Многомерная модель данных

На этом уроке

О многомерной модели данных как интуитивном представлении данных для аналитических задач.

Теория

1. Что такое многомерная модель данных? Как она помогает решать задачи BI?
2. Основные понятия многомерной модели данных: (гипер-)кубы, показатели (метрики, параметры), измерения.
3. Язык запросов MDX как многомерно-ориентированная замена SQL.

Многомерная модель данных

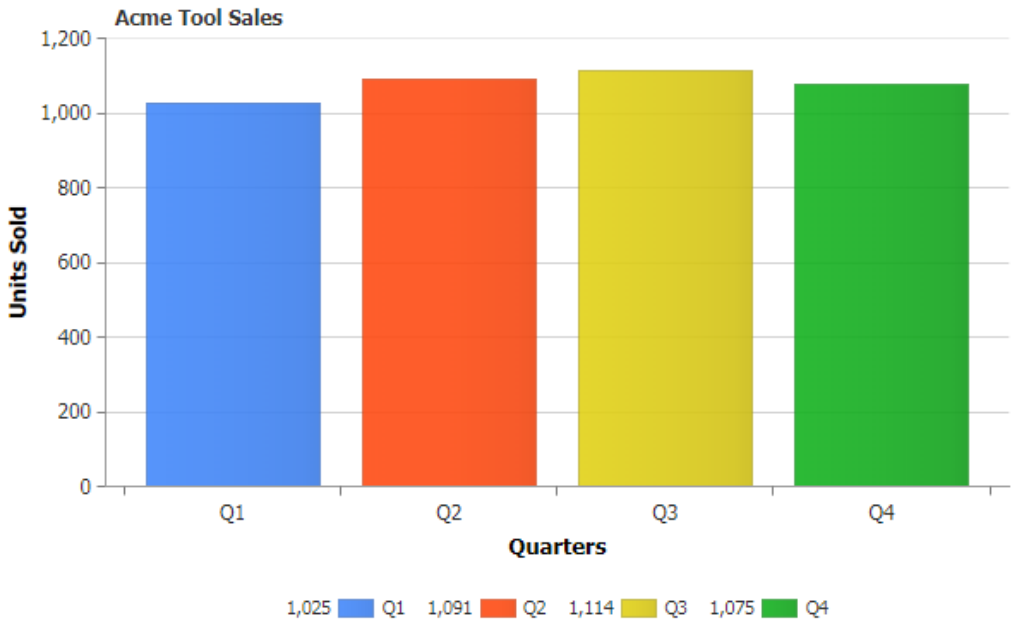
Многомерная модель данных — это структура, позволяющая анализировать данные на различных уровнях их обобщения, как во всем множестве значений этих данных, так и в отдельных их подмножествах. Существует, как табличная реализация этого подхода, так и реализация хранения в виде многомерной матрицы данных. Ключевым моментом при таком построении является функция агрегации, которая должна выполняться на всех уровнях такого обобщения.

Чтобы, понять как строится такая модель, рассмотрим построение такой модели, на примере анализа выручки в платежной системе. То есть наша мотивация — это анализ выручки.

