

Использование сегментов в Хранилище данных Microsoft SQL Server 2000

Джой Манди (Joy Mundy), перевод Intersoft Lab
Материал был опубликован на корпоративном сайте компании [Intersoft Lab](#)

Содержание

[Обзор](#)

[Использование сегментов в реляционном Хранилище данных SQL Server 2000](#)

[Использование сегментов в SQL Server 2000 Analysis Services](#)

[Выводы](#)

[Для дополнительной информации](#)

[Приложение: Примеры кода на VBScript для клонирования сегмента](#)

Обзор

Эта статья рассуждает о роли сегментации данных в Хранилище. И реляционное Хранилище данных, и кубы Analysis Services поддерживают сегментацию данных. Логический принцип, положенный в основу сегментации, аналогичен принятому в обеих машинах Microsoft® SQL Server™: горизонтальная сегментация данных с помощью некоторого ключа, например - даты. В реляционной базе данных сегментация сопровождается созданием отдельных физических таблиц - например, одной таблицы для данных каждого месяца - и определением общего представления (view) таблиц элементов. Таким же образом Analysis Services в SQL Server Enterprise Edition поддерживает явную сегментацию кубов. Как в реляционной базе данных, так и в машине OLAP сложность физического хранения скрыта от пользователя-аналитика.

Преимущества сегментации Хранилища данных таковы:

- Существенно уменьшается время запроса.
- Улучшается время загрузки и обслуживаемость баз данных.
- Разрешается проблема отсечения данных, связанная с удалением старых данных из активной базы данных.

Этот метод требует построения несколько более сложного приложения для деления данных, чем несегментированная система. Эта статья описывает лучшие примеры проектирования реализации и обслуживания горизонтально сегментированных Хранилищ данных.

Сегментация настоятельно рекомендуется в случае больших систем Analysis Services, поскольку эффективный план сегментации существенно улучшает работу запросов. Сегментацию реляционного Хранилища данных обычно производить не рекомендуется, хотя это может быть эффективным и хорошо работающим решением для ряда специфических проблем обслуживания хранилища.

Использование сегментации в реляционном Хранилище данных SQL Server 2000

Сегментированное представление соединяется с горизонтально сегментированными данными из набора членов, благодаря чему данные представлены так, как будто они взяты из одной таблицы. SQL Server 2000 различает локальные и распределенные сегментированные представления. В локальном сегментированном представлении все представления и таблицы-участники располагаются на одном и том же физическом SQL Server'e. В распределенном сегментированном представлении по меньшей мере одна из таблиц-участников расположена на другом, удаленном сервере. Распределенные сегментированные представления не рекомендуется применять для приложений Хранилищ данных.

Хранилища данных, имеющие измерения, структурированы на основании фактов и измерений и обычно физически представлены в виде схем типа "звезда", "снежинка" и очень редко - как полностью денормализованные плоские таблицы, объединяющие и факты и измерения. Эта статья затрагивает использование сегментации в схеме, имеющей измерения, так как такие схемы являются наиболее распространенными среди структур реляционных Хранилищ данных. Приведенные рекомендации применимы и к более общим схемам Хранилищ.

Преимущества сегментации

Отсечение данных

Многие администраторы Хранилищ данных предпочитают по прошествии некоторого времени архивировать старые данные. Например, нарезанное (clickstream) Хранилище данных может содержать подробные данные в онлайн-режиме только за три-четыре месяца. Другие распространенные технологические решения хранят информацию за 13, 37 месяцев или 10 лет в режиме онлайн, архивируя и удаляя из баз данных старые данные, как только они прокручиваются за пределы активного окна. Такая структура с прокручиваемым окном является обычным делом в случае больших хранилищ данных. Без сегментированных таблиц, процесс удаления старых данных из базы данных требует очень большого оператора DELETE, например:

```
DELETE FROM fact_table  
WHERE date_key < 19990101
```

Этот оператор неэффективен с точки зрения затрат на его выполнение и, вероятно, на это уйдет больше времени, чем на процесс загрузки в ту же таблицу. В случае же сегментированных таблиц наоборот администратор переопределяет представление UNION ALL для исключения самой старой таблицы и удаляет эту таблицу из базы данных (вероятно, после обеспечения ее резервного копирования) - что выполняется практически мгновенно.

Как будет сказано ниже, обслуживание сегментированных таблиц требует больших затрат. Если отсечение данных является единственным основанием для введения сегментации, проектировщику необходимо исследовать метод вырезания данных для удаления старых данных из несегментированной таблицы. Скрипт, удаляющий 1000 строк за один раз (используя команду "set rowcount 1000") может выполняться непрерывно с низким приоритетом, пока не будут удалены все подлежащие удалению данные. Этот подход эффективно используется на больших системах и представляет собой более прямой метод, чем построение системы управления сегментацией. В зависимости от объемов загрузки и использования системы этот метод может оказаться подходящим для ряда систем, и может применяться к конкретной системе после некоторой предварительной оценки.

Скорость загрузки

Самым быстрым способом загрузки данных является загрузка их в пустую таблицу или таблицу, не имеющую индексов. Инкрементный процесс загрузки может быть гораздо более эффективным, если загрузка будет производиться в более мелкие сегментированные таблицы.

Простота обслуживания

Как только созданы промежуточные приложения Хранилища данных для поддержки сегментации, упрощается все обслуживание системы в целом. Обслуживание, в том числе и загрузка данных, и резервное копирование, и восстановление таблиц, может выполняться параллельно, необычайно повышая производительность. Можно ускорить и упростить процесс инкрементного наполнения кубов, растущих по направлению потока данных (downstream).

Скорость запросов

Скорость запросов не должна рассматриваться в качестве причины сегментации реляционной базы данных Хранилища. Выполнение запроса происходит одинаково для сегментированных и несегментированных таблиц фактов. Если сегментированная база данных спроектирована правильно, в план запроса реляционной машины войдут только те сегменты, которые необходимы для выполнения этого запроса. Например, если база данных сегментирована по месяцу и запросу ограничен Январем 2000 года, план запроса будет содержать только сегмент для Января 2000 года. Итоговый запрос будет выполняться над сегментированной таблицей так же, как и над индексированной объединенной таблицей с индексом кластера, соответствующим ключу сегментации.

Недостатки сегментации

Сложность

Самым большим недостатком сегментации является необходимость построения администратором приложения для управления сегментами. Начало разработки Хранилища данных, использующего горизонтальную сегментацию с реляционной базой данных, может осложниться потребностью в предварительной разработке, тестировании и выпуске такого приложения. Одна из задач этой статьи состоит в обсуждении вопросов и проектировочных решений, положенных в основу приложения для управления сегментами.

Ограничения проектирования запросов

Для оптимального выполнения запросов все они должны размещать условия ключа фильтрации непосредственно в таблице фактов. Запрос, накладывающий ограничение на вторую таблицу, например, на таблицу измерения Дата, включит в запрос все сегменты.

Проблемы проектирования

Хранилище данных, имеющее измерения, структурированы по фактам и измерениям, и обычно физически изображаются в виде схем типа "звезда", "снежинка", и очень редко - как полностью денормализованные плоские таблицы, объединяющие и факты и измерения. Администратор такого хранилища обычно сегментирует только таблицы фактов. При этом сегментация таблицы измерений дает ряд преимуществ. При некоторых обстоятельствах очень большая таблица измерений, содержащая более миллионов 10 элементов (members) может также выиграть от применения сегментации. Не имеющее измерений реляционное Хранилище данных также может быть сегментировано с применением общих замечаний, данных в этой статье.

