

ОПТИМИЗАЦИЯ ЗАПРОСОВ В ORACLE

Бухвалова И.А., Пильник Н.Ю.
БНТУ, Минск, Беларусь, irena_buchval@mail.ru

Когда СУБД выполняет запрос сначала требуется определить наилучший план выполнения. Для этого нужно рассчитать, как и в каком порядке обращаться к данным и соединять их, как и когда выполнять вычисления и агрегации т.д. За это отвечает подсистема, которая называется Оптимизатор запросов. В СУБД Oracle есть два основных вида оптимизаторов [1]:

1. Оптимизатор, основанный на анализе заданных правил (rule-based optimizer).
 - Этот оптимизатор выбирает методы доступа на основе предположения о статичности СУБД
 - Такой оптимизатор учитывает иерархическое старшинство операций.
 - Если для какой-либо операции существует более одного пути ее выполнения, то выбирается тот путь, чей ранг выше, т.к. в большинстве случаев он выполняется быстрее, чем путь с более низким рангом.
 - План выполнения запроса формируется из выбранных путей доступа с максимальными рангами.
2. Оптимизатор, основанный на анализе затрат (cost-based optimizer).
 - При использовании этого метода оптимизатор сначала строит несколько возможных планов выполнения запроса.
 - При этом он применяет некоторые эвристики, т.е. правила, полученные опытным путем.
 - Эти правила позволяют сузить пространство поиска оптимального плана благодаря тому, что неэффективные планы отбрасываются в самом начале и не рассматриваются.
 - Для каждого из построенных планов рассчитывается его стоимость.

Оптимизация выполнения запроса осуществляется в следующем порядке:

1. Вычисление выражений и условий, содержащих константы.
2. Преобразование сложной команды в эквивалентную ей с использованием соединения (проводится не всегда).
3. Если команда выполняется над представлением, то оптимизатор обычно объединяет запрос на создание представления и запрос к этому представлению в одну команду.
4. Выбор метода оптимизации.
5. Выбор путей доступа к таблицам, к которым обращается запрос.
6. Выбор порядка соединения (если в запросе соединяются несколько таблиц, то оптимизатор определяет, какие две таблицы будут соединяться первыми, какая таблица следующей будет подключаться в результате и т.д.).
7. Выбор операции соединения для каждой команды соединения.

Для указания режима оптимизации в файле параметров `init.ora` следует использовать приведенные ниже значения параметра `OPTIMIZER_MODE`.

CHOOSE. При установке этого значения будет выбрана оптимизация, основанная на анализе затрат, при наличии у оптимизатора соответствующих статистических данных. В противном случае будет использована оптимизация, основанная на анализе правил.

RULE. При установке этого значения будет использована оптимизация, основанная на анализе правил.

Оптимизация запроса предполагает удаление причин неэффективности запроса, среди которых наиболее весомыми являются:

- плохая статистика таблиц и индексов, участвующих в запросе (наиболее важный фактор, на который в первую очередь надо обратить внимание);

Oracle ежедневно в определенные часы в рабочие дни и в определенные часы в выходные сам собирает статистику по таблицам. Но для этого DML операции с таблицей должны привести к изменению не менее 10% строк таблицы. Иначе нужно обновлять статистику, используя процедуры сбора статистики внутри пакетов, а в ряде случаев использовать JOB, запускающийся в определенные часы для анализа и обновления статистики. Статистика по таблице и индексу (на примере таблицы AGREEMENT и индекса X_AGREEMENT в схеме HIST) обновляется соответственно процедурами:

для таблицы:

```
execute DBMS_STATS.GATHER_TABLE_STATS
        ('HIST', 'AGREEMENT', NULL, 10, NULL, 'FOR ALL INDEXED
        COLUMNS SIZE AUTO', 4);
```

для индекса:

```
execute DBMS_STATS.GATHER_INDEX_STATS('HIST',
        'X_AGREEMENT', null, 10, null, null, 4);
```

Одним из приемов, оправдавшим себя на практике, является блокировка сбора статистики. Используется при интенсивном изменении числа строк в таблице в течение дня (многочисленные удаления и вставки строк). Результаты ручного или ночного сбора статистики, осуществляемый Oracle, закрепляется путем блокировки дальнейшего сбора статистики.

- проблемы с индексами: отсутствие нужных индексов, неэффективно построенные индексы, неэффективно используемые индексы, большое значение фактора кластеризации;
- проблемы с хинтами: отсутствие хинтов или они неэффективны;
- неэффективная структура запроса (запрос построен не корректно).

Причин неэффективности запроса несколько:

- неэффективное соединение таблиц;
- использование NOT и NOT IN в условии where;
- блокировка индекса в силу использования неправильных функций к столбцу, по которому построен индекс;
- большая вложенность запроса или большая его длина;
- большой объем выбираемых данных, требующих подключения в работу дисков, в том числе для выполнения агрегированных функций (order by, group by и т.д.);
 - неэффективные хранимые процедуры, используемые в запросе и др.

При построении запросов, если необходим доступ к значительной части строк какой-либо таблицы полное сканирование является более эффективным, чем индексное. Для сканирования индекса и извлечения строки требуются, по крайней мере, две операции чтения для каждой строки, а в некоторых случаях и больше — в зависимости от количества уникальных данных в индексе. При полном сканировании таблицы для извлечения строки требуется только одна операция чтения. При доступе к большому количеству строк становится очевидной неэффективность использования индекса по сравнению с полным сканированием таблицы, при котором строки считываются непосредственно из таблицы.

Oracle содержит полезный инструмент *SQL Tuning Advisor*, который позволяет осуществлять автоматическую настройку SQL-запросов, создавать так называемые профили, позволяющие при том же исходном наборе объектов (например, без создания новых индексов) построить более эффективный план запроса.[2]

Литература.

1. R. Ahmed, «Cost-Based Query Transformation in Oracle», Proceedings of the 32th VLDB Conference, Seoul, S. Korea, 2006.
2. Кузнецов С., Оптимизация запросов в системах баз данных, http://www.citforum.ru/database/articles/query_optimization/