Глава 19. Борьба за производительность Power Pivot

Это продолжение перевода книги Роб Колли. Формулы DAX для Power Pivot. Главы не являются независимыми, поэтому рекомендую начать сначала.

Предыдущая глава Содержание Следующая глава

Исследования показывают, что люди воспринимают «сейчас» как три секунды. Что-то, что занимает три секунды или меньше, происходит «сейчас» и что-то, что длится дольше, приводит к ожиданию. Кто-то из Microsoft недавно спросил: «Как вы думаете, как долго пользователи будут ждать, когда они нажмут на срез?». Я ответил: «Это должно быть быстро. Их не волнует, что за этим стоит много данных. Пределы человеческого терпения не соответствуют нашим проблемам объема данных или их сложности». Когда мы готовим интерактивные отчеты или дашборды, мы должны иметь в виду, что скорость взаимодействия имеет решающее значение. Если ожидание превысит три секунды, мы рискуем потерять потребителя. (Интересная статья по теме – Обнимания соблюдают правило 3 секунд.)

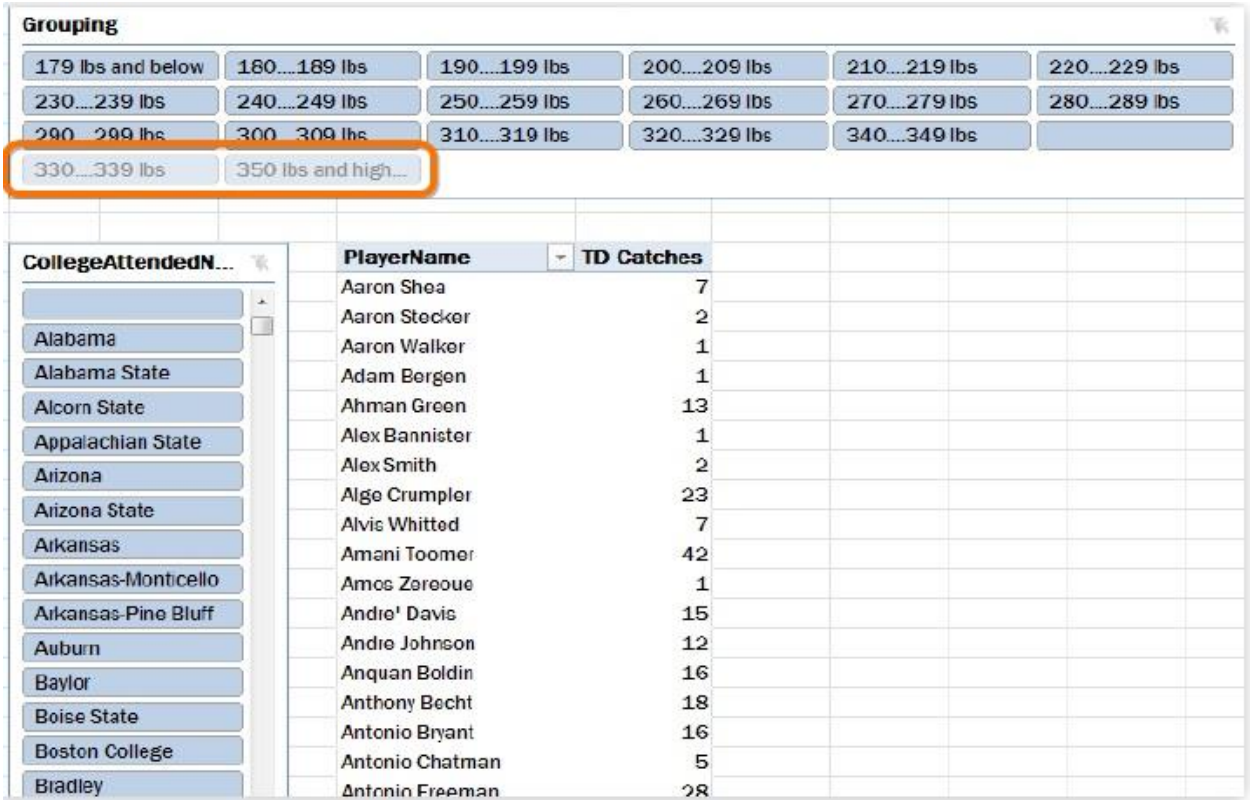

Рис. 19.1. На срезах нет фильтров, но ни один игрок в диапазонах 330–339 фунтов и 350 и более фунтов никогда не поймал тачдаун (по крайней мере, не в этом наборе данных)