Эффективный план сегментации разрабатывается в контексте архитектуры системы и целей проектирования. Даже при идентичных дизайнах схем, реляционное Хранилище данных, предназначенное только для наполнения кубов Analysis Services, может предполагать структуры сегментации, отличные от напрямую запрошенных аналитиками. Система с прокручивающимся окном нуждается в обязательной сегментации по времени, другие системы этого не требуют.

Если Хранилище данных содержит кубы Analysis Services, Microsoft рекомендует параллельно структурировать сегменты реляционного Хранилища и баз данных Analysis Services. Обслуживающее приложение упрощается: приложение создает новый сегмент куба в тот же момент, что и новую таблицу в реляционной базе данных. Администраторам необходимо понять только стратегию сегментации. Тем не менее, приложение может иметь существенные основания для сегментации двух баз данных различным образом, и единственным недостатком этого была бы сложность обслуживающего приложения.

Обзор схем сегментации

Сегментированные таблицы в базе данных SQL Server могут использовать обновляемые (updatable) или запрашиваемые (необновляемые) (queryable, nonupdatable) сегментированные представления. В обоих случаях сегменты таблицы создаются с помощью проверки CHECK, проверяющего правильность данных, содержащихся в каждом сегменте. Обновляемое сегментированное представление будет поддерживать выполнение INSERT (или UPDATE, или DELETE) над представлением и направлять операцию в нужную базовую таблицу. Несмотря на то, что это дает большое преимущество, приложение Хранилища данных, как правило, нуждается в ускорении загрузки, а это невозможно с использованием представлений. Таблица, приведенная ниже, обобщает требования, преимущества и недостатки обновляемых и запрашиваемых сегментированных представлений.

Требования	Преимущества	Недостатки
Обновляемое сегментированное представление		
<ul style="list-style-type: none"> • Ключ(и) сегментации, обусловленные ограничением (ями) CHECK. • Ключ(и) сегментации, являющиеся частью первичного ключа. • Ключ(и) сегментации, не являющиеся частью какого-либо ограничения базы данных. 	<ul style="list-style-type: none"> • <i>Эффективность выполнения запроса:</i> план запроса включает только те таблицы элементов (member tables), которые необходимы для выполнения данного запроса. • <i>Простота обслуживающего приложения:</i> Данные могут загружаться в представление UNION ALL и вставляются в соответствующую 	<ul style="list-style-type: none"> • <i>Медленность выполнения загрузки:</i> загрузка данных через представление оказывается слишком медленной для того, чтобы этот метод имел право на жизнь относительно большинства приложений Хранилищ данных. • <i>Негибкость:</i> схема базы данных может требовать

<ul style="list-style-type: none"> • Представление UNION ALL, определенное над таблицами элементов (member tables). 	таблицу элементов (member table)	дополнительных ограничений на ключ сегментации.
Запрашиваемое сегментированное представление		
<ul style="list-style-type: none"> • Ключ(и) сегментации, обусловленные ограничением (ями) CHECK. • Представление UNION ALL, определенные над таблицами элементов (member tables). 	<ul style="list-style-type: none"> • <i>Эффективность выполнения запроса:</i> план запроса включает только те таблицы элементов (member tables), которые необходимы для выполнения запроса. • <i>Улучшение выполнения загрузки:</i> увеличение загрузки напрямую таблицы элементов (member tables) с повышением производительности. • <i>Эффективность хранения:</i> для сегментированного представления не требуется индекс первичного ключа, хотя обычно и рекомендуется объявлять первичный ключ и создавать по нему. 	<ul style="list-style-type: none"> • Представление ограничено 256 таблицами элементов (member tables). • Для управления сегментами и загрузкой должно быть создано обслуживающее приложение.

Microsoft рекомендует проектировать таблицу фактов как локальное (на отдельном сервере) сегментированное объединенное представление с определенным первичным ключом. В большинстве случаев это определение также даст в результате сегментированное обновляемое представление, однако обслуживающее приложение Хранилища данных должно быть спроектировано для увеличения загрузки большинства данных не через представление, а напрямую в таблицы элементов (member tables).

Синтаксис шаблона

Приведенный ниже пример кода иллюстрирует синтаксис для определения таблиц элементов (member tables) и объединенного представления и для вставки данных в представление:

```
-- Создаем таблицу фактов для 1999
CREATE TABLE [dbo].[sales_fact_19990101] (
[date_key] [int] NOT NULL
CHECK ([date_key] BETWEEN 19990101 AND 19991231),
```

```
[product_key] [int] NOT NULL ,
[customer_key] [int] NOT NULL ,
[promotion_key] [int] NOT NULL ,
[store_key] [int] NOT NULL ,
[store_sales] [money] NULL ,
[store_cost] [money] NULL ,
[unit_sales] [float] NULL
)
ALTER TABLE [sales_fact_19990101]
ADD PRIMARY KEY (
[date_key], [product_key], [customer_key], [promotion_key], [store_key])
;
-- Создаем таблицу фактов для 2000
CREATE TABLE [dbo].[sales_fact_20000101] (
[date_key] [int] NOT NULL
CHECK ([date_key] BETWEEN 20000101 AND 20001231),
[product_key] [int] NOT NULL ,
[customer_key] [int] NOT NULL ,
[promotion_key] [int] NOT NULL ,
[store_key] [int] NOT NULL ,
[store_sales] [money] NULL ,
[store_cost] [money] NULL ,
[unit_sales] [float] NULL
)
ALTER TABLE [sales_fact_20000101]
ADD PRIMARY KEY (
[date_key], [product_key], [customer_key], [promotion_key], [store_key])
;

--Создаем представление UNION ALL.
CREATE VIEW [dbo].[sales_fact]
AS
SELECT * FROM [dbo].[sales_fact_19990101]
UNION ALL
SELECT * FROM [dbo].[sales_fact_20000101]

--Теперь вставляем несколько строк данных, например:
INSERT INTO [sales_fact]
VALUES (19990125, 347, 8901, 0, 13, 5.3100, 1.8585, 3.0)

INSERT INTO [sales_fact]
VALUES (19990324, 576, 7203, 0, 13, 2.1000, 0.9450, 3.0)

INSERT INTO [sales_fact]
VALUES (19990604, 139, 7203, 0, 13, 5.3700, 2.2017, 3.0)

INSERT INTO [sales_fact]
VALUES (20000914, 396, 8814, 0, 13, 6.4800, 2.0736, 2.0)

INSERT INTO [sales_fact]
VALUES (20001113, 260, 8269, 0, 13, 5.5200, 2.4840, 3.0)
```

Для проверки правильности работы сегментации, используем Query Analyzer, показывающий план запроса для такого запроса, как:

```
SELECT TOP 2 * FROM sales_fact WHERE date_key = 19990324
```

В этот план запроса следует включать только таблицу 1999. Сравните этот план запроса с тем, что сгенерирован теми же двумя таблицами с удаленным первичным ключом: таблица 2000 остается исключенной. Сравните различия между этими планами с планом запроса, сгенерированным для схемы с удаленным ограничением по date_key. С удаленным ограничением обе таблицы - и 1999 и 2000 включаются в запрос.

Заметьте, что, в целом, полезно использовать синтаксис "TOP N" при выполнении исследовательских запросов к большим таблицам, поскольку он возвращает результаты быстро и с минимальной нагрузкой ресурсов сервера. При рассмотрении планов запросов к сегментированным таблицам это становится еще более важным, так как план запроса, сгенерированный оператором "SELECT *", труден для проверки синтаксиса. Для случайного наблюдателя это выглядит так, будто план запроса включает в себя все таблицы компонент представления UNION ALL, хотя во время выполнения запроса в нем используются только соответствующие ему таблицы.

