Обработка и исполнение запросов в СУБД

(Лекция 11)

Физический дизайн в СУБД

v3

Георгий Чернышев

Высшая Школа Экономики

chernishev@gmail.com

9 декабря 2020 г.

План

- Задача настройки СУБД.
  - Автоматическая настройка СУБД;
  - Физические структуры и что можно получить;
  - Формальная постановка, решение.
- “Классификация” методов решения на примере задачи верт. фрагментирования:
  - процедурные;
  - стоимостные;
  - смешанные;
- Горизонтальное фрагментирование:
  - типы с точки зрения пользователя;
  - фрагментирование на основе предикатов;
  - распространение по ключам;
- Размещение данных
  - Цели
  - Типы: классификация по объекту размещения и избыточности;
  - Размещение и фрагментирование;
  - Модели в задаче размещения;
  - Динамическая постановка;
- Современные работы.

Задача настройки СУБД (1)

Задача настройки СУБД под известную схему данных и известный набор запросов — одна из самых старых задач области баз данных.

Успешное решение позволяет существенно повысить производительность СУБД или добиться нужных значений других необходимых характеристик.

Задача настройки СУБД (2)

Администратор баз данных может:

- переписать запросы, расставить подсказки или иным образом повлиять на план выполнения конкретного запроса;
- изменить параметры СУБД (используемый размер оперативной памяти, максимальное количество соединений к базе, политики журналирования и прочее);
  - Пример: параметр *work_mem* в PostgreSQL задает максимально допустимый объем оперативной памяти, используемый в том числе для сортировки. Если данные не влезут — будет использоваться внешняя сортировка, со свопом на диск.
- выбрать наиболее подходящие структуры физического уровня.

Это делали (и делают) вручную, однако с момента появления области баз данных постоянно предпринимались попытки автоматизации.

Задача автоматической настройки СУБД (1)

Самые ранние работы по автоматизации выбора физических структур, что мне удалось найти:

- в базах данных (задача вертикального фрагментирования): $\sim$ 1975 год [Hoffer, 1975];
- задача о размещении файла в компьютерных сетях: $\sim$ 1969 год [Chu, 1969];
- задача о снабжении военных складов: $\sim$ 1940 годы;
- ... дальше в прошлое не заглядывал, поэтому могут найтись еще предшественники!

Задача автоматической настройки СУБД (2)

Несмотря на зрелость, этот род задач до сих пор актуален:

- Востребованность решения — индустрия заинтересована в качественных решениях, они дают экономический эффект;
- *NP*-трудность подавляющего большинства содержательных постановок приводит к необходимости использовать алгоритмы дающие приближенное решение.

Кроме того:

- Новые модели данных;
- Новое оборудование;
- Новые сценарии.

Настройка СУБД: методы

- Фрагментирование данных (data partitioning или data fragmentation): горизонтальное и вертикальное фрагментирование и их комбинации (hybrid или mixed partitioning).
- Размещение данных по узлам распределенной СУБД (data allocation).
- Использование дополнительных структур — индексов и материализованных представлений.
- Репликация данных.

Что можно получить в случае успешного решения? (1)

Увеличить мощность системы

Организовать эффективное добавление новых узлов и устройств хранения в распределенную систему.

Внутризапросный параллелизм

Успешное решение создаст необходимые условия для более широкого применения внутризапросного параллелизма, что позволит уменьшить время выполнения отдельного запроса.

Что можно получить в случае успешного решения? (2)

Межзапросный параллелизм

Ускорит выполнение набора запросов с помощью межзапросного параллелизма.

Понизить совокупную стоимость владения системы

Упрощение системы в целом, упрощение или даже устранение задач администрирования, конфигурирования и настройки приведет к снижению затрат на персонал.

Затраты на оборудование могут быть так же снижены: например, применение интервального горизонтального фрагментирования может уменьшить гранулярность операций резервирования и восстановления [Lightstone, 2009]

Формальная постановка задачи настройки СУБД (1)

Поведение СУБД опишем как функцию [Chaudhuri, 2009]:

$$f : C \times W \to P,$$

где C обозначает конфигурацию системы, W — нагрузку, а P — производительность.

Конфигурация — это описательная характеристика системы, включающая в себя:

1. аппаратную конфигурацию (количество и характеристики процессоров, характеристики оперативной памяти, характеристики дисковых устройств, характеристики используемой сети и т.д.);

Формальная постановка задачи настройки СУБД (2)

Поведение СУБД опишем как функцию [Chaudhuri, 2009]:

$$f : C \times W \to P,$$

где C обозначает конфигурацию системы, W — нагрузку, а P — производительность.

2. программную конфигурацию — настраиваемые параметры СУБД, вступающие в силу после загрузки;
3. конфигурацию физического уровня СУБД (использование индексов, материализованных представлений и др.);
4. процедуры резервного копирования и репликации;
5. стратегии адаптации к изменяющимся условиям;
6. стратегии обработки исключительных ситуаций.

Формальная постановка задачи настройки СУБД (3)

Поведение СУБД опишем как функцию [Chaudhuri, 2009]:

$$f : C \times W \to P,$$

где C обозначает конфигурацию системы, W — нагрузку, а P — производительность.

Нагрузка — это предполагаемый набор запросов и их характеристики: затрагиваемые атрибуты, типы запросов, интенсивность поступления в заданные периоды (рабочие или пиковые часы), распределение параметров этих запросов и многое другое.

Нестандартный пример: настройка БД в случае набора повторяющихся сценариев работ [Agrawal, 2006].

Формальная постановка задачи настройки СУБД (4)

Поведение СУБД опишем как функцию [Chaudhuri, 2009]:

$$f : C \times W \to P,$$

где C обозначает конфигурацию системы, W — нагрузку, а P — производительность.

Производительность P — характеристика работы системы:

1. пропускная способность системы, измеряемая в количестве обработанных запросов в единицу времени (throughput);
2. время ответа на запрос (response time);
3. различные метрики измерения функциональной надежности (dependability): доступность (availability), надежность (reliability) и производительность в условиях ослабленных конфигураций (performability).