Что происходит, когда что-то занимает больше трех секунд?

Если клик на срезе или иная команда выполняется слишком много времени, происходят три вещи:

1. Ход мыслей пользователей во время ожидания нарушается. Они отвлекаются, и часто переключаются на электронную почту, или Интернет. Иногда они забывают вернуться…
2. Если они будут знать, что время отклика системы велико, они вообще могут не нажимать.
3. В конечном счете, ваше реноме как профессионала будет подорвано. Ваша работа будет недооценена и рассматриваться как расходы.

Поэтому важно думать о производительности в не меньшей степени, чем о содержании отчетов. Скорость не менее важна, чем правильные цифры.

Срезы

Возможно, вы удивитесь, узнав, что эти дружелюбные маленькие помощники обычно являются самыми дорогими частями вашего отчета.

Операция считается «дорогой», если она потребляет много времени. Например, проверка <column> = <static value> – недорогая для двигателя DAX, а <column> = <measure> – потенциально дорогая.

Вы, вероятно, видели **перекрестную фильтрацию**, но не задумывались об этом. Пример данных НФЛ (американский футбол), которые мы иногда используем в блоге приведен на рис. 19.1. Выберите два диапазона в верхнем срезе – на 320 и 340 фунтов:

Рис. 19.2. Обратите внимание, что срез CollegeAttended (посещаемый колледж) теперь имеет только два доступных значения; все остальные отключены

Почему срез CollegeAttended осуществил автофильтрацию, хотя мы не сделали на нем никакого выбора? Срез показывает, что нажатие любого другого колледжа приведет к пустой сводной таблице. Если щелчок по какой-то «плитке» в срезе приведет к пустой сводной таблице, то эта плитка будет отключена. Это то, что мы называем перекрестной фильтрацией, и это поведение среза, которое по умолчанию включено для всех срезов. Как правило, перекрестная фильтрация – очень полезная функция. Но… она является дорогостоящей с точки зрения производительности

Более того, перекрестная фильтрация нагружает ядро PowerPivot, если мы выводим в сводную таблицу какую-нибудь меру (рис. 19.3).

Рис. 19.3. Добавлена мера – [Total Catches], и теперь еще два колледжа являются активными

Таким образом, срезы не просто чувствительны друг к другу, они также чувствительны к мерам, выводимым в сводную таблицу. Это означает, что меры должны быть оценены для каждого из колледжей в срезе (даже если мы не нажали ни одного), чтобы увидеть, будет ли любая мера возвращать значение для каждой плитки! Чтобы включить или отключить плитки в срезе, перекрестная фильтрация фактически повторно запускает весь обсчет сводной таблицы («за

кулисами»), как если бы плитки в срезе были в Строках (или Столбцах). Те строки сводной, которые «за кулисами» вернули хотя бы одно непустое значение, являются плитками, которые будут отображаться как кликабельные.

Этот процесс «за кулисами» повторяется для каждого среза, подключенного к вашей сводной таблице, каждый раз, когда потребитель отчета что-то нажимает. Итак, каждый срез, который вы добавляете, так же дорог, как и добавление новой сводной таблицы. Другими словами, одна сводная с пятью срезами будет работать приблизительно с той же скоростью, как и шесть сводных.