Применение условий напрямую к таблице фактов

Для наилучшего выполнения запросов все они должны помещать условия на ключ фильтрации напрямую в таблицу фактов. Запрос, налагающий ограничение на вторую таблицу, например, на таблицу измерения Дата, включит в запрос все сегменты. Стандартные объединенные запросы типа "звезда" в таблице фактов UNION ALL работают достаточно хорошо:

- Создайте стандартным образом выражение WHERE для запроса типа "звезда", налагая условия на атрибуты любой несегментированной таблицы измерений.
- Включите атрибуты из сегментирующего измерения (Дата).
- Спроектируйте запросы к сегментированной имеющей измерения схеме в точности как вы это сделали бы для несегментированной схемы, только с учетом того, что условия по датам наиболее эффективны при применении напрямую к ключу дат в таблице фактов.

Если каждая сегментированная таблица имеет кластеризованный индекс с датой в качестве первого столбца этого индекса, затраты на обращение ко всем сегментам для выполнения незапланированного запроса относительно невелики. Предопределенные запросы, как те, что генерируют стандартные отчеты или инкрементно обновляют лежащие на их пути базы данных, должны быть как можно более эффективными.

Выбор ключа(ей) сегментации

Данная таблица фактов может быть сегментирована по множеству измерений, но профессионалы чаще всего сегментируют только дату. Как описано выше, сегментирование по дате дает возможность просто управлять "прокручивающимся окном" и более старые сегменты даже могут располагаться в другом месте или быть проиндексированы менее жестко, чем последние по времени сегменты. Кроме того, большинство запросов в Хранилище данных имеют фильтр по дате.

Для приложений, сегментированных по дате, переменные решения таковы:

- **Сколько данных хранить в онлайнном режиме?** Это решение должно приниматься главным образом в соответствии с требованиями бизнеса, скорректированными соотношением типа "цена/качество" применительно к хранению очень больших объемов данных в режиме онлайн.
- **Как должна выглядеть дата?** Широко распространенным способом представления даты в Хранилищах данных является использование суррогатных ключей для таблиц измерений и фактов. Для таблиц фактов, сегментированных по дате, рекомендуется использовать "интеллектуальные" целочисленные суррогатные ключи в форме уuuuymmdd. Будучи целым числом, такой ключ использует только 4 байта, тогда как ключ datetime требует 8 байтов. Естественные ключи по дате такого типа (datetime) используются во многих Хранилищах данных.

- **Насколько мелкими должны быть сегменты?** Хотя вышеприведенный пример использует сегменты по годам, большинство систем выбирают более детальную сегментацию - по месяцам, неделям или дням. И хотя достаточно интересно было бы выяснить, насколько успешно проходят пользовательские запросы по границам месяцев или недель, гораздо более важным является общий размер и управляемость системы. Вспомним, что любой SQL-запрос может ссылаться не более чем на 256 таблиц. Для Хранилищ данных, содержащих данные более чем за несколько месяцев, представление UNION ALL в ежедневном режиме превысит допустимый предел. По приблизительным подсчетам фактическая таблица, сегментированную только по дате, следует сегментировать скорее всего по неделям.
- **Как определяется области сегментации?** Синтаксис BETWEEN имеет наиболее непосредственный характер, наиболее естественен для понимания и достаточную эффективность выполнения. Например, рассмотрим сегменты по месяцу в следующей форме:
 - o `date_key < 19990101`
 - o `date_key BETWEEN 1990101 AND 19990131`
 - o `date_key BETWEEN 19990201 AND 19990229`
 - o ...
 - o `date_key BETWEEN 19991201 AND 19991231`
 - o `date_key > 19991231`Обратите внимание на первый и последний сегменты в вышеприведенной записи: даже если вы не собираетесь помещать в них данные, тем не менее, будет полезно определить эти сегменты, чтобы охватить весь спектр возможных значений данных. Кроме того, заметьте, что сегмент Февраль (February) охватывает данные до 29 Февраля включительно, несмотря на то, что 1999 год не является високосным. Эта структура исключает необходимость в присоединении логики високосных лет к схеме приложения, создающего сегменты и ограничения.
- **Объединяются ли сегменты с течением времени?** Чтобы минимизировать число активных сегментов администратор базы данных может принять решение разработать сегментирующее приложение таким образом, чтобы ежедневные сегменты сливались в недели или месяцы. Такой подход подробно обсуждается далее, в разделе, посвященном наполнению и обслуживанию сегментов.

Это подробное обсуждение процесса сегментации по дате должно помочь в рассмотрении других предполагаемых ключей сегментации.

Загрузка данных: Если существует однозначная тенденция выравнивания входных данных по другому измерению, или, например, если данные для каждого Магазина или Дочерней компании обеспечиваются различными системами, тогда они являются естественными ключами сегментации.

Запрос данных из кубов: Несмотря на то, что не существует никакого технологического основания для одинаковой сегментации реляционной базы данных и кубов Analysis Services, обычно их сегментируют одинаково. При этом упрощается обслуживающее приложение. Таким образом, даже если реляционная база данных существует исключительно для наполнения кубов Analysis Services, при выборе ключей сегментации следует рассматривать общие модели запросов.

Соглашения о наименованиях

Соглашения о наименовании таблиц элементов (member tables) горизонтально сегментированной таблицы фактов должны следовать из схемы сегментации. Для наибольшей универсальности в заголовке можно использовать полную начальную дату сегмента:

[sales_fact_yuuummdd] предпочтительнее, чем [sales_fact_yuuu], даже если сегментирование ежегодное.

Если база данных поддерживает несколько уровней детализации сегментов, соглашения о наименованиях должны отражать промежуток времени, расположенный внутри каждого сегмента. Например, для ежемесячной сегментации используют sales_fact_20001101m, а для ежедневной - sales_fact_20001101d. Имена таблиц элементов (member tables) скрыты от конечных пользователей, работающих с данными через представление, поэтому имена таблиц элементов (member table) должны быть сориентированы на обслуживающее приложение.

Сегментация для расположенных по ходу кубов

Если единственное применение реляционного Хранилища данных состоит в поддержке кубов Analysis Services, нет необходимости определять представление UNION ALL. В этом случае предел в 256 таблиц к приложению не применяется, однако не рекомендуется сегментировать реляционное Хранилище данных таким образом, чтобы исключалась возможность определения представления UNION ALL.

Управление сегментированной таблицей фактов

Сегментированное Хранилище данных не должно реализовываться до завершения автоматизации и тестирования управления сегментами. Система управления сегментами представляет собой простое приложение, общие требования к которому излагаются в этой статье.

Далее считаем, что сегментация будет происходить по дате.

Метаданные

Мощная система управления сегментацией основана на метаданных. Эти метаданные могут храниться где угодно, поскольку доступ к ним формируется программным образом. Большинство систем Хранилищ данных использует пользовательские (custom) таблицы метаданных, определенные на сервере Хранилища данных - SQL Server, или Microsoft SQL Server Meta Data Services.

Вне зависимости от особенностей механизма хранения метаданных, содержимое их должно включать следующую информацию по каждому сегменту:

- Имя сегмента.
- Дату создания сегмента.
- Интервал дат данных в сегменте.
- Сегмент дат, представляемый в онлайн (включенный в представление UNION ALL).
- Сегмент дат, представляемый в автономном режиме (удаленный из представления).
- Удаленный сегмент дат.

Другие таблицы метаданных, входящих в общую систему управления Хранилищем данных, должны отслеживать, когда и сколько данных загружается в каждый сегмент.

Создание новых сегментов

Первой задачей системы управления сегментацией является создание новых сегментов. Эти действия должны выполняться периодически, для создания новой таблицы, которая станет следующим сегментом.