Формальная постановка задачи настройки СУБД (5)

Поведение СУБД опишем как функцию [Chaudhuri, 2009]:

$$f : C \times W \to P,$$

где C обозначает конфигурацию системы, W — нагрузку, а P — производительность.

Задача: найти конфигурацию физического уровня, обладающую заданной производительностью. Однако чаще всего требуемый уровень производительности заранее не известен, и необходимо найти максимально достижимый.

Итог: экстремальная задача, где в качестве максимизируемой функции выступает производительность P в зависимости от C и W, ограничения описываются набором уравнений и неравенств, а искомая конфигурация — набором параметров.

Уточнение рассматриваемой задачи

1. Конфигурация определяется как конфигурация физического уровня СУБД. Все остальные компоненты конфигурации (аппаратная или программная часть) фиксированы и выступают в роли условий. Иногда [Agrawal, 2004] часть конфигурации известна заранее;

2. Информация о нагрузке дополнена достаточно подробной информацией о данных, к которым обращаются запросы (количество и характеристики атрибутов таблиц и т.д.);

3. Пользовательские ограничения, не связанные напрямую с системой (могут считаться частью нагрузки). Например, передача данных между узлами в распределенной компьютерной сети осуществляется за плату, и для решения задачи выделен определенный бюджет.

О решении

Для нахождения решения необходимы:

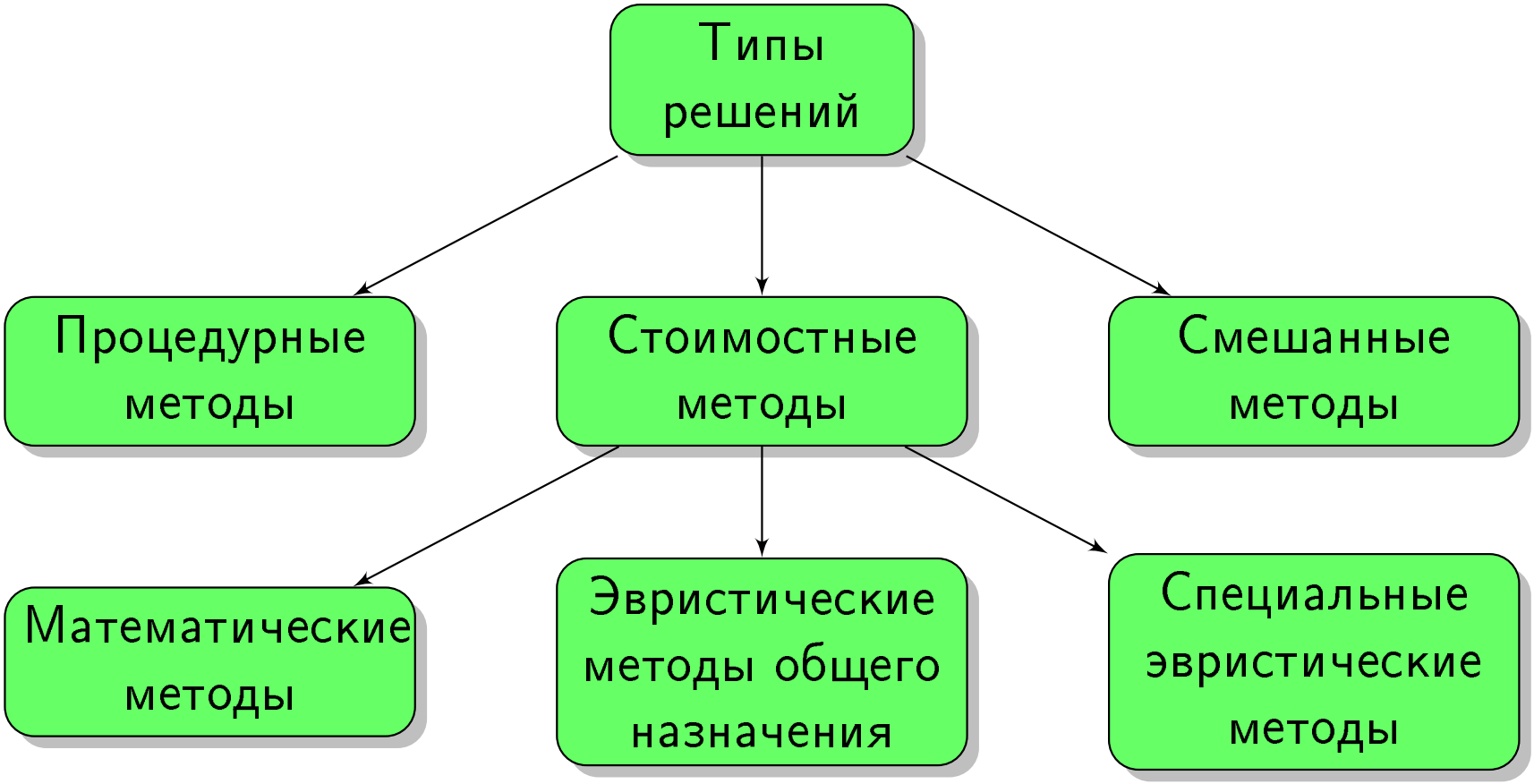
1. способ оценивать конфигурации $\rightarrow$ стоимостная модель, учитывающую характеристики аппаратной конфигурации, характеристики нагрузки и, возможно, другие пользовательские ограничения.

2. способ перебирать конфигурации
   - необходимо проверить экспоненциальное количество решений, например количество вариантов вертикального фрагментирования равно числу Белла $B_{n+1} = \sum_{k=0}^{n} \binom{N}{K} B_k$
   - это много:

$$B_1 = 1, ..., B_5 = 15, ..., B_{10} = 21147, ..., B_{15} = 190899322$$

$$B_{20} = 5832742205057, ..., B_{25} = 4638590332229999353$$

“Классификация” методов решения



Процедурные методы

- процедура не работает с моделью, описывающей поведение системы, и не обращается к функции стоимости выполнения запросов для оценки конкретной конфигурации;
- процедура последовательно формирует ответ, исходя, часто из неочевидных, посылок о том, каким должно быть решение;
- процедура иногда сопровождается доказательством или обоснованием, почему она дает хорошее решение.