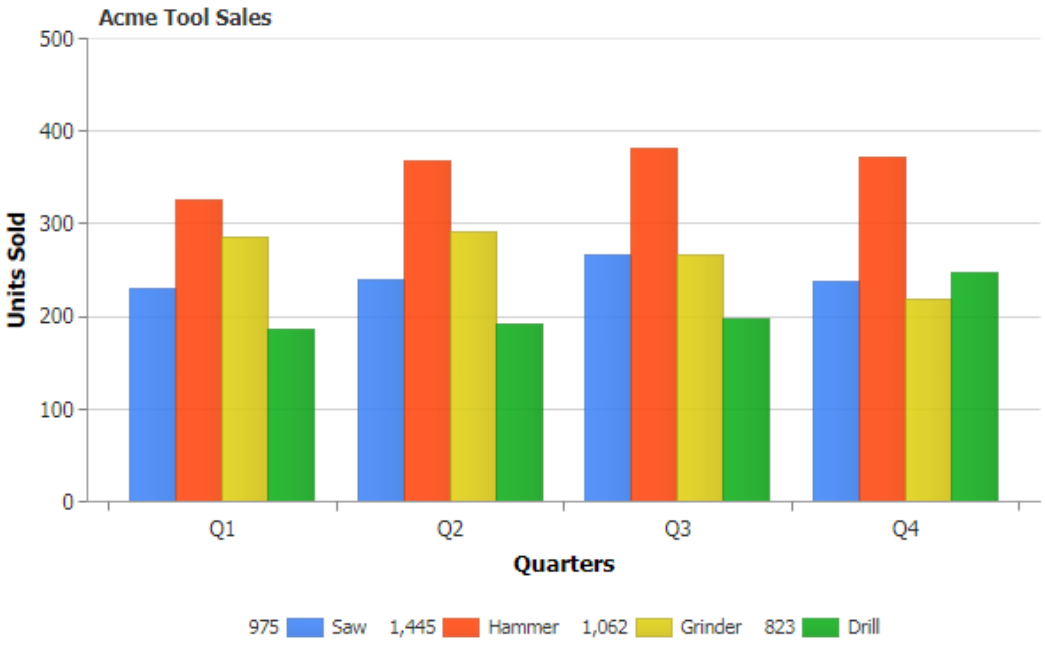
Мотивация

Итак, у нас есть БД с транзакциями и нам требуется вычислить сумму одного из полей в такой базе данных (например, сумму величин транзакций). На языке SQL примерно так: `SELECT SUM(value) FROM transactions`. Из запроса видно, что функция агрегации у нас СУММА.

Этот запрос возвращает единственное число, что не дает нам достаточной детализации. Предположим, мы захотим построить разбивку по кварталам, и это приведет к выполнению следующего запроса: `SELECT QUARTER(dt), SUM(value) FROM transactions GROUP BY QUARTER(dt)`. Получим уже график:



Если потребуется разбить каждый столбец по категории, потребуется выполнить запрос с большим числом группировок, причем с соединенной таблицей: `SELECT QUARTER(dt), category, SUM(value) FROM transactions JOIN categories USING category_id GROUP BY QUARTER(dt), category`.
Визуализация также станет детальнее:



Как видно из примеров выше, **каждое уточнение визуализации приводит к новому запросу к данным**. Если каждый запрос выполнять по не подготовленным данным, то на достаточно больших датасетах **ожидание отрисовки значительно ухудшит производительность**.

Так как детальные данные (запись о каждой отдельной транзакции) в визуализации мы не используем, требуемые **агрегаты можно предпосчитать**, превратив миллионы записей в тысячи — отзывчивость таблицы значительно меньшего объема очевидно станет выше.

Таким образом мы получили первое представление об устройстве многомерной модели. Точнее её реализации в реляционной базе данных. Такие решения относят к ROLAP решениям. Подробно они рассматриваются в следующем уроке курса, на этом же занятии описывается модель данных, положенная в их основу.

Измерения и метрики

В терминах многомерной модели данных квартал и категория из примеров выше — это *измерения* (англ. dimensions), а `SUM(value)` — это *метрика* (также называется показателем, параметром или мерой; «метрика» и «мера» ближе к англоязычным вариантам — metric, measure). Далее поясняется происхождение терминов.

Таблица `SELECT QUARTER(dt), SUM(value) FROM transactions GROUP BY QUARTER(dt)` отражает сумму транзакций для каждого квартала. По сути по результату этого запроса можно ответить на вопрос: «каким был объем транзакций в третьем квартале рассматриваемого года»?

Таким образом логически мы «подаем на вход» квартал, а «на выходе» получаем значение объема транзакцией. По сути этот **объем является функцией от квартала**: `SUM(value) = f(QUARTER(dt))`. Важно, что имеется в виду именно логический запрос суммы за конкретный квартал: технически запрос к базе данных возвращает таблицу с данными за все кварталы, которые и выводятся в визуализации.

В функции выше всего один аргумент. Она одномерная: такой график можно изобразить на двумерной плоскости. Ровно это и сделано на столбчатой диаграмме выше: горизонтальная ось отражает кварталы (аргумент), а вертикальная — значение объемов транзакций в соответствующем квартале (функцию, зависящую от аргумента).

