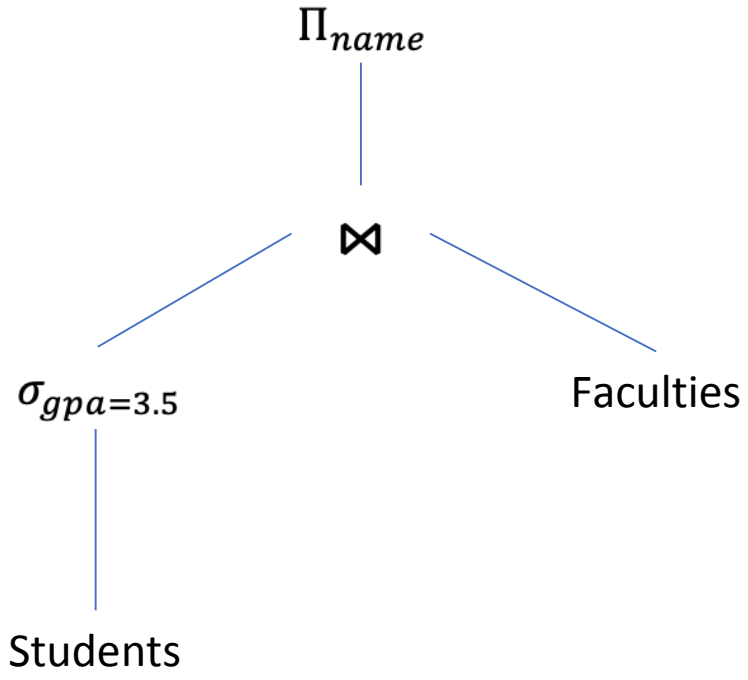
Операцию реляционной алгебры можно представить в виде дерева операций: $\Pi_{name}(\sigma_{gpa=3.5}(Students) \bowtie Faculties)$




```
SELECT f.name FROM Students s
JOIN Faculties f USING(f_id)
WHERE s.gpa = 3.5;
```

Логический план запроса

Операцию реляционной алгебры можно представить в виде дерева операций: $\Pi_{name}(\sigma_{gpa=3.5}(Students) \bowtie Faculties)$



На листах дерева находятся отношения, в других вершинах – операции.

В данном примере входными элементами для соединения являются отношение Faculties из БД и временное отношение, полученное из выборки из Students.

Физический план запроса

Создается на основе логического плана посредством выбора конкретных алгоритмов для каждой операции.

Материализация

Материализованное вычисление представляет собой последовательность вычислений, в которой создается каждая промежуточная операция (материализуется) и используется для вычислений дальнейших шагов.

Для вычисления стоимости требуется добавить стоимость всех операций и стоимость записи временных результатов на диск.

Одним из вариантов оптимизаций является двойная буферизация (один из буферов осуществляет вычисление, другой – запись)

Конвейер

- $\Pi_{a,b}(r \bowtie s)$
- Уменьшение стоимости чтения и записи временных отношений
- Довольно быстрый отклик в передаче результатов запроса, если план запроса позволяет полноценный конвейер.

Реализация конвейера

Конвейер может быть 2 типов:

- Demand-driven (Ленивое вычисление). Результат запроса операций низкого уровня не передается автоматически на более высокий уровень. Он будет передан, когда будет осуществлен запрос более высокого уровня.
- Producer-driven (Жадное вычисление). Операции не ждут запрос для вычисления кортежей. Каждая операция в producer-driven конвейере моделируется как отдельный процесс или поток в системе, которая на вход принимает поток кортежей и на выходе также поток кортежей.

Конвейеры с ленивым вычислением:

- Каждая операция может быть разработана как итератор, который предоставляет следующие функции: open(), next(), close()
- После вызова open(), каждое обращение к next() возвращает следующий кортеж на выход операции.
- Функция close() сообщает итератору, что больше кортежей не требуется
- Итератор хранит состояние вычисления между вызовами.

Конвейеры с жадным вычислением:

- Для каждой смежной пары операций в конвейере, система создает буфер для фиксации кортежей, передающиеся с одной операции на другую.
- Как только операция использует кортеж из входного буфера, он удаляется из буфера.
- Процессы (или потоки) разных операций работают параллельно.

Вычисление алгоритмов для конвейеров

- Планы запросов могут указывать, для каких вершин выполняется конвейеризация (конвейерные вершины). Другие вершины называются блокирующими вершинами (материализованными).
- Множество всех соединенных конвейерных вершин вычисляется параллельно.
- План запроса может быть разделен на поддеревья, вершины между которым являются блокирующими.
- Каждое такое поддерево называется стадией конвейера.