Примеры: графовый подход к вертикальному фрагментированию [Navathe, 1989] и методология для фрагментирования и размещения [Tamhankar, 1998].

Стоимостные методы (1)

Математические методы — решение экстремальной задачи с помощью методов целочисленного программирования (integer programming).

1. Исторически первый способ решения задачи, доминировали в 60-90 гг, особенно в задачах размещения.
2. Метод решения (линеаризация уравнений) составлял значительную часть предмета исследований. Ныне — редок, используются матпакеты [Papadomanolakis, 2007].
3. Неэффективен. Пример [March, 1995]: задача размещения для трех отношений имеющих по три фрагмента, 4 узла, 6 запросов давала 1200 переменных и ограничений. Авторы утверждают что в то время такой размер мог решаться только на суперкомпьютерах.
4. Сложность линеаризации [Hammer, 1979].

Пример постановки в виде экстремальной задачи [March, 1995]

$$\textit{Min Cost} = \sum_k \sum_j f(k,j) \sum_m (COM(k,j,m) + IO(k,j,m) + CPU(k,j,m)) + \sum_t STO(t)$$

X_{it} = 1 if file fragment i is stored at node t
0 otherwise

Z_{kit} = 1 if query k uses file fragment i at node t
0 otherwise

Y_{kmt} = 1 if step m of query k is done at node t
0 otherwise

$\sum_t X_{it} \geq 1$ for all file fragments, i = 1, 2, ... , number of fragments (all file fragments must be stored at one or more nodes)

$Z_{kit} \leq X_{it}$ for all queries, k = 1, 2, ... , number of queries for all file fragments, i = 1, 2, ... , number of fragments for all nodes, t = 1, 2, ... , number of nodes (a file fragment cannot be accessed from a node unless it is stored at that node)

$\sum_t Y_{kmt} = 1$ for all queries, k = 1, 2, ... , number of queries for all steps m, m = 1, 2, ... , number of steps for query k (all query steps must be processed at some node).

Стоимостные методы (2)

Специальные эвристические методы: методы, разработанные специально для решения подобных постановок.

- Результат отказа от матметодов, одна из первых работ [Hammer, 1979];
- В настоящее время — подход доминирует в работах с топовых конференций;
- В современных работах модель довольно часто убрана в оптимизатор, используются what-if вызовы;
- Считается (например [Nehme, 2011]), что они лучше чем эвристические методы общего назначения, однако ...

Стоимостные методы (3)

Эвристические методы общего назначения: генетические алгоритмы, алгоритмы поиска локального минимума, метод отжига, муравьиные алгоритмы, particle swarm optimization и т.п.

- Модель присутствует в явном виде;
- Популярен в задачах размещения, хотя и не ограничен ими;
- Работ этого типа много, я смотрел десятки-сотни работ.

Смешанные методы

Были очень популярны в 80-90-е, когда вычислительные ресурсы были ограничены.
Пример [Navathe, 1984], в нем:

- применение процедуры, упрощение задачи;
- выбор финального решения с использованием стоимостной функции.

Attribute	5	1	7	2	8	3	9	10	4	6
5	75	75	50	25	25	25	25	0	0	0
1	75	75	50	25	25	25	25	0	0	0
7	50	50	85	60	60	25	25	0	0	0
2	25	25	60	110	110	75	75	0	0	0
8	25	25	60	110	110	75	75	0	0	0
3	25	25	25	75	75	115	115	15	15	15
9	25	25	25	75	75	115	115	15	15	15
10	0	0	0	0	0	15	15	40	40	40
4	0	0	0	0	0	15	15	40	40	40
6	0	0	0	0	0	15	15	40	40	40

(c)

Пример современной стоимостной работы (1)

Задача [Agrawal, 2004]: выбрать атрибуты для индексирования, провести горизонтальное (хеш-фрагментирование и интервальное) и вертикальное фрагментирование для известной схемы и известного набора запросов.

- Структура физического уровня — тройка:

  (объект, метод горизонтального фрагментирования, упорядоченный набор колонок);

- Объект — таблица + метод доступа (хип, индексы);
  - Объект может быть подтаблицей.

Пример современной стоимостной работы (2)

Конфигурация определяется как набор допустимых структур физического уровня. Допустимая структура — та, которую можно реализовать в данной схеме базы данных. Например, невозможно иметь два кластеризованных индекса для одной таблицы.

Задача — найти такую конфигурацию, состоящую из структур физического уровня указанных типов, которая:

1. Минимизирует суммарную стоимость исполнения известного набора запросов;
2. Укладывается в указанное ограничение на объем данных;
3. Выполняет другие ограничения на структуры физического уровня.

Пример современной стоимостной работы (3)

Постановка:

1. Конфигурация системы C — набор структур физического уровня, ограничения на них, а также ограничение на объем занимаемой этими структурами памяти;
2. Нагрузка W представлена в виде набора запросов;
3. Производительность P, вычисляемая как суммарное время выполнения всех запросов.

Функция f, связывающая эти компоненты, представлена оптимизатором запросов в режиме what-if.

Пример современной стоимостной работы (4)

Этапы решения:

1. Предварительный отсев вертикальных фрагментов. Цель — уменьшить количество вертикальных фрагментов, учитываемых в дальнейшем. Из рассмотрения исключаются те, что не несут потенциальной выгоды, и, скорее всего, не будут присутствовать в решении. Оценка выгоды производится на основании модели, учитывающей интенсивность использования атрибутов (конкретно здесь — frequent-itemset generation).

   Используемые атрибуты берутся из запросов.
2. Выбор базовых структур. Выбираются наиболее выгодные физические структуры для каждого запроса в отдельности. Для оценки применяются what-if вызовы оптимизатора;

Пример современной стоимостной работы (5)

Этапы решения:

3. Этап слияния. Множество структур, полученных на предыдущем шаге, дополняется структурами, полученными в результате их смешивания. Этот шаг используется, например, для борьбы с чрезмерной специализацией структур;
4. Поиск итогового решения среди структур, полученных на предыдущем шаге. Авторы работы разработали алгоритм $Greedy(m, k)$, гарантирующий оптимальный ответ при выборе m структур из k, а для поиска остальных использующий жадный алгоритм.