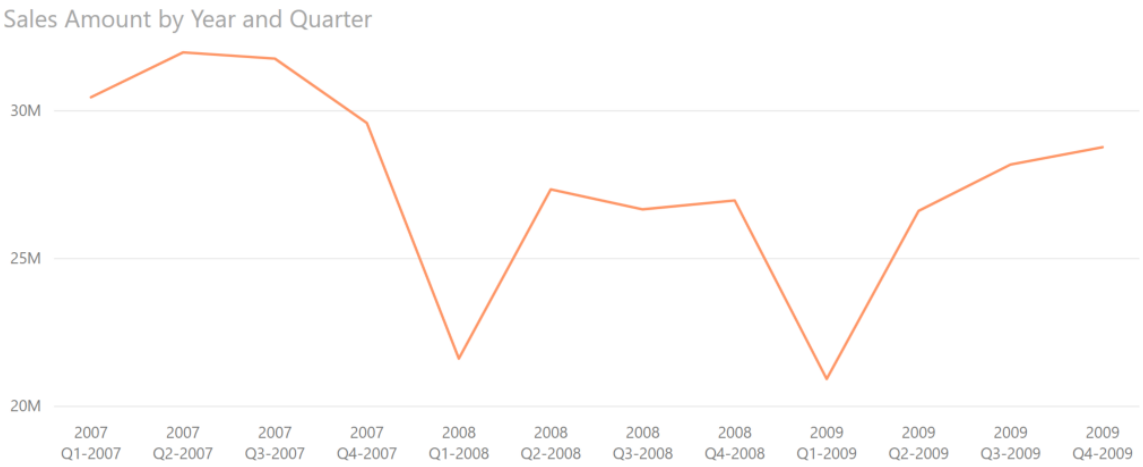
Если мы обратимся к следующему примеру — разбивке объема транзакций дополнительно по категории, то получим, что объем зависит уже от двух аргументов: квартала и категории. Таким образом, **при расширении ключа группировки размерность функции возрастает**. Теперь мы имеем двумерную функцию: `SUM(value) = f(QUARTER(dt), category)`. Двумерная функция раскладывается по двум (независимым) *измерениям*.

С математической точки зрения объем не вычисляется фактически от квартала, то есть эта «функция» не была предопределена заранее, она лишь строится на основе данных. Математизация здесь введена лишь с целью провести параллели для объяснения термина «измерение».

Гиперкуб

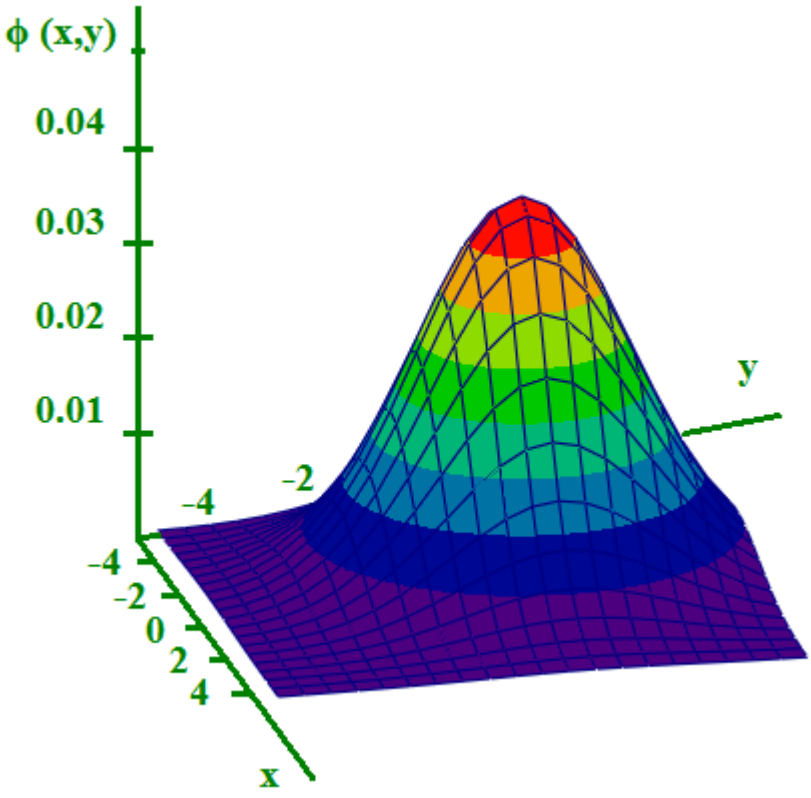
Примеры агрегатов выше можно также изобразить «наивно», то есть без применения хитростей визуализации с группировкой столбцов.