Существует множество эффективных способов выполнения этой задачи. Рекомендуемый подход состоит в использовании SQL-DMO (Distributed Management Objects) для создания новой таблицы со структурой и индексами, аналогичными существующим сегментам, но с новым табличным именем, наименованиями индексов, определениями ограничений ключей сегментации, файловых групп и т.д.:

- Получите шаблон определения таблицы (обычно это самый последний по времени сегмент);
- Модифицируйте свойства таблицы и индекса Name, проверьте свойства ограничения Text и другие;
- Используйте метод ADD для обработки таблицы.

При использовании интеллектуальных соглашений об именах задача решается несколькими сроками кода. Как будет сказано далее, приложение может использовать сегменты Analysis Services для кубов системы Хранилища данных. В этом случае скрипт или программа, создающая сегментированные таблицы в PCYБД RDBMS, может продолжать создавать соответствующие сегменты кубов, используя Decision Support Objects (DSO).

Наполнение сегментов

Как это описано выше, данные могут загружаться в представление UNION ALL. В теории это является замечательным свойством структуры сегментации таблиц, однако на практике применительно к приложениям Хранилищ данных это делать не рекомендуется. Загрузка данных в представление UNION ALL не может выполняться большими объемами; процесс загрузки в этом случае будет слишком медленным для Хранилища данных, имеющего достаточно большие размеры, необходимые для сегментированных таблиц.

Вместо этого приложение, осуществляющее загрузку в Хранилище данных, должно проектироваться под быструю загрузку данных каждого периода в соответствующую целевую таблицу. Если приложение сбора данных (data staging application) реализовано в SQL Server Data Transformation Services (DTS), Dynamic Properties Task может легко изменить bvz целевой таблицы для Data Pump Task или Bulk Insert Task.

Поскольку новое представление еще не участвует в представлении UNION ALL, загрузка данных не требует простоя системы.

Промежуточное приложение Хранилища данных должно иметь возможность работать с входящими данными, не принадлежащими текущему сегменту. Это случай может возникнуть как исключение среди нормальных событий, если процесс загрузки Хранилища данных не завершился за ночь. Другие системы начинают постоянно сталкиваться со вновь появившимися старыми данными. И схема системы должна учитывать вероятность, частоту и объемы данных таких исключений.

Если старые данные возникают в достаточно небольшом объеме, простейшая схема для загрузки всех данных, не принадлежащих текущему сегменту, воспользуется обновляемым представлением UNION ALL.

Определение представления UNION ALL

Как только инкрементная загрузка успешно закончена, необходимо исправить представление UNION ALL. И снова рекомендацией будет использовать для этого SQL-DMO: воспользуйтесь методом ALTER для изменения свойств TEXT объекта VIEW. Список сегментов для включения в определение представления наилучшим образом можно получить из приведенной выше таблицы метаданных.

Слияние сегментов

На первый взгляд, слияние нескольких сегментов в один большой сегмент выглядит нерациональным. Однако Хранилище данных с большим объемом ежедневной загрузки и маленьким временем, в течение которого необходимо загрузить данные, может существенно выиграть от этого с помощью:

- Создания текстового файла с загружаемыми данными, отсортированными в порядке кластеризованного индекса.
- Загрузки данных большими фрагментами в пустые ежедневные сегменты.
- Создания всех некластеризованных индексов.
- Выведения в онлайн нового сегмента путем создания заново представления UNION ALL.
- Еженедельно создавая и заполняя новые еженедельные сегменты, вставляя в них данные из ежедневных сегментов, перестраивая индексы и пересоздавая представление UNION ALL. После этого ежедневные сегменты могут быть удалены.
- Перемещая данные в еженедельные или даже ежемесячные сегменты по мере устаревания данных и обеспечивая таким образом возможность хранения большего числа сегментов в режиме онлайн в рамках представления UNION ALL.

Использование сегментов в SQL Server 2000 Analysis Services

Analysis Services в SQL Server Enterprise Edition явно поддерживает сегментированные кубы аналогичные сегментированным таблицам в реляционной базе данных. Для куба среднего или большого размера сегменты могут существенно улучшить выполнение запросов, ход загрузки и облегчить обслуживание кубов. Сегменты могут быть спроектированы по одному или более измерениям, однако кубы часто сегментируются только по измерению Дата. Инкрементная загрузка сегментированного куба, включая и создание новых сегментов, должна осуществляться специально разработанным приложением.

Примечание: Сегменты могут храниться локально или могут быть распределены по множеству физических серверов. Несмотря на то, что очень большие системы могут также выигрывать от распределения сегментов по различным серверам, наши тесты показали, что решения с распределением сегментов обеспечивает наибольший выигрыш в случаях, когда кубы имеют размер порядка многих терабайт. Эта статья рассматривает только локально сегментированные кубы. Инкрементная загрузка сегментированного куба, включая и создание новых сегментов, должна выполняться специально разработанным приложением.

Преимущества сегментации

Выполнение запросов

Сегментация куба существенно улучшает выполнение запросов. Даже кубы среднего размера,

основанные на всего лишь 100-гигабайтных объемах данных из реляционной базы, выиграют от сегментации.

Преимущества сегментации куба особенно заметны при многопользовательской загрузке.

Улучшение выполнения запросов для каждого приложения, отличается в зависимости от структуры куба, использования шаблонов и схемы сегментирования. Запрос, требующий данные только за один месяц из куба, сегментированного по месяцу, получит доступ только к одному сегменту. В целом, ожидается, что переход от большого куба в единственном сегменте к хорошо спланированной локальной стратегии сегментирования даст в результате в среднем улучшение выполнения запросов на 100-1000 %.

Отсечение старых данных

Как и в случае с реляционным Хранилищем данных, системный администратор Analysis Services может предпочесть сохранение только последних по времени данных в кубе. Для одного сегмента единственным способом очистки его от старых данных является новая обработка куба. Сегментация по измерению Дата дает администратору возможность удалить старые сегменты без простоя системы.

Обслуживание

С организационной точки зрения сегменты представляют собой независимые единицы данных, которые могут добавляться и уничтожаться, не влияя при этом друг на друга. Это способствует управлению жизненным циклом данных системы. Каждый сегмент куба хранится в отдельном наборе файлов. Управлять операциями резервного копирования (Backup) и восстановления (Restore) этих файлов данных проще при более мелких файлах сегментов. Это особенно справедливо, когда каждый файл сегмента имеет размер менее двух гигабайт. В этом случае будут эффективны утилиты архивации и восстановления. Если часть куба разрушена или в нем выявлены неверные или несогласующиеся данные, этот сегмент может быть обработан заново быстрее, чем целый куб. К тому же, для экономии пространства возможно изменять режим хранения и схему агрегирования более старых сегментов.

Различные сегменты могут использовать различные источники данных. Отдельный куб может объединять данные из множества реляционных баз данных. Например, корпоративные данные могут создаваться таким образом, что данные из Европы или Северной Америки хостировались бы на разных серверах. Если куб сегментирован по географическому признаку, он может логически объединить эти различные источники данных. Чтобы определение отдельного куба проходило нормально, реляционная схема на всех серверах должна быть абсолютно идентична.

Ход загрузки

Сегментированный куб может загружаться гораздо быстрее, чем не сегментированный, так как сегменты могут загружаться параллельно. Для параллельной обработки сегментов вам необходимо приобрести инструмент стороннего поставщика или создать такую программу самостоятельно. На многопроцессорной машине преимущества существенно заметнее. Средство параллельной обработки должно использовать 90 процентов мощности центрального процессора. Такой режим работы достигается обычно за счет одновременной обработки от одного до двух сегментов для каждого двух процессоров. Например, на четырехпроцессорной машине, где все процессоры заняты обработкой куба, можно попробовать обрабатывать от двух до четырех сегментов одновременно. Если попробовать обрабатывать большее число сегментов, чем существующее количество процессоров, производительность резко упадет. Один сегмент для каждого двух процессоров - это

достаточно осторожное решение. Идеальное число сегментов зависит от скорости потока данных из исходных баз данных, схемы агрегации, хранения и других факторов. При некоторых условиях более эффективным способом будет перестройка сегмента, а не его инкрементная обработка. Разумеется, это гораздо менее вероятно, если куб содержится в единственном сегменте.