Определение, свойства (1)

Горизонтальное фрагментирование

Горизонтальное фрагментирование — разделение отношения на несколько отношений, каждое из которых содержит подмножество записей исходного.

Горизонтальное фрагментирование:

1. повышает производительность СУБД за счет уменьшения объемов данных, которые потребуется обработать при выполнении запроса. Данные стремятся локализовать в одном или нескольких фрагментах меньших объемов, чем исходное отношение;
2. свойства: полнота, восстановимость, попарное непересечение фрагментов [Özsu, 1999];

Определение, свойства (2)

Горизонтальное фрагментирование

Горизонтальное фрагментирование — разделение отношения на несколько отношений, каждое из которых содержит подмножество записей исходного.

3. полезно не только в распределенных, но и в централизованных СУБД;
4. в отличии от индексов и материализованных представлений не реплицирует данные, не привносит дополнительных затрат при обновлениях и в хранении;
5. мощный инструмент администратора базы данных, поддерживающийся практически в каждом диалекте SQL DDL.

Типы горизонтального фрагментирования (1): [Bellatreche, 2009, Taniar, 2008]

Атрибутное фрагментирование:

- Интервальное фрагментирование (range data partitioning). Домены атрибутов фрагментирования делятся на несколько непересекающихся интервалов. Запись относится к фрагменту, если значение атрибута(ов) принадлежит к характеризующему интервалу.
- Фрагментирование с помощью хеш-функции (hash data partitioning). Принадлежность записи к конкретному фрагменту определяется хеш-функцией от атрибута(ов).
- Списковое фрагментирование (list partitioning) — каждый фрагмент задается списком допустимых значений.
- Виртуальное колоночное фрагментирование (virtual column partitioning). В отношение добавляется виртуальный атрибут, значение которого задается формулой от набора других.

Типы горизонтального фрагментирования (2)

Современные СУБД поддерживают и смешанное (composite partitioning):

- range-hash: сначала интервальное, потом хеш-функция;
- на каждом шаге атрибуты могут различны.

Изучалось в ранних работах [Copeland, 1988], [Ghandeharizadeh, 1990], [Ghandeharizadeh, 1992], [Ghandeharizadeh, 1994].

Типы горизонтального фрагментирования (3)

Фрагментирование, не зависящее от значений атрибутов:

- Случайное равномерное фрагментирование (random-equal data partitioning). Записи распределяются между фрагментами случайным образом, но так, чтобы фрагменты были равны по размеру.

- Круговое фрагментирование (round-robin data partitioning). Является наиболее популярной реализацией этого класса методов фрагментирования. Записи относятся к фрагментам последовательно, по кругу, в зависимости от номера.

Типы горизонтального фрагментирования (4): сравнение

Атрибутное фрагментирование:

- Позволяет использовать сложную семантику при оптимизации запроса. Пример: интервальное позволит задействовать только один узел из набора;
- Склонно к асимметрии (skew) размеров фрагментов.

Фрагментирование, не зависящее от значений атрибутов:

- Использование семантики невозможно;
- Ассиметрия отсутствует.

Формирование фрагментов: кластеризация записей (record clustering)

Идея: создаем фрагменты в результате исполнения запросов, так чтобы совместно запрашиваемые данные находились в одном фрагменте. Для достижения этого предлагается, в частности, использовать различные алгоритмы кластеризации.

Избранные работы: [Bell, 1984, Moghrabi, 2000] и серия работ N. Gorla [Gorla, 2003].

Идея похожа на современное направление адаптивного индексирования (adaptive indexing) [Idreos, 2007].

Формирование фрагментов: использование предикатов

Идея: создавать фрагменты на основании предикатов в запросах [Ceri, 1982, Ceri, 1983].

- Очень красивая и влиятельная идея, породила массу работ–последователей, включая смежные области — вертикальное фрагментирование [Zhang, 1994], фрагментирование хранилищ данных [Noaman, 1999], фрагментирование в объектных СУБД [Bellatreche, 2000].
- Метод получения фрагментов (хорошо описан в [Özsu, 1999]) напоминает алгоритм Ханта (Hunts algorithm) — алгоритм для построения дерева решений.
- О применении на практике мне ничего не известно.

Формирование фрагментов: распространение по ключам

Работа [Ceri, 1983] делит методы фрагментирования на основные (primary partitioning) и производные (derived).

Основное фрагментирование отношения определяется только значениями собственных атрибутов, в то время как производное дополнительно зависит и от фрагментов другого отношения.

Производное фрагментирование применяется при фрагментировании отношения, связанного посредством внешнего ключа с некоторым другим. Идея: сформировать фрагменты таким образом, чтобы каждому фрагменту первого отношения сопоставлялся фрагмент второго.

Формирование фрагментов: распространение по ключам

1. Этот метод позволяют получать цепочки фрагментирования наборов отношений, связанных ключами;
2. В современных работах [Bellatreche, 2009] эти два типа фрагментирования называются моно фрагментирование (mono) и таблице-зависимое (table-dependent);
3. Ранняя реализация на практике [Didriksen, 1995], интересный факт — цепочек длины больше чем 3 не бывает;
4. В СУБД Oracle 11gR1 сделали поддержку в 2008 году [Eadon, 2008], называют ссылочное (referential).

Размещение данных (1)

Задача горизонтального фрагментирования часто рассматривается в условиях распределенной СУБД, тогда она решается совместно с задачей размещения данных.

Цели размещения данных:

- Повышение производительности;
- Повышение надежности;
- Балансировка нагрузки.

Как самостоятельная задача это наиболее старая из всех задач о выборе структур физического уровня, происходит от задачи о размещении файла в компьютерной сети.

Размещение данных (2): типы

Типы [Oates, 2001]:

1. Чистое размещение данных (data allocation) — размещение только фрагментов данных, с целью увеличения производительности или повышения надежности системы. Подавляющее большинство рассмотренных работ относятся к этому классу.