Одномерная функция изображается привычным нам образом:

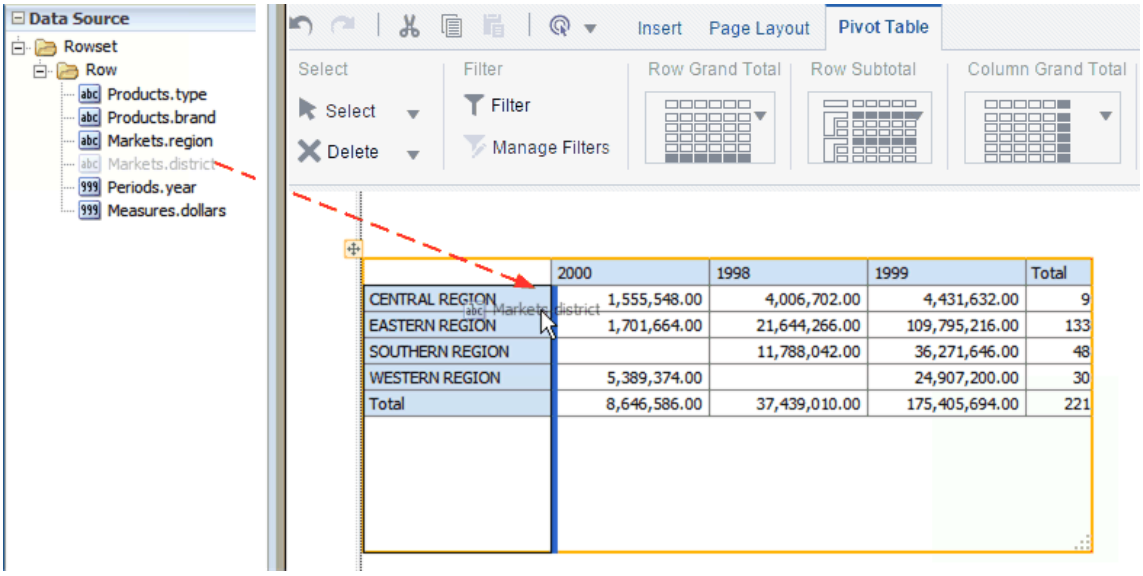


Она имеет форму ломаной, потому что число значений аргумента дискретно.

Двумерную функцию мы изобразили бы уже в трех измерениях: оси OX и OZ отводятся под измерения квартала и категории, а OY под значение функции. В случае непрерывных функций такая визуализация оправдана:

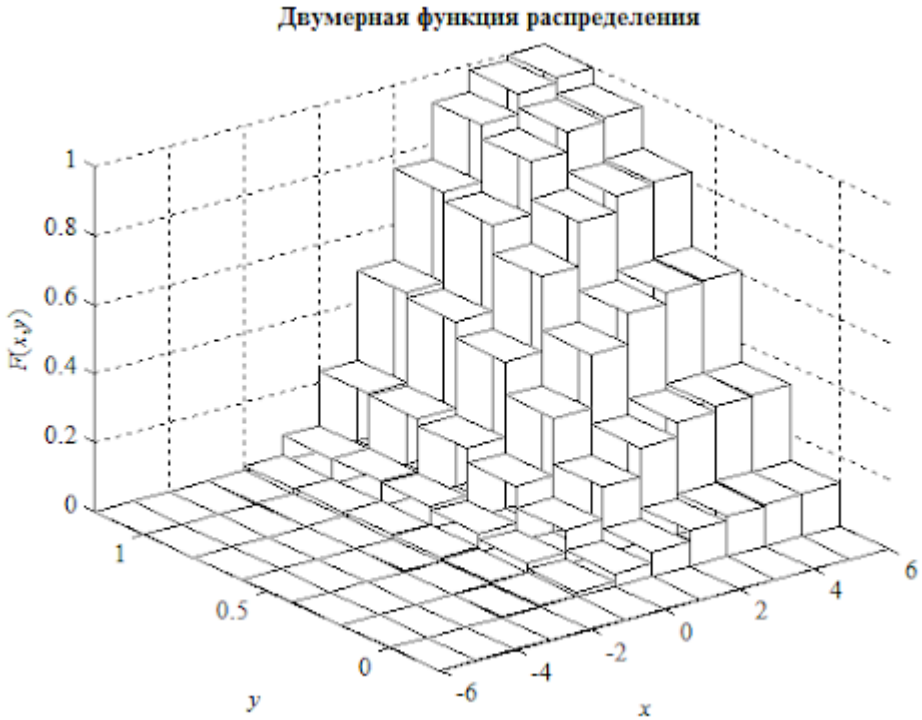


Но в случае, если множества значений аргументов конечны (разнообразие категорий невелико, а кварталов всего 4 в году), ее же можно изобразить плоской в виде таблицы:



Данные на скриншоте выше построены по другому датасету, но общая идея остается: одна из «осей» таблицы — это года, вторая — это регион, а значениями ячеек являются значения метрики. Год и регион — это измерения. От трехмерной визуализации эта отличается лишь тем, что значение записано в ячейке явным образом, а не представлено высотой соответствующего столбца.