Недостатки сегментации

Сложность

Основным недостатком сегментации является необходимость в создании администратором специального приложения для обслуживания сегментов. Неразумно начинать реализацию сегментированного куба до завершения проектирования, тестирования и выпуска приложения для обслуживания его сегментов. Одной из задач этой статьи является обсуждение проблем и решений в области проектирования приложений для обслуживания сегментов.

Операции с метаданными

С ростом количества сегментов ухудшается выполнение операций над метаданными (например, броузинг определения куба). Это в большей степени проблема администратора, а не конечного пользователя, однако чрезмерно сегментированный куб администрировать достаточно сложно.

Вопросы проектирования

Обзор сегментов

Эффективный план запроса должен соблюдать баланс различных аспектов:

- **Количество сегментов:** Analysis Services никак не ограничивают число сегментов куба, однако куб с несколькими тысячами сегментов труден для обработки. К тому же, в определенный момент затраты на компоновку результирующего множества из многих сегментов перевешивают преимущества обработки запросов с возможностью сегментной избирательности. Сложно найти способ практически определить такой момент, так как это зависит от схемы куба, шаблонов запроса и аппаратного обеспечения, однако разумно иметь один сегмент для каждого хранимого кубом гигабайта или каждых десяти миллионов строк фактических данных. Другими словами, куб в 10 Гб (т.е. 1 миллиард фактов) на аппаратном обеспечении, соответствующем таким объемам данных, должен с легкостью поддерживать 200 сегментов. Если же схема сегментации требует существенно большего количества сегментов, разумно протестировать альтернативный план сегментации.
- **Загрузка и обслуживание:** Данные могут естественным образом попадать в куб по ряду измерений (таких, как, например, время). Для того, чтобы поддерживать промежуточное приложение для наполнения и инкрементного обновления куба, этими измерениями могут стать естественные сегментационные срезы. Измерение Даты, например, обычно является первым в сегментации. Другие приложения могут получать данные, сегментированные по географическому региону, сегменту потребителей, и т. д. Так как различные сегменты используют различные источники данных, программа наполнения куба может эффективно загружать данные из распределенного Хранилища данных или другой исходной системы.
- **Выполнение запроса:** Эффективная схема сегментации основана на некоторых знаниях об общих свойствах запросов пользователей. Идеальное измерение сегментации

слишком избирательно для большинства детализованных пользовательских запросов. Например, сегментация по Дате часто улучшает выполнение запросов, так как многие запросы относятся к не слишком давним временным периодам. Аналогично, возможно, что многие пользователи обращаются в запросах к географическим или организационным направлениям. Для максимального улучшения работы запросов, можно затрагивать в них как можно меньше сегментов.

Проще управлять сегментами по статичным измерениям или измерениям, которые, подобно Дате, изменяются предсказуемо. Например, сегмент по "Штатам США" относительно статичен, так как проектировщики приложений вправе ожидать множества предупреждений о создании 51-го штата. С другой стороны, сегментация по измерению Продукт с большей вероятностью будет меняться с течением времени, по мере относительно частого появления новых продуктов. Возможно, при этом все равно будет потребность в сегментации по динамическому измерению, однако проектировщик должен иметь ввиду, что администрирующая система обязательно должна быть сложной. Если измерение помечено как "изменяющееся", то сегментация по нему не допускается. В любом случае, разумно создать сегмент "все прочие" для хранения данных по неожиданным элементам измерений.

Срезы и фильтры

Как и в случае реляционных сегментов, сегменты Analysis Services основаны на действиях администратора по определению данных для каждого сегмента. РСУБД использует для этого CHECK CONSTRAINT; Analysis Services использует срез. Срез установлен для отдельного элемента измерения, такого, как, например, [Dates].[1999] или [Dates].[1999].[Q1]. В мастере сегментации Analysis Manager Partition Wizard срез установлен в экране, озаглавленном "Выберите срез данных (необязательно)". В DSO доступ к срезу и его установка производится с использованием свойства SliceValue объекта уровня измерения сегмента. Пример синтаксиса приведен далее.

Определение каждого сегмента включает также информацию об источнике потока данных этого сегмента. Сегмент метаданных хранит информацию, необходимую для наполнения сегмента. Администратор может установить источник данных и таблицу фактов с помощью мастера сегментации Partition Wizard, или программно с помощью DSO. Во время обработки сегмента установки его свойства SliceValue автоматически трансформируются в фильтр для источника. Определение сегмента может содержать дополнительный фильтр, свойство SourceTableFilter, используемое для уточнения запроса, обеспечивающего наполнение сегмента. Во время обработки сегмента оператор запроса WHERE, обращенный к исходным данным, будет содержать установленные по умолчанию условия, основанные на определении среза и любой(ые) дополнительный(ые) фильтр(ы), определенные свойством SourceTableFilter.

Для правильной работы сегментов и срезы и фильтры должны быть определены правильно. Задача Среза состоит в улучшении выполнения запроса. Машина Analysis Services использует информацию сегмента. Определение среза предназначено для направления запроса именно к сегменту(ам), содержащим базовые данные. Запросы будут точно выполняться по сегментированному кубу и без определения срезов сегментов, однако их выполнение будет неоптимальным, так как в отсутствие определения среза каждый запрос должен будет исследовать все сегменты.

Задача фильтра и исходных метаданных состоит в определении данных, предназначенных для сегмента. Эти элементы должны быть правильно определены, иначе весь куб будет содержать неверные данные. Когда идет обработка сегмента, Analysis Services ограничивают данные, хранимые в кубе, согласно требованиям Среза. Но при этом не выполняется никаких проверок относительно того, не загружены ли данные и в другой сегмент.

Например, представьте, что вы сегментировали куб по году и неверно установили Срез для сегмента 1998 как [Dates].[Year].[1997], а фильтр при этом устанавливает ограничение 1998. Сегмент при обработке получит ноль строк: возможно, это желаемый результат. С другой стороны, если у вас существовал сегмент для 1998 и добавился новый для Декабря 1998, будет довольно несложно загрузить Декабрь 1998 дважды, и при этом не получить никакого предупреждения от Analysis Services о том, что такое произошло.

Не трудно поддерживать выравненность срезов сегмента и фильтров, однако проектировщик управляющей системы должен обязательно быть в курсе таких проблем.

Дополнительные срезы и фильтры

Большинство стратегий сегментации идентифицируют уровень измерения для сегмента и помещают данные для каждого элемента этого измерения в его собственный сегмент. Например, "сегмент по году" или "сегмент по штату".

Принято также определять план сегментации, углубляющийся (drill down) в одну из сторон куба. Например, последние по времени данные могут быть сегментированы по дню или неделе, а более старые - по месяцу или году.

В зависимости от области использования и количества данных может понадобиться спроектировать более сложный план сегментации. Например, представьте, что 80 процентов потребителей живут в Калифорнии, 10 процентов в Орегоне и остальные 10 процентов равномерно распределены по остальным штатам. Далее, большинство аналитических исследований сосредоточено на местных потребителях (Калифорния). В этом случае, администратор может выбрать сегмент на уровне округа для Калифорнии, сегмент на уровне штата Орегона и один сегмент для всех остальных штатов.