2. Размещение данных и запросов (data and query allocation) — постановка размещает еще и запросы.

3. Размещение пользовательской нагрузки (user load allocation) — постановка размещает только запросы.

Типы постановок задачи размещения данных

Задача размещения бывает поставлена в двух вариантах [Ceri, 1987]:

- Неизбыточное размещение, при котором каждый фрагмент или отношение находится ровно на одном узле;
- Избыточное размещение, то есть репликация, может применяться для повышения надежности или производительности СУБД.

Размещение данных (3): постановка задачи

Постановка задачи размещения данных описывается схемой, однако требуются некоторые уточнения:

1. Аппаратная конфигурация включает в себя число узлов и их характеристики. Также описываются параметры сети.

2. Искомая конфигурация физического уровня должна сопоставлять каждому фрагменту узел и при этом не нарушать ограничения (например, объем жесткого диска узла не должен быть превышен).

3. В качестве метрики P дополнительно может выступать суммарный объем переданных по сети данных. Бывают и многокомпонентные метрики, но всё-равно, передача по сети вносит значительный вклад.

Размещение данных и фрагментирование

Один из острейших вопросов данной области исследований — целесообразно ли разрабатывать метод распределения данных в отрыве от фрагментирования (итеративный дизайн [Bellatreche, 2009, Bellatreche, 2010]) или же следует решать эти два аспекта совместно (комбинированный или интегрированный [Zilio, 2004] дизайн).

- Комбинированный недостаточно гибок и очень трудоемок.
- Итеративное решение хуже, теряется часть семантики.

В итоге, данная проблема привела к появлению двух независимых сообществ исследователей [Bellatreche, 2009].

Принцип локальности ссылки

Движущий принцип как размещения, так и фрагментирования — “локальность ссылки” (locality of reference) [Tamhankar, 1998].

Идея: делаем совместное хранение данных для данных, которые будут совместно запрашиваться. Причем, идея применяется ко всему: будь то записи внутри фрагмента, фрагменты внутри узла и пр. Во многих случаях это позволяет снизить стоимость обработки запроса. Согласно [Ceri, 1987] в хорошо спроектированной базе данных, 90% обращений к данным должны быть локальными.

Значительное количество работ опирается на данную идею.

Исторический экскурс

Задача о размещении данных в контексте СУБД “выросла” из задачи о размещении файла в компьютерных сетях. Самая первая найденная мной работа [Chu, 1969].
Почему выросла:

- Заимствованные подходы оказались слабо приспособленными: модели не учитывали специфику исполнения запросов в СУБД (например, необходимость узлов обмениваться информацией друг с другом при выполнении реляционной операции соединения).

Обзор работ данной тематики, представлено более десяти типов моделей для задачи о размещении файлов [Dowdy, 1982].
Классифицируют по метрике, количеству копий и методу решения.

Модели для размещения данных в контексте СУБД

Классификация моделей для размещения данных в контексте СУБД:

1. Модели, учитывающие проведение распределенных транзакций. В качестве параметров в них дополнительно может фигурировать стоимость запроса замка или количество сообщений, которые необходимо передать;
2. Модели, требующие обеспечения определенного уровня надежности;
3. Модели, учитывающие характер и стоимости отдельных реляционных операторов (например, оператора соединения), составляющих план запроса.

Ссылок очень много, есть в статье.

Динамическая постановка задачи и миграция данных

Первые работы по размещению данных рассматривали задачу в статическом контексте — когда ни запросы, ни данные не изменяются.

Однако со временем нагрузка может меняться или уточняться, может поменяться топология сети [Rivera-Vega, 1990] и т.д. Это может негативно сказаться на производительности системы.

Поэтому позже появились работы, изучавшие и миграцию (migration) данных [Rivera-Vega, 1990], [Brunstrom, 1995], [Hauglid, 2010], [Kamali, 2011] в условиях меняющейся нагрузки.

Общая схема решения задачи в динамической постановке

В результате миграции данных обеспечивается выравнивание загрузки узлов при исполнении запросов. То есть, можно сказать, что в данных работах решается задача балансировки нагрузки.

1. Собирается статистика обращения к фрагментам (например, с помощью счетчиков);
2. Через определенный промежуток времени или по выполнению определенных условий (например, появлению новых запросов) производится анализ существующей конфигурации;
3. В зависимости от пороговых значений счетчиков или потенциальной стоимости новой конфигурации принимается решение об изменениях — миграции фрагмента на другой узел или его изменении (расщеплении или слиянии с другим).

Коммерческие СУБД и современные работы по фрагментированию и размещению

- DB2: индексы, материализованные представления, и структуры многомерной кластеризации [Zilio, 2004a]
- Microsoft SQL Server: индексы (семейство работ), вертикальное фрагментирование [Chaudhuri, 2007]
- Microsoft SQL Server 2008 Parallel Data Warehouse: фрагментирование, размещение с репликацией [Nehme, 2011]
- PostgreSQL: фрагменты, индексы [Alagiannis, 2010, Maier, 2010, Papadomanolakis, 2004]
- ...

Статья

Ввиду ограниченности времени смотрите в старой статье:

Г. А. Чернышев, “Обзор подходов к организации физического уровня в СУБД”, Тр. СПИИРАН, 24 (2013), 222–276. url: http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=trspy&paperid=580&option_lang=rus

Ссылки I


Hoffer J. A., Severance D. G. The use of cluster analysis in physical data base design // Proceedings of the 1st International Conference on Very Large Data Bases. VLDB '75. New York, NY, USA : ACM, 1975. P. 69–86. URL: http://doi.acm.org/10.1145/1282480.1282486.


Chu W. W. Optimal file allocation in a multiple computer system // IEEE Trans. Comput. 1969. Vol. 18, no. 10. P. 885–889. URL: http://dx.doi.org/10.1109/T-C.1969.222542.


Lightstone S. Physical database design for relational databases // Encyclopedia of Database Systems / Ed. by Ling Liu, M.Tamer ´OZSU. Springer US, 2009. P. 2108–2114. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_644.


