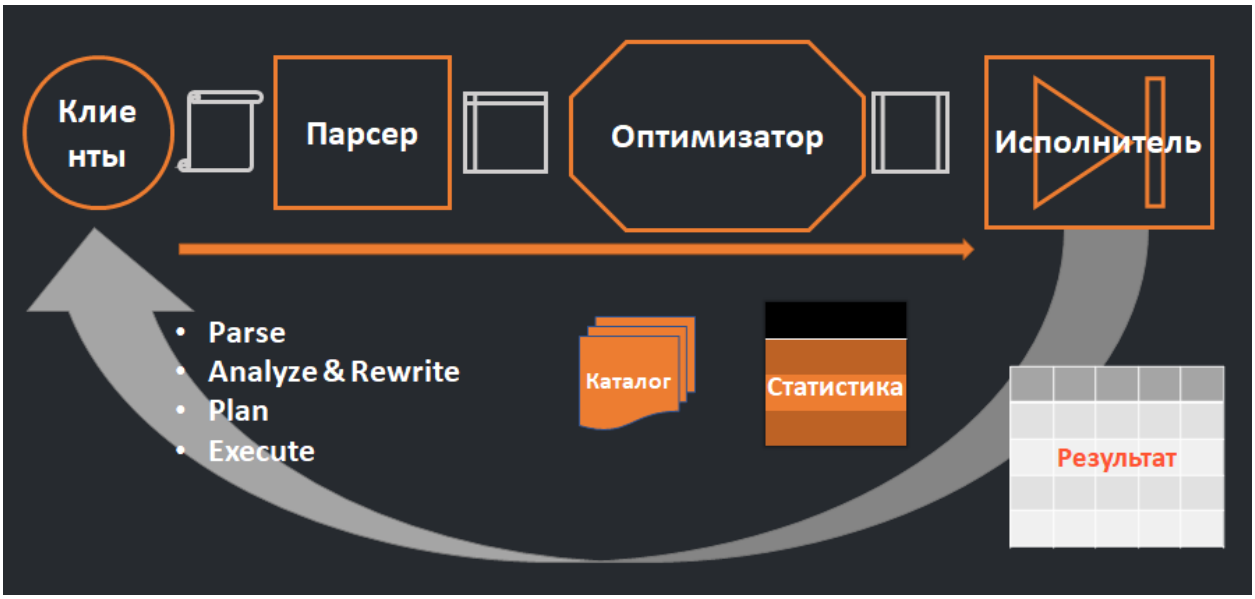
KARPOV.COURSES >>>
КОНСПЕКТ

> Конспект > 3 урок > Обработка запросов в обычной СУБД и в MPP СУБД

> Оглавление

> Жизненный цикл запроса



Общая схема жизненного цикла запроса

Клиенты: приложение, аналитики, пользователи, которые пишут запрос к БД. Запрос из клиентского приложения попадает в СУБД.

Парсер: очищает запрос, проверяет синтаксическую правильность, выдаёт ошибки (если есть), преобразует запрос в вид понятный программе оптимизатора.

Оптимизатор: выполняет наиболее важные функции в обработке запроса

- Analyze – анализирует запрос,
- Optimize – оптимизирует, чтобы запрос выполнился быстрее,
- Rewrite – запрос может быть переписан в соответствии с правилами СУБД,
- Plan – строит план выполнения запроса, опираясь на информацию из **каталога** обо всех объектах, которые участвуют в запросе (их свойствах, где и как они хранятся), и используется **статистика** (какие максимальные, средние значения содержатся в ячейках объектов запроса, насколько высокая селективность у каждой из ячеек, участвующих в запросе). План описывает, как должен выполниться запрос исполнителем, начиная с чтения данных в листьях, обработки в промежуточных узлах плана и предоставлению реузультата в корневом узле плана.

Исполнитель: получает план и выполняет его по шагам (считывает индексы, данные, производит сортировки, обработки, арифметические операции), собирает

результат, который направляется клиенту, как результат выполнения запроса.