Срезы будут выглядеть примерно так:

- Округа Калифорнии: [All USA].[CA].[Amador] ... [All USA].[CA].[Yolo]
- Штат Орегон: [All USA].[OR]
- Остальные штаты: [All USA]

Как сказано выше, для правильного наполнения эти сегменты фильтры исходных данных должны быть определены правильно. Заметьте, что запрос, требующий объединения данных по Калифорнии и Орегону, будет обязательно просматривать и сегмент "Остальные штаты". Поскольку для Analysis Services не требуется больших затрат на просмотр этого сегмента, дабы убедиться, что там нет соответствующих данных, выполнение запроса будет проходить эффективнее при условии, что куб сегментирован по штату однотипно с углублением (drilldown) по Калифорнии CA. Логика приложения, необходимая для обслуживания неравномерных сегментов, также сложнее, и в целом не рекомендуется использовать этот метод сегментации. Тем не менее, при соответствующей осторожности в проектировании обслуживаемого приложения и правильной оценке компромиссов выполнения запросов этот метод может помочь решить некоторые специфические проблемы проектирования.

Выравнивающие сегменты

Поскольку первая часть этой статьи рассматривает сегментацию в РСУБД, естественно задаться вопросом, должны ли сегменты Analysis Services выравниваться с реляционными сегментами. Эти две стратегии сегментации не требуют идентичности, однако приложение, управляющее сегментами, проще спроектировать, разработать и понять в том случае, если они аналогичны друг другу. Общепринятая стратегия заключается в идентичной сегментации по дате в обеих системах и возможном срезе по второму или даже третьему измерению куба.

Простейшая стратегия состоит в использовании представления UNION ALL в качестве исходной таблицы фактов для всех сегментов куба. Если сегменты куба выровнены с реляционными сегментами, каждый куб может указывать непосредственно на соответствующий ему реляционный сегмент, обманывая таким образом представление UNION ALL. При такой конфигурации запрос, работающий с кубом и извлекающий данные из реляционной базы данных, будет выполняться быстрее. Компромисс этого улучшения заключается в том, что обслуживающее приложение требует гарантий правильной увязки исходной таблицы с каждым сегментом.

Если реляционная база данных предназначена только для наполнения кубов Analysis Services и не обслуживает никакие другие запросы, системный администратор может не создавать и не оперировать представлением UNION ALL. Для оптимизации отдельных запросов, загружающих данные в куб, к реляционным таблицам могут создаваться индексы. В этом случае реляционная база данных служит скорее промежуточной, подготовительной областью, чем полноценным Хранилищем данных.

Режимы хранения и планы агрегации

Каждый сегмент может иметь собственный план хранения и агрегации. Данные, к которым обращаются реже, могут агрегироваться лишь слегка или храниться как ROLAP или HOLAP вместо MOLAP. Куб, инкрементно загружаемый с течением времени, скорее всего не воспользуется этой функциональной возможностью применительно к измерению времени своих сегментов, так как изменение этих параметров потребовало бы новой обработки сегмента. Затраты на время обработки и сложность системы в большинстве ситуаций вряд ли оправдали бы минимальную экономию пространства.

Сегменты по другим измерениям, наоборот, скорее всего, будут иметь различные планы агрегации. Агрегация для каждого сегмента проектируется мастером оптимизации. Чтобы схема агрегации сохраняла максимально возможную актуальность, системный администратор должен сориентировать мастер оптимизации на последние по времени сегмент и всегда строить схему агрегации для каждого нового набора сегментов на основе самых актуальных по времени сегментов.

Управление сегментированным кубом

При создании управляющей системы для реляционных сегментов разработчик может использовать разнообразные инструменты. Настоятельно рекомендуется использовать SQL-DMO, однако эффективные системы создавались и с помощью хранимых процедур, расширенных хранимых процедур и даже скриптов на языке Perl, проверяющих синтаксис текстовых файлов, содержащих определения таблиц. А программа обслуживания сегментов куба должна наоборот, использовать DSO.

Системным разработчикам, имеющим опыт работы с классическими базами данных, метод использования объектной модели для представления объектов базы данных может показаться странным. Для разработки модулей, использующих DMO и DSO, такие разработчики могут применять знакомый им скриптовый язык, например, Microsoft® Visual Basic® Scripting Edition (VBScript), Microsoft® JScript® или Perl или среду разработки наподобие Visual Basic (VB) или C++. Эти модули могут планироваться из операционной системы, из SQL-Agent или вызываться из пакетов DTS. Необходимость в использовании DSO для построения

управляющей системы не должна рассматриваться как основание для отказа от использования сегментов, даже если разработчик никогда ранее не применял объектные модели. Далее в статье приведен пример на VBScript, иллюстрирующий использование скриптинга для клонирования сегментов.

Если реляционное Хранилище данных использует сегменты, система управления сегментами куба должна разрабатываться как часть системы управления сегментами реляционной базы данных. Система управления сегментами куба должна выполнять следующие функции:

- Создавать новые сегменты по мере необходимости, обычно на основании расписания, соответствующего измерению Дат.
- Загружать данные в сегменты.
- Удалять старые сегменты (дополнительно).
- Выполнять слияние сегментов (дополнительно).

Создание новых сегментов

Одновременно с созданием в реляционной базе данных нового сегмента даты система управления сегментами должна создавать все необходимые сегменты куба, связанные с этой датой. Принято инкрементно обновлять измерения куба до создания новых сегментов, так как новый элемент измерения может добавляться по одному из срезов сегмента. Простейшим случаем является сегментация куба исключительно по дате. Система управления сегментами просто создает один новый сегмент согласно соответствующему расписанию (день, неделя, месяц и т.д.).

Если куб, дополнительно к дате, сегментирован по другому измерению, система управления сегментами будет добавлять за один раз множество сегментов. Например, рассмотрим куб, сегментированный по месяцу и по штату США. Ежемесячно система будет создавать 50 новых сегментов штата. В этом случае, безопасно создавать эти ежемесячные сегменты путем клонирования сегментов последнего перед этим месяца, редактируя необходимые атрибуты, например, срез и имя исходной таблицы, и обновляя описание сегмента в кубе.

Однако, рассмотрим куб, сегментированный по месяцу и брэндю продукта. Бренды продукта гораздо более непостоянная величина, чем штаты или области; имеет смысл ожидать, что в новый бренд добавится к иерархии продуктов в период существования данного куба. Обслуживающее приложение должно обеспечивать создание сегмента для хранения этих данных о новом бренде. Рекомендуется поступать следующим образом:

- Обработать измерения до создания новых сегментов.
- Клонировать существующие измерения для обеспечения преемственности режимов хранения и планов агрегации.
- Найти в обработанном измерении новые элементы, создавая сегмент для любых новых элементов уровня сегмента. Система должна определить режим хранения и план агрегации по умолчанию.

Для обеспечения выравнивания и сохранения точности срезов сегментов и определений фильтров с течением времени система управления сегментами должна быть тщательно разработана. Если сегментируется реляционная база данных, и эти сегменты периодически сливаются, как это было описано выше, система управления сегментами должна обновлять определения сегментов куба для их синхронизации с исходными данными. Сегмент куба не должен подвергаться новой обработке, однако, на случай возникновения потребности в этом в будущем, определение необходимо изменять.

Целостность данных

Задача схемы куба и системы управления сегментами состоит в обеспечении обработки данных в один и только один сегмент. Analysis Services не проверяют инициализацию всех строк таблицы фактов в кубе и не подтверждают загрузку строки в один единственный сегмент. Если ряд фактов нечаянно загружается в два сегмента, Analysis Services будут рассматривать его как два различных факта. Любая агрегация будет удваивать эти данные и запросы будут возвращать неверные результаты.