Chaudhuri S., Weikum G. Self-management technology in databases // Encyclopedia of Database Systems / Ed. by Ling Liu, M.Tamer ´Ozsu. Springer US, 2009. P. 2550–2555. URL: http://dx.doi.org/10.1007/978-0-387-39940-9_334.

Ссылки II


Agrawal S., Chu E., Narasayya V. Automatic physical design tuning: workload as a sequence // Proceedings of the 2006 ACM SIGMOD international conference on Management of data. SIGMOD '06. New York, NY, USA : ACM, 2006. P. 683–694. URL: http://doi.acm.org/10.1145/1142473.1142549


Database tuning advisor for Microsoft SQL Server 2005 / Sanjay Agrawal, Surajit Chaudhuri, Lubor Kollar et al. // Proceedings of VLDB. 2004. P. 1110–1121.


Navathe S. B., Ra M. Vertical partitioning for database design: a graphical algorithm // Proceedings of the 1989 ACM SIGMOD international conference on Management of data. SIGMOD '89. New York, NY, USA : ACM, 1989. P. 440–450. URL: http://doi.acm.org/10.1145/67544.66966.


Tamhankar A. M., Ram S. Database fragmentation and allocation: an integrated methodology and case study // Trans. Sys. Man Cyber. Part A. 1998. Vol. 28, no. 3. P. 288–305. URL: http://dx.doi.org/10.1109/3468.668961.


March S. T., Rho S. Allocating data and operations to nodes in distributed database design // IEEE Trans. on Knowl. and Data Eng. 1995. Vol. 7, no. 2. P. 305–317. URL: http://dx.doi.org/10.1109/69.382299.

Ссылки III


Nehme R., Bruno N. Automated partitioning design in parallel database systems // Proceedings of the 2011 international conference on Management of data. SIGMOD '11. New York, NY, USA : ACM, 2011. P. 1137–1148. URL: http://doi.acm.org/10.1145/1989323.1989444.


Vertical partitioning algorithms for database design / Shamkant Navathe, Stefano Ceri, Gio Wiederhold, Jinglie Dou // ACM Trans. Database Syst. 1984. Vol. 9. P. 680–710. URL: http://doi.acm.org/10.1145/1994.2209.


Agrawal S., Narasayya V., Yang B. Integrating vertical and horizontal partitioning into automated physical database design // Proceedings of the 2004 ACM SIGMOD international conference on Management of data. SIGMOD '04. New York, NY, USA : ACM, 2004. P. 359–370. URL: http://doi.acm.org/10.1145/1007568.1007609.


Hammer M., Niamir B. A heuristic approach to attribute partitioning // Proceedings of the 1979 ACM SIGMOD international conference on Management of data. SIGMOD '79. New York, NY, USA : ACM, 1979. P. 93–101. URL: http://doi.acm.org/10.1145/582095.582110.

Ссылки IV


Papadomanolakis S., Ailamaki A. An integer linear programming approach to database design // Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop. ICDEW '07. Washington, DC, USA : IEEE Computer Society, 2007. P. 442–449. URL: http://dx.doi.org/10.1109/ICDEW.2007.4401027


Özsu M. T., Valduriez P. Principles of distributed database systems (2nd ed.). Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1999. ISBN: 0-13-659707-6.


Bellatreche L., Woameno K. Y. Dimension table driven approach to referential partition relational data warehouses // Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP. DOLAP '09. New York, NY, USA : ACM, 2009. P. 9–16. URL: http://doi.acm.org/10.1145/1651291.1651294.


High Performance Parallel Database Processing and Grid Databases / David Taniar, Clement H. C. Leung, Wenny Rahayu, Sushant Goel. Wiley Publishing, 2008. ISBN: 0470107626, 9780470107621.

Ссылки V


Applying genetic algorithms in database partitioning / Vincent Ng, Dik Man Law, Narasimhaiah Gorla, Chi Kong Chan // Proceedings of the 2003 ACM symposium on Applied computing. SAC '03. New York, NY, USA : ACM, 2003. P. 544–549. URL: http://doi.acm.org/10.1145/952532.952639.


Moghrabi I., Makholian R. A new approach to clustering records in information retrieval systems // Information Retrieval. 2000. Vol. 3. P. 105–126. 10.1023/A:1009901830009. URL: http://dx.doi.org/10.1023/A:1009901830009.


Bell D. A. Physical record clustering in databases // Kybernetes. 1984. Vol. 13. P. 31–37.


Idreos S., Kersten M. L., Manegold S. Database cracking // CIDR. www.cidrdb.org, 2007. P. 68–78.


Ceri S., Negri M., Pelagatti G. Horizontal data partitioning in database design // Proceedings of the 1982 ACM SIGMOD international conference on Management of data. SIGMOD '82. New York, NY, USA : ACM, 1982. P. 128–136. URL: http://doi.acm.org/10.1145/582353.582376.

Ссылки VI


Ceri S., Navathe S., Wiederhold G. Distribution design of logical database schemas // IEEE Transactions on Software Engineering. 1983. Vol. 9. P. 487–504.


Zhang Y., Orlowska M. E. On fragmentation approaches for distributed database design // Information Sciences - Applications. 1994. Vol. 1, no. 3. P. 117–132. URL: http://www.sciencedirect.com/science/article/pii/1069011594900051.


Noaman A. Y., Barker K. A horizontal fragmentation algorithm for the fact relation in a distributed data warehouse // Proceedings of the eighth international conference on Information and knowledge management. CIKM '99. New York, NY, USA : ACM, 1999. P. 154–161. URL: http://doi.acm.org/10.1145/319950.319972.


Bellatreche L., Karlapalem K., Simonet A. Algorithms and support for horizontal class partitioning in object-oriented databases // Distributed and Parallel Databases. 2000. Vol. 8. P. 155–179. 10.1023/A:1008745624048. URL: http://dx.doi.org/10.1023/A:1008745624048.

Ссылки VII

Supporting table partitioning by reference in oracle / George Eadon, Eugene Inseok Chong, Shrikanth Shankar et al. // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. SIGMOD '08. New York, NY, USA : ACM, 2008. P. 1111–1122. URL: http://doi.acm.org/10.1145/1376616.1376727.