Смягчение последствий перекрестной фильтрации

Есть несколько возможностей:

1. Ничего не делать. Если вы все еще укладываетесь в 3 сек., то вы можете не напрягаться.
2. Использовать меньшее количество срезов.
3. Отключить перекрестную фильтрацию для некоторых срезов. И здесь есть два аспекта: **как** и **когда** это делать.

Чтобы отключить перекрестную фильтрацию, выберите срез. На ленте появится вкладка *Параметры* (рис. 19.4). Кликните кнопку *Настройка среза*.

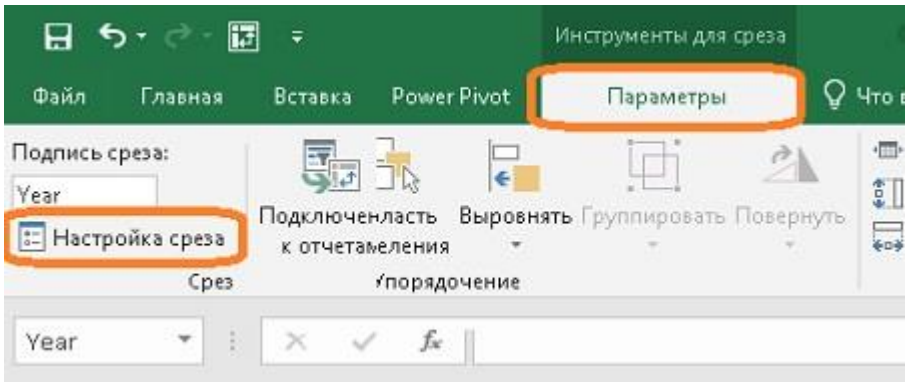

Рис. 19.4. Вкладка *Параметры*

В открывшемся окне *Настройка среза* снимите выделенный флажок:

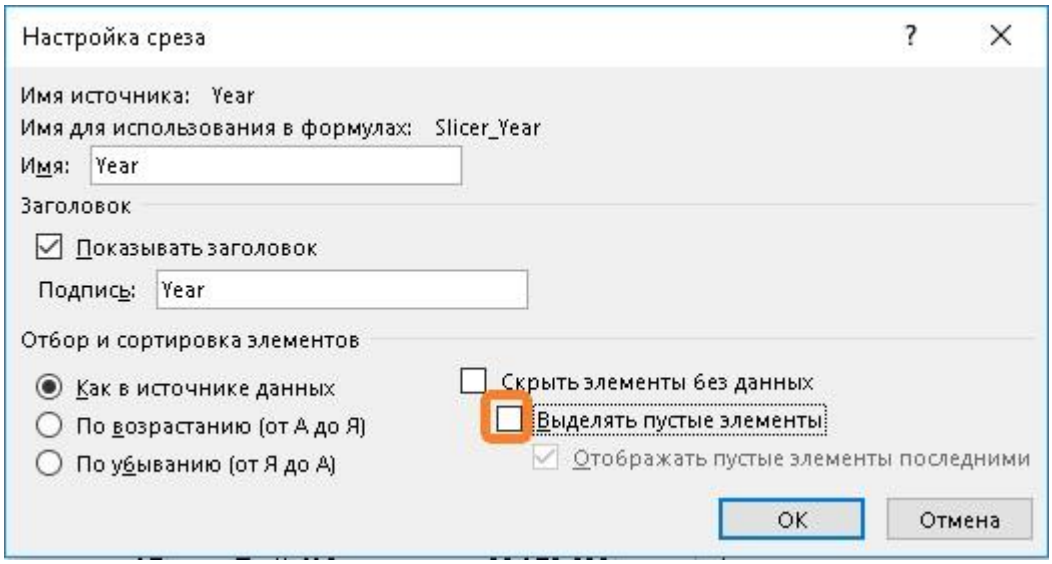

Рис. 19.5. Окно *Настройка среза*

Обратите внимание: отключение перекрестной фильтрации влияет только на **этот** срез. Т.е., фильтры в других срезах никак не изменят подсветку плиток в этом срезе. В то же время фильтрация в нашем срезе по-прежнему влияет на подсветку плиток в других срезах. Другими словами, отключение опции перекрестной фильтрации, изменяет входящую фильтрацию среза, но изменяет исходящую из него фильтрацию.