Обработка сегментов

Обработка сегмента в основе своей сходна с обработкой куба. Естественная единица обработки - это один сегмент. Мастер обработки в Analysis Manager обеспечивает следующие три модели обработки куба или сегмента:

- **Incremental Update** (инкрементное обновление) добавляет новые данные к существующему кубу или сегменту и обновляет и добавляет агрегации, на которые влияют эти новые данные.
- **Refresh Data** (регенерация данных) удаляет все данные и агрегации куба или сегмента и заново формирует данные в кубе или сегменте.
- **Full Process** (полная обработка) полностью заново формирует структуру куба или сегмента и затем обновляет данные и агрегации.

Инкрементная обработка требует от администратора определить условия фильтрации к исходному запросу для идентификации набора новых данных для данного куба. Обычно этот фильтр строится на основе даты (даты события или даты обработки), хранящейся в таблице фактов.

Абсолютно те же функциональные возможности имеются у DTS Cube Processing Task. Большинство систем используют DTS Cube Processing Task для составления расписания обработки куба. Инкрементно обработанные кубы используют задачу Dynamic Properties для изменения исходного фильтра. Эта же функциональная возможность может быть реализована и через специальное программирование в DSO, хотя инкрементное обновление требует несколько больше строк кода, чем нужно для простого обновления данных.

При проектировании системы управления сегментами важно помнить о том, что инкрементная обработка куба или сегмента требует предварительной обработки сегмента в прошлом. Не следует применять инкрементную обработку к необработанному кубу или сегменту.

Куб, сегментированный только по измерению Дат, нуждается в непосредственном управлении загрузкой. Обычно существует единственный сегмент для обновления в каждом цикле загрузки; единственным аспектом решения, с которым требуется определиться - это инкрементно ли обновлять эти данные или путем регенерации. Большинство кубов, имеющих измерение Дата, управляются простым DTS-пакетом.

Куб, сегментированный по множеству измерений, имеет следующие дополнительные проблемы и преимущества:

- **Проблема:** Большое число сегментов подлежащих обработке.
- **Проблема:** Возможное изменение числа сегментов.
- **Преимущество:** Параллельная загрузка сегментов.
- **Преимущество:** Существенно улучшенное выполнение высокоизбирательных запросов.

Большинство приложений, сегментированных по множеству измерений, предполагает

систему обработки куба для параллельной загрузки сегментов. Система параллельной загрузки может выпускать множество синхронных DTS-пакетов, параметры которых были обновлены с помощью задачи Dynamic Properties. При всей своей допустимости эта структура достаточно громоздка и многие системы для обновления сегментов предпочтут использовать вместо нее "родной" DSO-код. Имеется и шаблонный инструмент обработки сегментов в параллельном режиме.

Слияние сегментов

Куб, сегментированный по Дате, сталкивается с ростом числа сегментов с течением времени. Как было сказано выше, теоретически существует точка, начиная с которой выполнение запроса резко ухудшается с ростом числа сегментов. Проведенное нами тестирование, включая разработку куба с более чем 500 сегментами, не достигло этой точки. Системные администраторы, возможно, просто взбунтуются раньше, чем эта точка будет достигнута, так как прочие недостатки большого числа сегментов - например, медленность операций с метаданными - необычайно усложнит обслуживание базы данных.

Analysis Services, с помощью и DSO и Analysis Manager, поддерживает возможность слияния сегментов. Когда сливаются два измерения, данные одного из них включаются во второй сегмент. Оба сегмента должны иметь идентичные режимы хранения и планы агрегации. По завершении слияния первый сегмент удаляется, а второй содержит объединенные данные. Обработка слияния имеет место только относительно данных куба, а обращения к источнику данных при этом нет. Процесс слияния двух сегментов очень выгоден системе.

Если схема системы включает объединенные сегменты, процесс слияния должен происходить программным образом, а не через Analysis Manager. Слияние сегментов происходит непосредственно и, как и другие операции DSO, требует всего нескольких строк кода. Чтобы сегмент мог при необходимости быть заново обработан, система объединения сегментов должна отвечать за подтверждение наличия в окончательном объединенном сегменте точных метаданных для исходного фильтра. Процесс слияния сегментов правильно изменяет определения срезов и по мере возможности объединяет определения Фильтра. Однако процесс слияния не требует, чтобы оба сегмента были наполнены данными из одной и той же таблицы или источника данных, таким образом, имеется возможность объединять два сегмента, которые невозможно наполнить заново.

Другой проблемой является то, что объединенный сегмент, как и все сегменты, не может быть переименован.

Этих проблем можно избежать, воспользовавшись следующими полезными приемами системного проектирования:

- Использование ясных соглашений о наименованиях.
- Точное следование согласованному плану слияния сегментов.
- Внимание к соответствию сегментов куба реляционным сегментам или сохранение реляционного Хранилища данных несегментированным.

Рассмотрим, например, куб Sales (Продажи), сегментирующий дату по неделям. Текущая неделя сегментируется по дням и затем в конце недели объединяется. Сегменты имеют названия типа Sales_yyyymmdd, где дата в имени представляет собой первый день даты сегмента. В Ноябре 2000 будут еженедельные сегменты Sales_20001105, Sales_20001112, Sales_20001119 и Sales_20001126. В течение следующей недели создаются и обрабатываются Sales_20001203, Sales_20001204 и так далее до Sales_20001209. В рамках времени обработки в воскресенье, когда система используется менее интенсивно, можно объединить сегменты от 20001204 до 20001209 в Sales_20001203, формируя и оставляя только этот один недельный

сегмент. С другой стороны, можно эффективно переименовать сегмент, создавая новый пустой сегмент с требуемым именем и объединяя в него другие сегменты.

Исключение старых сегментов

Удаление старых данных в кубе сегментированном по Дате, производится так же просто, как удаление более старых сегментов (набора сегментов). Аналогично другим операциям, описанным выше, этот процесс должен управляться программно, а не в незапланированном режиме через Analysis Manager. Если вы уже проделали большую часть работы, возможно, вы сможете написать программу и протестировать этот модуль за считанные часы.

Выводы

Использование локальных сегментов рекомендуется для средних и больших кубов Analysis Services, содержащих более 100 миллионов строк фактов. Сегментация улучшает выполнение запроса к базе данных Analysis Services. Сегментированные кубы проще обслуживать, особенно если из них удаляются старые данные. Тем не менее, сегментация куба требует специального приложения для управления такими сегментами.

Принципы сегментации в базе данных реляционного Хранилища данных сходны с сегментацией в Analysis Services. Как и в случае Analysis Services, здесь необходимо создать приложение для управления реляционными сегментами. Нет никаких особых аргументов в пользу сегментации в реляционном Хранилище данных. Сегментация помогает решать некоторые проблемы обслуживания, такие, как отсечение старых данных, но происходит это за счет усложнения системы. Выполнение запросов, по сравнению с грамотно проиндексированной отдельной таблицей не улучшается.

И реляционная база данных Analysis Services и реляционная база данных SQL Server поддерживает распределенную сегментацию, при которой сегменты располагаются на различных серверах. Обсуждение распределенных сегментов в Analysis Services отнесено в другую статью. Не рекомендуется использовать распределенные реляционные сегменты для системы Хранилища данных SQL Server 2000, поддерживающей незапланированные запросы.

Сегментированные кубы отличаются улучшенным выполнением запросов при большом числе сегментов. Разработчику большого куба следует рассмотреть сегментацию по нескольким измерениям для максимизации избирательности пользовательских запросов и улучшения хода процесса, вводя таким образом возможность параллельной обработки.

Сегментация настоятельно рекомендуется для больших систем Analysis Services. Сегментация реляционного Хранилища данных в целом не рекомендуется, хотя она может неплохо работать и считаться полезным решением для достижения некоторых специфических задач обслуживания Хранилища.