Oates M. J., Corne D. W. Global web server load balancing using evolutionary computational techniques // Soft Computing. 2001. Vol. 5, no. 4. P. 297–312. URL: http://dx.doi.org/10.1007/s005000100103

Bellatreche L., Benkrid S. A joint design approach of partitioning and allocation in parallel data warehouses // Data Warehousing and Knowledge Discovery / Ed. by Torben Pedersen, Mukesh Mohania, A Tjoa. Springer Berlin / Heidelberg, 2009. Vol. 5691 of Lecture Notes in Computer Science. P. 99–110. 10.1007/978-3-642-03730-6 9. URL: http://dx.doi.org/10.1007/978-3-642-03730-6_9.

Ссылки VIII

Bellatreche L., Cuzzocrea A., Benkrid S. F&A: a methodology for effectively and efficiently designing parallel relational data warehouses on heterogenous database clusters // Proceedings of the 12th international conference on Data warehousing and knowledge discovery. DaWaK'10. Berlin, Heidelberg : Springer-Verlag, 2010. P. 89–104. URL: http://dl.acm.org/citation.cfm?id=1881923.1881934.

DB2 design advisor: integrated automatic physical database design / Daniel C. Zilio, Jun Rao, Sam Lightstone et al. // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 1087–1097. URL: http://dl.acm.org/citation.cfm?id=1316689.1316783.

Ceri S., Pernici B., Wiederhold G. Distributed database design methodologies // Proceedings of the IEEE. 1987. Vol. 75, no. 5. P. 533–546.

Dowdy L. W., Foster D. V. Comparative models of the file assignment problem // ACM Comput. Surv. 1982. Vol. 14, no. 2. P. 287–313. URL: http://doi.acm.org/10.1145/356876.356883.

Ссылки IX


Rivera-Vega P., Varadarajan R., Navathe S. Scheduling data redistribution in distributed databases // Data Engineering, 1990. Proceedings. Sixth International Conference on. 1990. feb. P. 166–173.


Brunstrom A., Leutenegger S. T., Simha R. Experimental evaluation of dynamic data allocation strategies in a distributed database with changing workloads // Proceedings of the fourth international conference on Information and knowledge management. CIKM '95. New York, NY, USA : ACM, 1995. P. 395–402. URL: http://doi.acm.org/10.1145/221270.221652.


Hauglid J. O., Ryeng N. H., Nørvåg K. Dyfram: dynamic fragmentation and replica management in distributed database systems // Distrib. Parallel Databases. 2010. Vol. 28. P. 157–185. URL: http://dx.doi.org/10.1007/s10619-010-7068-1.


Kamali S., Ghodsnia P., Daudjee K. Dynamic data allocation with replication in distributed systems // Performance Computing and Communications Conference (IPCCC), 2011 IEEE 30th International. 2011. nov. P. 1–8.

Ссылки X

Ghandeharizadeh S., DeWitt D. J. Hybrid-range partitioning strategy: A new declustering strategy for multiprocessor database machines // Proceedings of the 16th International Conference on Very Large Data Bases. VLDB '90. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1990. P. 481–492. URL: http://dl.acm.org/citation.cfm?id=645916.671988.

Ghandeharizadeh S., DeWitt D. J. Magic: A multiattribute declustering mechanism for multiprocessor database machines // IEEE Trans. Parallel Distrib. Syst. 1994. Vol. 5, no. 5. P. 509–524. URL: http://dx.doi.org/10.1109/71.282561.

Ghandeharizadeh S., DeWitt D. J., Qureshi W. A performance analysis of alternative multi-attribute declustering strategies // SIGMOD Rec. 1992. Vol. 21, no. 2. P. 29–38. URL: http://doi.acm.org/10.1145/141484.130293.

Data placement in Bubba / George Copeland, William Alexander, Ellen Boughter, Tom Keller // Proceedings of the 1988 ACM SIGMOD international conference on Management of data. SIGMOD '88. New York, NY, USA : ACM, 1988. P. 99–108. URL: http://doi.acm.org/10.1145/50202.50213.

Ссылки XI


Didriksen T., Galindo-Legaria C. A., Dahle E. Database de-centralization — a practical approach // Proceedings of the 21th International Conference on Very Large Data Bases. VLDB '95. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1995. P. 654–665. URL: http://dl.acm.org/citation.cfm?id=645921.673313.


Mehta M., DeWitt D. J. Data placement in shared-nothing parallel database systems // The VLDB Journal. 1997. Vol. 6. P. 53–72. URL: http://dx.doi.org/10.1007/s007780050033.


Partitioning key selection for a shared-nothing parallel database system : Rep. : IBM Research Report RC 19820 (87739) / IBM T.J. Watson Research Center ; Executor: Daniel C. Zilio, Anant Jhingran, Sriram Padmanabhan : 1994.


Zilio D. C. Physical database design decision algorithms and concurrent reorganization for parallel database systems : Ph. D. thesis / Daniel Costante Zilio ; University of Toronto. Toronto, Ont., Canada, Canada, 1998. AAINQ35386.

Ссылки XII

Rahm E., Marek R. Analysis of dynamic load balancing strategies for parallel shared nothing database systems // Proceedings of the 19th International Conference on Very Large Data Bases. VLDB '93. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993. P. 182–193. URL: http://portal.acm.org/citation.cfm?id=645919.672662.

Bruno N., Chaudhuri S. Automatic physical database tuning: a relaxation-based approach // Proceedings of the 2005 ACM SIGMOD international conference on Management of data. SIGMOD '05. New York, NY, USA : ACM, 2005. P. 227–238. URL: http://doi.acm.org/10.1145/1066157.1066184.

PARINDA: an interactive physical designer for PostgreSQL / Cristina Maier, Debabrata Dash, Ioannis Alagiannis et al. // Proceedings of the 13th International Conference on Extending Database Technology. EDBT '10. New York, NY, USA : ACM, 2010. P. 701–704. URL: http://doi.acm.org/10.1145/1739041.1739131.