> Особенности обработки запроса в GreenPlum

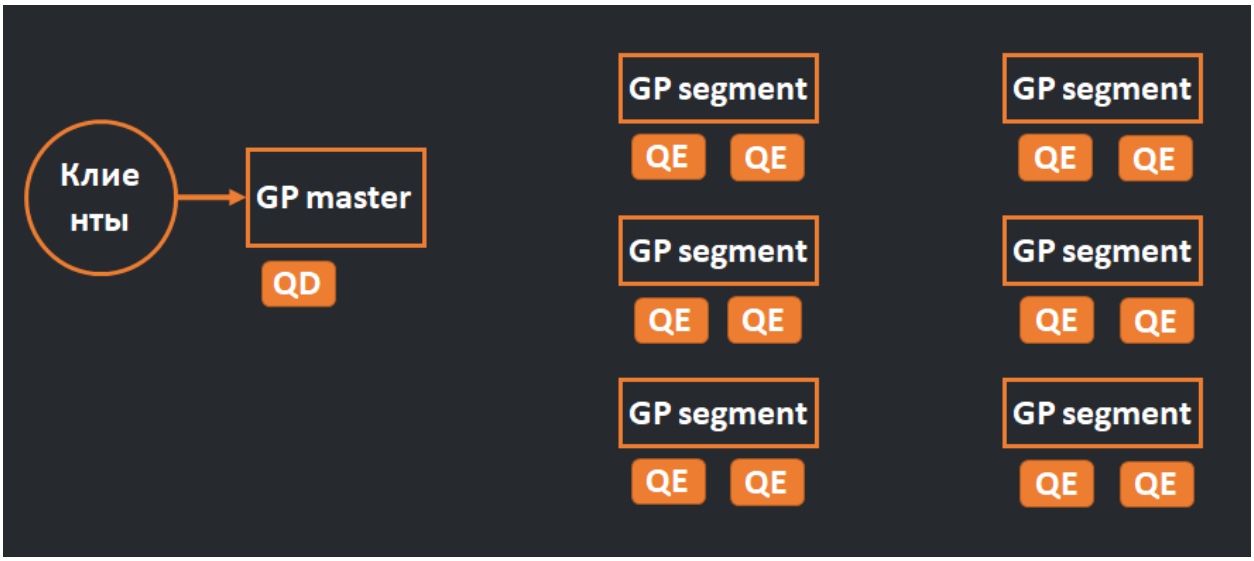


Схема обработки запросов в GreenPlum

Клиенты: формируют запрос и направляют его на GP master.

GP master: единая точка входа:

- принимает все запросы пользователей,
- выполняется парсер и оптимизатор, который подготавливает план запроса для выполнения на сегментах,
- поднимает процесс Query Dispatcher (QD), который открывает соединение к каждому из сегментов GP и передаёт план.

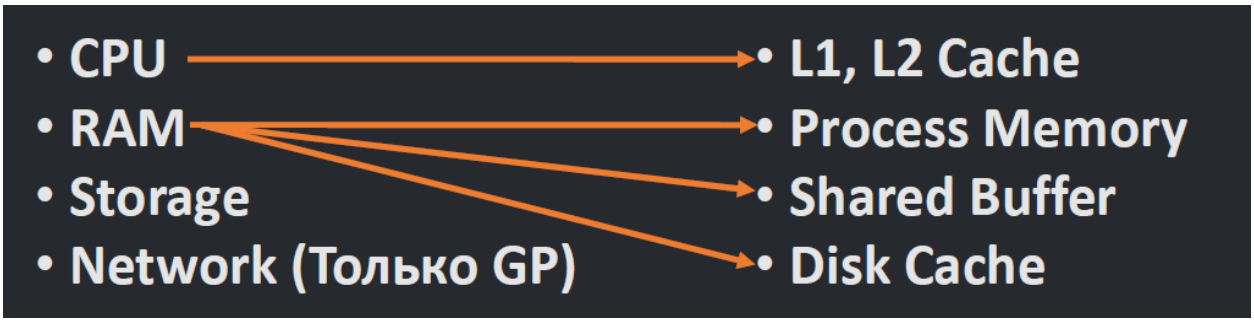
GP segment: сегменты GreenPlum, на которых лежат данные:

- на каждом сегменте поднимается **Query Executor (QE)**, если процесс можно распараллелить, то для ускорения выполнения запроса поднимаются несколько QE,
- QE являются исполнителями, которые выполняют все операции запроса и собирают промежуточный результат,

- может потребоваться уникальная для MPP СУБД операция **Redistribute** – сегментам необходимо обменяться данными (необходимое по плану количество раз),
- на каждом сегменте будет готов свой финальный набор данных **Result Set (RS)**, каждый QE передаёт свой RS на master в QD.

Далее QD собирает данные из всех сегментов, производит окончательные операции (limit, фильтры) и отдаёт результат запроса клиенту.

> Ресурсы доступные планировщику на сервере СУБД



CPU – вычислительные ядра процессора (L1, L2 Cache)

RAM – оперативная память:

- Process Memory (занимают процессы),
- Shared Buffer (используется для буферов),
- Disk cache (находится недавно использованная информация).

Storage – хранилище, где лежат данные (физические носители)

Network – сеть для GreenPlum, т.к. он содержит несколько сегментов.

> Как оптимизатор принимает решения

Cost Based Optimizer – оптимизаторы принимают решения о том какой выбрать план исходя из стоимости плана.