Пример трехмерной визуализации двумерной функции с конечным набором значений аргументов:



Если же мы добавим еще одно измерение, то изобразить такие данные графически уже не получится — нам потребуется четыре измерения (три на аргументы и четвертое — на функцию). А вот в виде куба такие данные

представимы:

Каждый из «заголовков» куба — это значения измерений. Они разложены по трем осям. А значения показателя находятся в соответствующих клетках куба. Три измерения определяют «координату» значения «в теле» куба — то есть три аргумента определяют значение функции.

Если мы добавим четвертое измерение в рассмотрение, то даже такое представление не будет удобно для визуализации набора данных. Однако кубы с ≥4 измерениями существуют и называются *гиперкубами*.

В общем случае куб с любым числом измерений называется гиперкубом соответствующей размерности (в том числе нулевой, одинарной и т.д.). В разговорной речи приставку «гипер» опускают, поэтому вы чаще будете слышать термин «OLAP-куб».

Кстати, рассмотренное выше плоское представление называется сводной таблицей (англ. pivot table) и может быть обобщено на любое количество измерений. Однако с ростом числа измерений удобство использование такой таблицы снижается.

А куб нулевой размерности — это рассмотренный в самом начале результат запроса `SELECT SUM(value) FROM transactions`. В нем нет никаких измерений, только метрика. Это не бессмысленный запрос, иногда в дашборде требуется изобразить виджет с общим числом заказов, общей выручкой или т.п. А детализация, например, по дате приводится на соседних графиках.

Факты и измерения

Описанные выше концепции **связаны с понятиями таблиц-фактов и таблиц-измерений** из части курса про архитектура хранилищ данных.

Показателями в кубе выступают агрегированные значения полей из таблиц-фактов, то есть факт это не сама метрика, а значение, которое подается на вход функции агрегации (не SUM(value), а value), пример таких значений:

- величина транзакции;
- количество транзакций (число строк или уникальных идентификаторов транзакций из таблицы-факта);
- число уникальных пользователей, совершивших транзакцию;
- средняя величина транзакции;
- время, затраченное на исполнение транзакции — например, квантиль от этой величины.

Измерениями в кубе выступают данные из таблиц-измерений, например:

- категория транзакции;
- регион пользователя, из таблицы Пользователей;
- страна пользователя, из таблицы География (сопоставляющая страну региону).

Отдельную таблицу-измерение для даты могут не строить, так как сама по себе дата уникальна и может выступать ключом, а извлечение года, месяца и квартала из даты можно сделать "на лету". Хотя, при больших объёмах, чтобы максимально увеличить производительность, избежать проблем с часовыми поясами или добавить свои группировки, для даты строят таблицу-измерение, где хранят в отдельных колонках год, месяц, квартал, день недели и тому подобное.

В домашнем задании вам потребуется соединить историю транзакций (таблицу-факт) с таблицей кодов MCC.

Сравнение многомерной модели и нормализованной базы данных

Выше упомянута одна из причин хранения данных для BI в многомерной модели данных вместо чтения из «боевой» базы, которая обычно находится в ≥3 нормальной форме: **построение агрегатов на базе многомерной модели эффективнее и повышает отзывчивость BI инструмента**.

Вторая причина в интуитивности: **многомерная модель проще для восприятия** человеку с недостаточным опытом работы с базами данных. Многомерная модель более бизнес-ориентированная, чем нормализованная база данных.

Пример из реальной жизни — схема БД в WordPress, самой распространенной системы управления контентом. WP служит основой для многих интернет-блогов. В таких блогах публикуются посты, которые рубрикуются и тегируются.

Если таблица фактов представляет лог просмотров постов пользователями на сайте, то в качестве измерения хотелось бы присоединять готовую таблицу соответствия идентификатора поста и рубрики и/или тегов. Чтобы присоединить таблицу в «сырой» схеме БД WordPress, потребуется сначала присоединить таблицу постов, к ней — таблицу таксономии, а уже к ней — таблицу с названиями тегов и рубрик. Воспроизвести такую цепочку соединений аналитику будет не слишком удобно.