Срезы, для которых следует отключить перекрестную фильтрацию

Есть три вида срезов для которых следует отключить перекрестную фильтрацию:

1. Срезы, для которых все плитки почти всегда имеют данные.
2. Срезы с небольшим количеством плиток. Функция перекрестной фильтрации особенно полезна для предотвращения прокрутки среза. А если плиток мало, то и прокручивать нечего.
3. Срезы, которые находятся на вершине иерархии. Например, если у вас есть три среза – для страны, штата и города – лучше, чтобы срезы штатов и городов сохраняли перекрестную фильтрацию (по причине длинной прокрутки), а вот срез для страны можно отключить. Кроме того, самый верхний срез в иерархии, как правило, имеет мало плиток.

Форма исходных таблиц также важна

Более узкие таблице лучше. Удалить столбцы, которые вы не собираетесь использовать. Переместите как можно больше столбцов из таблиц данных в таблицы поиска… даже если это означает создание новой таблицы поиска.

Например, в таблице продаж (рис. 19.6) нам не нужны все три столбца (город/штат/ZIP). Достаточно указать ZIP, так что мы можем перенести город и штат в другую таблицу (рис. 19.7). Затем мы свяжем таблицы *Sales* и *Location* (рис. 19.8).

StoreID	Date	TotalSales	City	State	ZIP
174	4/1/201...	33376	Duvall	WA	98019
187	4/1/201...	89909	Kirkland	WA	98033
205	4/1/201...	44317	Kirkland	WA	98033
276	4/1/201...	74610	Bellevue	WA	98005
302	4/1/201...	53480	Redmo...	WA	98052
309	4/1/201...	29123	Redmo...	WA	98052
323	4/1/201...	91802	Redmo...	WA	98052

Рис. 19.6. Исходная таблица *Sales* в модели данных

StoreID	Date	TotalSales	ZIP
174	4/1/201...	33376	98019
187	4/1/201...	89909	98033
205	4/1/201...	44317	98033
276	4/1/201...	74610	98005
302	4/1/201...	53480	98052

City	State	ZIP
Duvall	WA	98019
Kirkland	WA	98033
Kirkland	WA	98034
Bellevue	WA	98005
Bellevue	WA	98004
Redmo...	WA	98052

Рис. 19.7. Сокращенная таблица *Sales* и новая таблица поиска *Location*

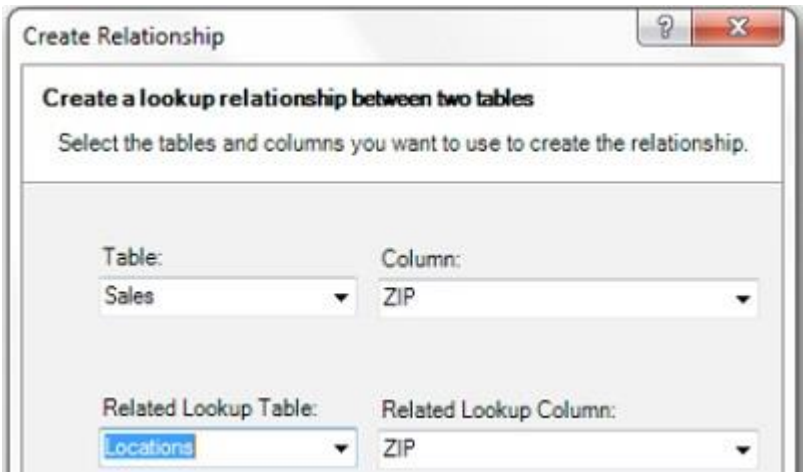

Рис. 19.8. Связь между таблицами *Sales* и *Location* по столбцу ZIP

Иногда может даже стоить «повернуть» исходные данные перед импортом (столбцы в строки). Power Query поддерживает преобразование UNPIVOT, которое поможет превратить широкую и короткую таблицу в высокую и узкую. Иногда это может иметь большое значение, а в других случаях не влияет на производительность, так что поэкспериментируйте.