Вычисление алгоритмов для конвейеров

- Операции, такие как сортировка или удаление дубликатов, по существу являются блокирующими, так как они не могут выдавать результат до того, как все кортежи были забраны из входных отношений.
- Однако, иногда блокирующие операции могут выдавать выходные результаты по мере их появления.
- Пример. Сортировка внешним слиянием. 1) Формирование шагов 2) Слияние. Шаг 1 может быть применен по мере появления кортежей во входном буфере, и поэтому может быть применена конвейеризация. Шаг 2 может посылать выходные результаты по мере их появления, но при этом для начала требуется полное завершение шага 1.

Вычисление алгоритмов конвейеризации

- Другие операции (например, соединение) могут быть не полностью не блокирующими, но при этом специфическое вычисление алгоритмов может приводить к блокировке.
- Например, алгоритм вложенных циклов может выдавать результирующие кортежи, как только кортеж получен для внешнего отношения. Поэтому данная часть является конвейерной.
- Но алгоритм хеш соединения является блокирующей операцией для обоих входов, так как требуется полное получение всех кортежей и секционирование отношения.

Вычисление алгоритмов конвейеризации

- В части приложений может быть алгоритм соединения, который является конвейером. Например, если оба входа отсортированы по атрибутам и каждое условие соединения является экви-соединением, может быть применен алгоритм merge-join.
- Существует альтернатива двойного-конвейерного соединения. Алгоритм предполагает, что входные кортежи являются конвейерными.

Оптимизация запросов

Методы оптимизации запросов

Два подхода (применяются в совокупности):

- Сравнительная оценка стоимости
- Эвристические правила

Оптимизация запросов

- Оптимизация запросов – процесс выбора наиболее эффективного плана из одной или многих стратегий, которые могут быть применены для обработки запросов
- Один из пунктов оптимизации возникает на уровне реляционной алгебры, когда система пытается найти выражение, которое эквивалентно указанному выражению, но наиболее эффективно для вычисления
- Другой пункт оптимизации – это выбор детализированной стратегии для обработки запроса, такой как выбор алгоритма для выполнения операции

Оптимизация запросов

Пусть Q — множество всех возможных запросов.

Запросы q1, q2 из Q называются **эквивалентными**, если в результате их работы получается одно и то же отношение для любых наборов экземпляров отношений (с точностью до порядка кортежей и атрибутов, если он явно не задан в запросе).

Общий план поиска

1) Получение всевозможных эквивалентных планов запроса
2) По каждому логическому запросу сгенерировать всевозможные физические планы запросов
3) Для каждого плана посчитать стоимость, выбрать наименьший

Правила эквивалентности

На основе:

- алгебраических свойств операций
  - ассоциативность (соединение, объединение, пересечение)
  - дистрибутивность (фильтрация, проекция)
- свойств логических операций
  - вычисление КНФ или ДНФ условного выражения
- свойств реляционных операций

Общий план поиска

1) Выбирается план запроса и делаете цикл:

- для каждого нового элемента
  - для каждого правила
    - пытаемся создать выражение на основе правила эквивалентности
    - если выражение новое, добавляем в список

2) По каждому логическому запросу сгенерировать всевозможные физические планы запросов
3) Для каждого плана посчитать стоимость, выбрать наименьший

Возможные оптимизации

- Использование указателей на подмножества планов
- Уменьшение количества генерируемых планов запроса (на основе эвристик и стат. данных)
  - используется информация из системного каталога (INFORMATION_SCHEMA)
- Кэширование планов

INFORMATION_SCHEMA

- Количество кортежей в отношении
- Количество блоков, содержащих кортежи отношения
- Данные об устройстве кортежа
- Распределение количества кортежей в блоке
- Гистограмма уникальных значений в таблице по полям

Законы преобразования

- последовательные выборки можно заменить одной
- последовательные проекции можно заменить одной
- выборка из проекций меняется на проекцию выборки
- выборка соединений меняется на соединение выборок
- внешние соединения меняются на обычные + доп. объединение
- использование логических свойств
  - для построения КНФ логических формул

Основная цель - уменьшение кол-ва обрабатываемых данных.

Эвристики

Эвристический алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев.

Используются знания, относящиеся к конкретной задаче.

Примеры:

- выполнять выборку как можно раньше

- выполнять проекцию как можно раньше

Подмножества планов

- Использование указателей на подмножества планов
  - пример применения - жадный алгоритм
  - или для рекурсивного поиска

- Кеширование планов
  - когда возникает аналогичный запрос, можно пропустить часть плана, если заранее хранить его результат