Дополнительная информация

Онлайновые книги Microsoft SQL Server Books Online содержат больше информации по индексированным представлениям. Дополнительную информацию можно найти на следующих ресурсах:

- На сайте Microsoft SQL Server на <http://www.microsoft.com/sql/>.
- В Центре разработки Microsoft SQL Server Developer Center на <http://msdn.microsoft.com/sqlserver>.

- В SQL Server Magazine на <http://www.sqlmag.com>.
- В конференциях microsoft.public.sqlserver.server и microsoft.public.sqlserver.datawarehouse на news://news.microsoft.com.
- В курсах Microsoft Official Curriculum по SQL Server. Актуальная информация по курсу имеется на <http://www.microsoft.com/trainingandservices>.

Приложение: Пример программы на VBScript для клонирования сегмента

```
'/*****
' File: ClonePart.vbs
'
' Desc: Этот пример создает новый сегмент в кубе FoodMart 2000 Sales
'       на основе последних по времени сегментов куба. Целью скрипта
'       является демонстрация видов DSO-вызовов, использующихся
'       для клонирования сегмента. Получившийся в результате сегмент обрабатываетс.
'       однако не добавляет никаких данных в куб.
'
'       При использовании этого скрипта можно, при желании, удалить получившийся
'       в результате сегмент после запуска скрипта и изучения его результатов.
'
' Parameters: None
' *****/

Call ClonePart

Sub ClonePart()

On Error Resume Next

Dim intDimCounter, intErrNumber
Dim strOlapDB, strCube, strDB, strAnalysisServer, strPartitionNew
Dim dsoServer, dsoDB, dsoCube, dsoPartition, dsoPartitionNew

' Инициализация переменных имени сервера, базы данных и куба.
strAnalysisServer = "localhost"
strOlapDB = "FoodMart 2000"
strCube = "Sales"

' VBScript не поддерживает прямое использование нумерованных констант.
' Однако, для вытеснения нумерации константы могут определяться.
Const stateFailed = 2
Const olapEditionUnlimited = 0

' Соединение с сервером Analysis.
Set dsoServer = CreateObject("DSO.Server")
dsoServer.Connect strAnalysisServer

' Если соединения не произошло, закончить скрипт.
If dsoServer.State = stateFailed Then
MsgBox "Error-Not able to connect to " & strAnalysisServer _
& "' Analysis server.", , "ClonePart.vbs"
Err.Clear
Exit Sub
End if

' Некоторые свойства управления сегментами имеются только
' в релизах Enterprise Edition и Developer Edition releases
' Analysis Services.
If dsoServer.Edition <> olapEditionUnlimited Then
MsgBox "Error-This feature requires Enterprise or " & _
"Developer Edition of SQL Server to " & _
```

```
"manage partitions.", , "ClonePart.vbs"
Exit Sub
End If

' Убеждаемся в существовании необходимого источника данных в базе данных.
Set dsoDB = dsoServer.mdStores(strOlapDB)
If dsoDB.Datasources.Count = 0 Then
MsgBox "Error-No data sources found in '" & _
strOlapDB & "' database.", , "ClonePart.vbs"
Err.Clear
Exit Sub
End If

' Поиск куба.
If (dsoDB.mdStores.Find(strCube)) = 0 then
MsgBox "Error-Cube '" & strCube & "' is missing.", , _
"ClonePart.vbs"
Err.Clear
Exit Sub
End If

' Установка переменной dsoCube к требуемому кубу.
Set dsoCube = dsoDB.MDStores(strCube)

' Поиск сегмента
If dsoCube.mdStores.Count = 0 Then
MsgBox "Error-No partitions exist for cube '" & strCube & _
"'.", , "ClonePart.vbs"
Err.Clear
Exit Sub
End If

' Установка переменной dsoPartition к требуемому сегменту.
Set dsoPartition = dsoCube.MDStores(dsoCube.MDStores.Count)
MsgBox "New partition will be based on existing partition: " & _
& chr(13) & chr(10) & _
dsoDB.Name & "." & dsoCube.Name & "." & _
dsoPartition.Name, , "ClonePart.vbs"

' Получение символов цитирования из источника данных, так как
' различные базы данных используют различные символы цитирования.
Dim sLQuote, sRQuote
sLQuote = dsoPartition.DataSources(1).OpenQuoteChar
sRQuote = dsoPartition.DataSources(1).CloseQuoteChar

'*****
' Создание нового сегмента на основе требуемого.
'*****

' Создание нового временного сегмента.
strPartitionNew = "NewPartition" & dsoCube.MDStores.Count
Set dsoPartitionNew = dsoCube.MDStores.AddNew("~temp")

' Клонирование свойств из требуемого сегмента
' в новый.
dsoPartition.Clone dsoPartitionNew

' Изменение имени сегмента с "~temp" на
' имя, предназначенное для нового сегмента.
dsoPartitionNew.Name = strPartitionNew
dsoPartitionNew.AggregationPrefix = strPartitionNew & "_"

' Установка таблицы фактов для нового сегмента.
dsoPartitionNew.SourceTable = _
```

```
sLQuote & "sales_fact_dec_1998" & sRQuote

' Установка свойств FromClause и JoinClause
' нового сегмента.
dsoPartitionNew.FromClause = Replace(dsoPartition.FromClause, _
dsoPartition.SourceTable, dsoPartitionNew.SourceTable)

dsoPartitionNew.JoinClause = Replace(dsoPartition.JoinClause, _
dsoPartition.SourceTable, dsoPartitionNew.SourceTable)

' Изменение определения среза данных, используемого новым
' сегментом, путем изменения свойств SliceValue
' попадающих под влияние уровней и измерений на требуемые значения.
dsoPartitionNew.Dimensions("Time").Levels("Year").SliceValue = "1998"
dsoPartitionNew.Dimensions("Time").Levels("Quarter").SliceValue = "Q4"
dsoPartitionNew.Dimensions("Time").Levels("Month").SliceValue = "12"

' Оценка rowcount.
dsoPartitionNew.EstimatedRows = 18325

' Добавление другого фильтра. SourceTableFilter обеспечивает
' дополнительную возможность добавить оператор WHERE к SQL-запросу,
' заполняющему этот сегмент. Мы используем этот фильтр для того,
' чтобы удостовериться, что наш новый сегмент содержит ноль строк.
' Для наших целей мы не будем менять данные
' в кубе FoodMart. Закомментируйте эту строчку,
' если хотите увидеть данные в новом сегменте.

dsoPartitionNew.SourceTableFilter = dsoPartitionNew.SourceTable _
& "." & sLQuote & "time_id" & sRQuote & "=100"

' Сохранение определения сегмента в репозитории метаданных
dsoPartitionNew.Update

' Проверка допустимости структуры нового сегмента.
IF NOT dsoPartitionNew.IsValid Then
MsgBox "Error-New partition structure is invalid."
Err.Clear
Exit Sub
End If

MsgBox "New partition " & strPartitionNew & " has been created and " _
& "processed. To see the new partition in Analysis Manager, you " _
& "may need to refresh the list of partitions in the Sales cube " _
& "of FoodMart 2000. The new partition contains no data.", , _
"ClonePart.vbs"

' Следующий закомментированный оператор обрабатывал бы сегмент.
' В реальной системе управления сегментами это, скорее всего, было бы
' отдельным процессом, возможно, управляемым с помощью DTS.
' dsoPartitionNew.Process

' Завершение.
Set dsoPartition = Nothing
Set dsoPartitionNew = Nothing
Set dsoCube = Nothing
Set dsoDB = Nothing
dsoServer.CloseServer
Set dsoServer = Nothing

End Sub
```