Стоимость складывается из:

- данных в **словаре данных**: размер объекта, тип хранения (поколоночный, сжатый для GreenPlum), есть ли индексы,
- данных **статистики**: селективность полей (насколько одинаковые или разные значения содержатся в одном поле), гистограмма значений объекта (чтобы оптимизатор понимал насколько часто встречаются те или иные объекты), оценка размеров Result Set.

Создаётся множество вариантов плана и производится расчёт стоимости в условных единицах (для PostgreSQL одна операция ввода-вывода, т.е. одну операцию чтения страницы с данными с диска).

Для каждого узла плана есть базовая стоимость, есть коэффициенты, которые влияют на стоимость.

Выбирается самый дешёвый план и передаётся исполнителю.

> Пример плана объединения двух таблиц

	QUERY PLAN
1	Limit (cost=0.00..970.99 rows=2 width=224)
2	-> Gather Motion 64:1 (slice1; segments: 64) (cost=0.00..970.97 rows=100 width=224)
3	Merge Key: lineitem.l_quantity
4	-> Limit (cost=0.00..970.92 rows=2 width=224)
5	-> Sort (cost=0.00..970.92 rows=5981 width=224)
6	Sort Key: lineitem.l_quantity
7	-> Hash Join (cost=0.00..875.62 rows=5981 width=224)
8	Hash Cond: (lineitem.l_orderkey = orders.o_orderkey)
9	-> Seq Scan on lineitem (cost=0.00..433.70 rows=7513 width=117)
10	Filter: (l_quantity > 10::numeric)
11	-> Hash (cost=431.56..431.56 rows=1515 width=107)
12	-> Seq Scan on orders (cost=0.00..431.56 rows=1515 width=107)
13	Filter: (o_totalprice > 100000::numeric)

План запроса на объединение 2х таблиц, сортировку значений, взятие первых 2 строк

План представляет из себя дерево

- корневой уровень – результат возвращается клиенту,
- ветви – выполняют операции над данными,
- листья – нижний уровень, начало работы с данными, осуществляется физическое чтение данных.

Что присутствует в плане

- cost – стоимость операции,
- rows – количество строк,
- width – примерная ширина возвращаемой строки,
- Seq Scan – непосредственный доступ к данным в листах,
- Hash – операции расчёта хэша,
- Hash Join – операции объединения,
- Sort, Limit и др. – операции запроса, которые влияют на стоимость, количество строк, столбцов,
- Gather Motion – специфичная для GreenPlum операция, означает что все данные полученные с сегментов собираются на мастере и мастер отдаёт клиенту результат.

План показывает самый дорогой оператор, где наибольшее количество строк, где торможение, где можно что-то подвинуть, поменять оператор, либо ускорить, что-либо сделать с данными.

> Как выполняется план

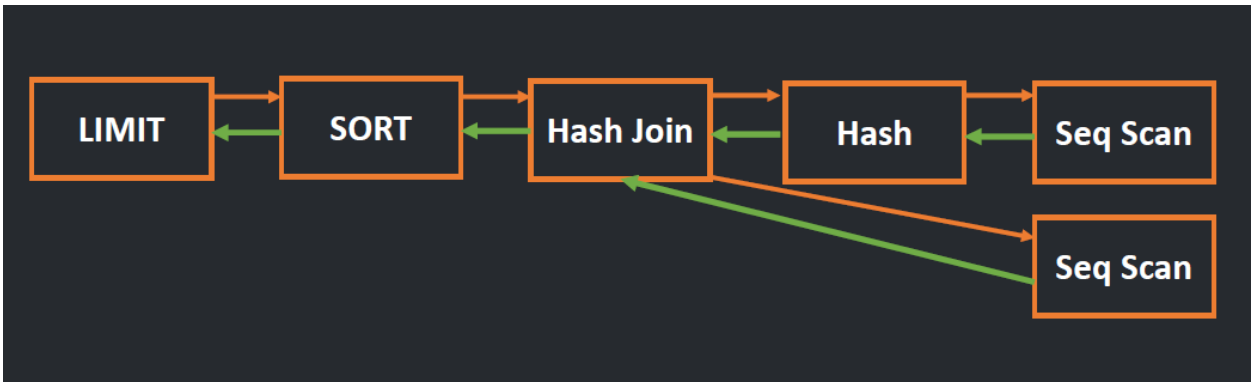


Схема выполнения плана

Последовательность выполнения плана:

1. Исполнитель просматривает план от корневого узла и добирается в листовые узлы,
2. В листовых узлах начинается выполнение плана, осуществляется доступ к данным, происходит последовательное сканирование таблицы или сканирование индекса, получает исходный набор данных,
3. Набор данных отправляется в ближайший узел плана запроса в сторону корня,
4. На каждом узле происходит соответствующая плану обработка данных, результаты отправляются на следующий узел в сторону корня,
5. Исполнитель агрегирует результаты работы по плану во время обработки на корневом узле,
6. Результат предоставляется клиенту.