Finkelstein S., Schkolnick M., Tiberio P. Physical database design for relational databases // ACM Trans. Database Syst. 1988. Vol. 13. P. 91–128. URL: http://doi.acm.org/10.1145/42201.42205.

Ссылки XIII


Automating physical database design in a parallel database / Jun Rao, Chun Zhang, Nimrod Megiddo, Guy Lohman // Proceedings of the 2002 ACM SIGMOD international conference on Management of data. SIGMOD '02. New York, NY, USA : ACM, 2002. P. 558–569. URL: http://doi.acm.org/10.1145/564691.564757.


DB2 design advisor: integrated automatic physical database design / Daniel C. Zilio, Jun Rao, Sam Lightstone et al. // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 1087–1097. URL: http://dl.acm.org/citation.cfm?id=1316689.1316783.


Lightstone S. S., Bhattacharjee B. Automated design of multidimensional clustering tables for relational databases // Proceedings of the Thirtieth international conference on Very large data bases - Volume 30. VLDB '04. VLDB Endowment, 2004. P. 1170–1181. URL: http://dl.acm.org/citation.cfm?id=1316689.1316789.


Recommending materialized views and indexes with the IBM DB2 design advisor / D.C. Zilio, C. Zuzarte, S. Lightstone et al. // Autonomic Computing, 2004. Proceedings. International Conference on. 2004. may. P. 180–187.

Ссылки XIV

Nehme R., Bruno N. Automated partitioning design in parallel database systems // Proceedings of the 2011 international conference on Management of data. SIGMOD '11. New York, NY, USA : ACM, 2011. P. 1137–1148. URL: http://doi.acm.org/10.1145/1989323.1989444.

Chaudhuri S., Narasayya V. Self-tuning database systems: a decade of progress // Proceedings of the 33rd international conference on Very large data bases. VLDB '07. VLDB Endowment, 2007. P. 3–14. URL: http://dl.acm.org/citation.cfm?id=1325851.1325856.

An automated, yet interactive and portable DB designer / Ioannis Alagiannis, Debabrata Dash, Karl Schnaitter et al. // Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. SIGMOD '10. New York, NY, USA : ACM, 2010. P. 1183–1186. URL: http://doi.acm.org/10.1145/1807167.1807314.

PARINDA: an interactive physical designer for PostgreSQL / Cristina Maier, Debabrata Dash, Ioannis Alagiannis et al. // Proceedings of the 13th International Conference on Extending Database Technology. EDBT '10. New York, NY, USA : ACM, 2010. P. 701–704. URL: http://doi.acm.org/10.1145/1739041.1739131.

Ссылки XV


Papadomanolakis S., Ailamaki A. AutoPart: automating schema design for large scientific databases using data partitioning // Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. 2004. june. P. 383–392.


Parallel OLAP query processing in database clusters with data replication / Alexandre Lima, A., Camille Furtado, Patrick Valduriez, Marta Mattoso // Distributed and Parallel Databases. 2009. Vol. 25, no. 1-2. P. 97–123. URL: http://hal.inria.fr/inria-00482183.


Papadomanolakis S., Dash D., Ailamaki A. Efficient use of the query optimizer for automated physical design // Proceedings of the 33rd international conference on Very large data bases. VLDB '07. VLDB Endowment, 2007. P. 1093–1104. URL: http://dl.acm.org/citation.cfm?id=1325851.1325974.


Colt: continuous on-line tuning / Karl Schnaitter, Serge Abiteboul, Tova Milo, Neoklis Polyzotis // Proceedings of the 2006 ACM SIGMOD international conference on Management of data. SIGMOD '06. New York, NY, USA : ACM, 2006. P. 793–795. URL: http://doi.acm.org/10.1145/1142473.1142592.

Ссылки XVI

Bruno N., Chaudhuri S. An online approach to physical design tuning // 2007 IEEE 23rd International Conference on Data Engineering. 2007. P. 826–835.

Multi-dimensional clustering: a new data layout scheme in db2 / Sriram Padmanabhan, Bishwaranjan Bhattacharjee, Tim Malkemus et al. // Proceedings of the 2003 ACM SIGMOD international conference on Management of data. SIGMOD '03. New York, NY, USA : ACM, 2003. P. 637–641. URL: http://doi.acm.org/10.1145/872757.872835.

Röhm U., Böhm K., Schek H.-J. OLAP query routing and physical design in a database cluster // Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology. EDBT '00. London, UK, UK : Springer-Verlag, 2000. P. 254–268. URL: http://dl.acm.org/citation.cfm?id=645339.650130.

Gebaly K. E., Aboulnaga A. Robustness in automatic physical database design // Proceedings of the 11th international conference on Extending database technology: Advances in database technology. EDBT '08. New York, NY, USA : ACM, 2008. P. 145–156. URL: http://doi.acm.org/10.1145/1353343.1353365.

Ссылки XVII


Bruno N. A critical look at the tab benchmark for physical design tools // SIGMOD Rec. 2007. Vol. 36, no. 4. P. 7–12. URL: http://doi.acm.org/10.1145/1361348.1361349.


Goals and benchmarks for autonomic configuration recommenders / Mariano P. Consens, Denilson Barbosa, Adrian Teisanu, Laurent Mignet // Proceedings of the 2005 ACM SIGMOD international conference on Management of data. SIGMOD '05. New York, NY, USA : ACM, 2005. P. 239–250. URL: http://doi.acm.org/10.1145/1066157.1066185.


Borovica R., Alagiannis I., Ailamaki A. Automated physical designers: What you see is (not) what you get // Proceedings of the Fifth International Workshop on Testing Database Systems. DBTest '12. New York, NY, USA : ACM, 2012. P. 9:1–9:6. URL: http://doi.acm.org/10.1145/2304510.2304522.


Schnaitter K., Polyzotis N. A benchmark for online index selection // Proceedings of the 2009 IEEE International Conference on Data Engineering. ICDE '09. Washington, DC, USA : IEEE Computer Society, 2009. P. 1701–1708. URL: http://dx.doi.org/10.1109/ICDE.2009.166.