Ниже пример широкой таблицы продаж: 9 столбцов и около 1М строк (рис. 19.9).

	Date	TotalSales	Normal Sales	Promo Sales	Returns	Rebates	Margin	Discounts
174	4/1/201...	33376	43204	24086	37931	34375	28185	26023
187	4/1/201...	89909	42600	21669	22323	44419	23394	26730
205	4/1/201...	44317	22952	35902	40079	28999	34696	24511
377	4/1/201...	70203	28269	25455	43945	25594	25531	20542
400	4/1/201...	34442	28631	20924	21041	44641	34339	39198
407	4/1/201...	10768	49734	44222	46381	25651	48936	28179
467	4/1/201...	59412	28251	33842	44064	40492	45455	32095
542	4/1/201...	69295	36118	48611	40257	37022	42440	29791

Рис. 19.9. На каждую трансакцию приходится 7 числовых столбцов

Вот та же таблица после операции UNPIVOT:

StoreID	Date	AmtType	Amt
174	4/1/201...	1	33376
187	4/1/201...	1	89909
205	4/1/201...	1	44317
377	4/1/201...	1	70203
400	4/1/201...	1	34442
407	4/1/201...	1	10768
467	4/1/201...	1	59412
542	4/1/201...	1	69295
651	4/1/201...	1	79463
787	4/1/201...	1	18329
893	4/1/201...	1	27108
174	4/1/201...	2	43204
187	4/1/201...	2	42600
205	4/1/201...	2	22952
377	4/1/201...	2	28269
400	4/1/201...	2	28631
407	4/1/201...	2	49734
467	4/1/201...	2	28251

Рис. 19.10. Теперь в таблице только 4 столбца, правда, 7М строк

Теперь мера [Total Sales] будет вычисляться не по формуле SUM(Sales[TotalSales]), а по формуле CALCULATE(SUM(Sales[Amt]);AmtType=1).

Импортированные данные, как правило, лучше, чем вычисляемые столбцы

Если вы можете реализовать вычисляемый столбец в исходном источнике данных, а затем импортировать этот столбец, а не вычислять его в Power Pivot, это может повысить производительность. Импортированные столбцы сжимаются более эффективно, что приводит к меньшим размерам файлов, меньшему потреблению оперативной памяти и, как правило, лучшей производительности срезов. Вот еще несколько замечаний:

- Чем больше строк в таблице, тем более эффективным это может быть (мы не беспокоимся об этом в небольших таблицах поиска).

- Если вычисляемый столбец используется в сводной таблице в областях *Строки / Столбцы / Фильтры*, это вызывает меньше проблем, чем его использование в области *Значения*.
- Если вычисляемый столбец является основой для связи, использование импорта еще более эффективно. Таким образом, если почти все столбцы в таблицах данных и поиска импортированы, но вы создали один вычисляемый столбец в таблице данных, чтобы связать его с таблицей поиска, вы, вероятно, платите б***о***льшую часть штрафа вычисляемого столбца, несмотря на ваши усилия в другом месте.