Язык запросов MDX

Чтобы восстановить связь с практической реальностью, рассмотрим два языка запросов: распространенный SQL общего назначения и специальный язык MDX — Multidimensional Expressions. Рассмотрим их в контексте построения сводной таблицы по двум измерениям и одной метрике.

Построить одномерную сводную таблицу с заголовочным столбцом можно стандартными средствами: `SELECT QUARTER(dt), SUM(value) FROM transactions GROUP BY QUARTER(dt)` . Получим следующее:

QUARTER(dt)	SUM(value)
Q1	100,500.00
Q2	200,600.00
Q3	300,700.00
Q4	400,800.00

Для построения двумерной сводной таблицы как результата SQL запроса потребуется явным образом перечислить все значения одного из измерений:

```
SELECT
    category,
    SUM_IF(value, QUARTER(dt) = Q1) AS Q1,
    SUM_IF(value, QUARTER(dt) = Q2) AS Q2,
    SUM_IF(value, QUARTER(dt) = Q3) AS Q3,
```

```
    SUM_IF(value, QUARTER(dt) = Q4) AS Q4
FROM transactions
JOIN categories USING category_id
GROUP BY category
```

Получим следующий результат:

category	Q1	Q2	Q3	Q4
Saw	10,100.00	20,200.00	...	...
Hammer	20,100.00	10,100.00	...	...
Grinder	30,100.00	40,100.00	...	...
Drill	40,200.00	40,200.00	...	...

В случае MS SQL можно так же воспользоваться SQL функциями CUBE или PIVOT, но это усложнит запрос. Но даже в таком виде запрос создает сложности, в случае, когда набор значений измерения слишком велик или неизвестен.

В некоторых случаях, BI-инструменты для отрисовки сводной таблицы выполняют простой запрос `SELECT QUARTER(dt), category, SUM(value) FROM transactions JOIN categories USING category_id GROUP BY QUARTER(dt), category` и затем уже самостоятельно правильным образом ориентируют данные.

Альтернативой SQL в этой задаче выступает язык запросов MDX. Впервые представленный компанией Microsoft в 1997 году, его можно использовать в инструментах Microsoft Power BI и SQL Server Analysis Services, которые, в свою очередь, довольно популярны, а также проприетарных BI-инструментах SAP BusinessObjects, IBM Cognos, Pentaho и open source OLAP сервере Pentaho Mondrian. При его использовании BI-инструментам не надо тратить свои ресурсы на правильную ориентацию данных, так как данные сразу придут с сервера в нужном виде. Язык MDX создавался для работы с MOLAP системами, в которых данные хранятся не в виде связанных таблиц, а виде многомерных представлений - OLAP-кубов. Отсюда в классических реляционных БД он не поддерживается, но может иметь место в различных OLAP решениях, построенных на базе таких БД.

Преимущество MDX в ориентированности на многомерную модель данных: в MDX на синтаксическом уровне поддержаны такие понятия, как измерения, метрики, и иерархии измерений.

Пример запроса на MDX:

```
SELECT
    { [Measures].[Sales Amount],
        [Measures].[Tax Amount] } ON COLUMNS,
    { [Date].[Fiscal].[Fiscal Year].&[2002],
        [Date].[Fiscal].[Fiscal Year].&[2003] } ON ROWS
FROM [Adventure Works]
WHERE ( [Sales Territory].[Southwest] )
```

В этом примере запрос содержит следующие сведения о результирующем наборе:

- Предложение `SELECT` задает оси запроса как элементы `Sales Amount` и `Tax Amount` в измерении `Measures` и как элементы 2002 и 2003 в измерении `Date`.
- Предложение `FROM` указывает, что источником данных является куб `Adventure Works`.
- Предложение `WHERE` определяет ось среза как элемент `Southwest` измерения `Sales Territory`.

Домашнее задание

Скачать таблицу кодов MCC. Попробуйте найти ее самостоятельно. Это необязательно: в конце курса перед финальным домашним заданием вам будет предоставлен файл с расшифровками MCC кодов.

Источники

1. Understanding Bar Charts and Column Charts
2. About Pivot Tables
3. Введение в OLAP и многомерные базы данных
4. MDX, Википедия
5. Материализованная интеграция данных и организация хранилищ больших данных
6. MDX, запрос многомерных выражений