> Какими бывают узлы плана

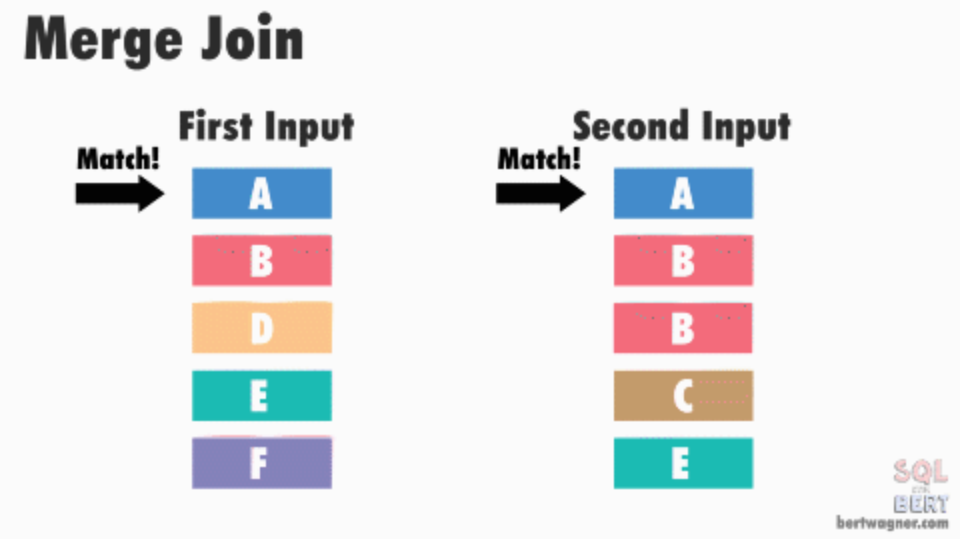
Листовые узлы плана

- Sequential Scan – последовательное чтение файла данных с диска,
- Index Scan – чтение индекса, чтобы получить ссылки на страницы с данными таблицы и далее чтение таблицы,
- Index Only Scan – в индексе содержатся не только поля, но и требуемые данные, нет необходимости читать таблицу,

- Bitmap Scan – построение битовой карты при чтении таблицы и выбор нужных результатов подходящих по битовой карте.

Узлы объединения

- Nested Loop Join – объединяются два Result Set (один внешний, другой внутренний), сравнивается каждый элемент внутреннего с каждым элементов внешнего, все элементы, для которых выполняется условие соединения, попадают в итоговый Result Set,
- Hash Join – для объединения двух объектов строится hash по ключу объединения на одном из объектов и в процессе объединения по полю объединения вычисляется hash второго объекта и происходит сопоставление элементов объектов по hash ключам объединения. Можно создать только отношение равенства,
- Merge Join – сравнивает первые строки с обоих отсортированных входных данных. Затем он продолжает сравнивать следующие строки со второго входа, если значения соответствуют значению первого входа.



Узлы плана

- Sort – выполняет сортировку Result Set,
- Aggregate/Group by Aggregate – выполняет агрегирование данных и выполняет расчёт агрегирующей функции,

- Limit – возвращает только часть Result Set.

Дополнительные операторы плана GreenPlum

- Redistribute Motion – перераспределение данных между сегментами, чтобы на каждом из сегментов был доступен весь объект,
- Gather Motion – сбор результатов расчёта с каждого из сегментов на master и последующая выдача клиенту,
- External Scan – узел нижнего уровня, применяется к внешним таблицам (с внешних источников).

> Как повлиять на план

Для начала можно расследовать, почему план работает неэффективно, и понять, что можно было бы улучшить:

- Изучить план запроса, выполнить команду explain – посмотреть теоретический план запроса, понять что ему не хватает, что не сходится, где слишком большие цифры,
- Выполнить команду explain analyze – фактически выполняет запрос и позволяет понять, что происходило в СУБД в реальности с реальным временем выполнения и реальным количеством полученных строк в результате выполнения каждого из узлов,
- Подумать, как эффективней сохранить объект (для GreenPlum есть возможность колоночного хранения, есть возможность сжатия данных, поколоночного сжатия и т.д.).

Влияние на ускорение выполнения плана:

- Hints (не используются в PostrgeSQL или GreenPlum, но широко используются в Oracle, Microsoft SQL) – позволяют в ручном режиме подсказывать базе, как лучше выполнить запрос,
- Параметры оптимизатора – ими можно управлять, например, не использовать Hash Join, Nested Loop Join, сказать оптимизатору, что Sequential Scan будет стоить дорого и лучше воспользоваться Index Scan,

- Переписать запрос – например, слишком много join и может возникнуть проблема с выбором оптимального варианта.