Одноуровневые таблицы поиска лучше многоуровневых

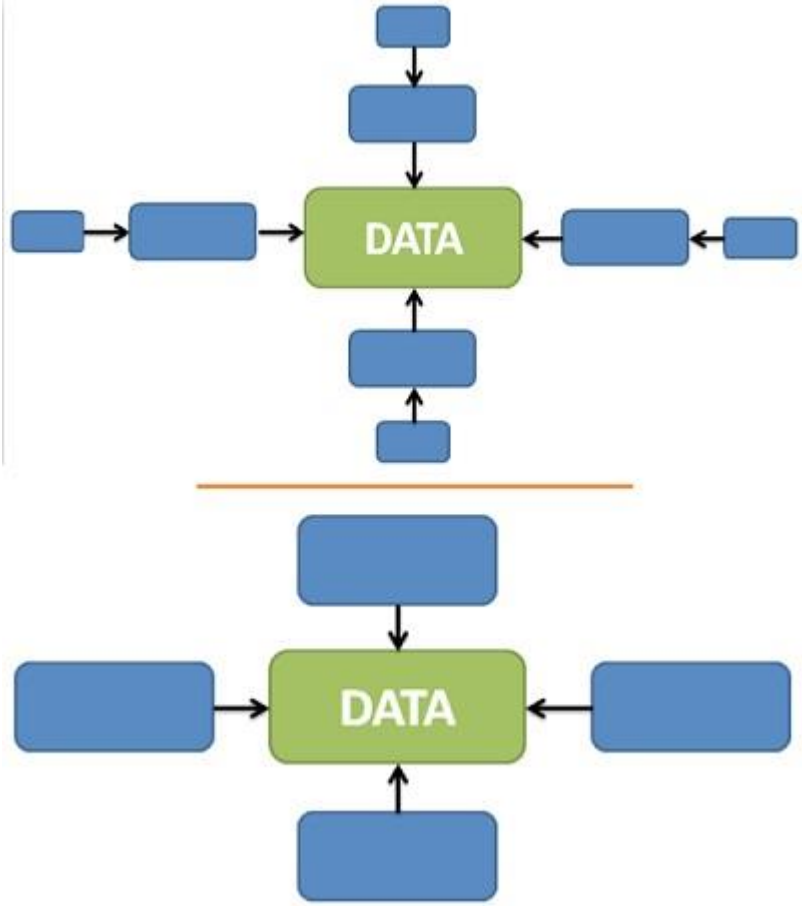

Рис. 19.11. Многоуровневая и одноуровневая схемы таблиц поиска

Если вы можете цепочку таблиц поиска объединить в единую, большую таблицу, это часто может повысить производительность (хотя выше мы ратовали за более узкие таблицы, это относилось к таблицам данных; с таблицами поиска это не так важно).

Еще несколько советов.

DISTINCTCOUNT() намного быстрее, чем COUNTROWS(DISTINCT())

FILTER() следует использовать для таблиц поиска и других «малых» столбцов. И вот почему. Аргумент <filter> функции CALCULATE() имеет возможность проверять «блоки» строк сразу, чтобы увидеть, должны ли эти строки быть активными в соответствии с контекстом фильтра, например, таким как Products[Color]="Blue". Именно это делает <filter> настолько быстрым для оценки даже сотен миллионов строк данных.

Функция FILTER() не имеет этой возможности «проверки блоков» и всегда проходит через строки в аргументе <table> по одному за раз. <filter> функции CALCULATE(), который сканирует 100 миллионов строк, может просмотреть 1000 различных блоков, чтобы решить, какая из 100 миллионов строк должна быть активной. Но если вы используете FILTER() на тех же 100 миллионов строк, ему может потребоваться 100 миллионов проверок, а не 1000, что означает, что он может быть в 100 000 раз медленнее! На практике DAX использует различные методы оптимизации и кэширования, поэтому ваши результаты будут отличаться в зависимости от модели данных и сложности меры.

Помните, что функции «X» являются циклами. Например, если у вас есть мера SUMX(), то она смотрит на каждую строку в аргументе <table> по одному, как и FILTER(). Иногда легко забыть, сколько работы SUMX() и подобные функции делают за кулисами. Особенно легко забыть, сколько работы происходит, если столбец или таблица из аргумента <table> не отображается в сводной таблице.

По нашему опыту, одна функция X редко является проблемой. Но когда вы начинаете делать поиски внутри других поисков, и чтобы вычислить значение в одной ячейке нужно выполнить 1000 циклов, в каждый из которых вложено еще 1000 циклов, то 1M вычислений может существенно снизить производительность.