Глава 2. Microsoft SQL Server – платформа для построения аналитических систем

Microsoft SQL Server включает в себя компоненты и функции, которые являются частью платформы бизнес-аналитики, а также включает службы, которые можно использовать для реализации различных архитектурных решений хранилищ данных. Эти компоненты и функции представлены на рис. 15.

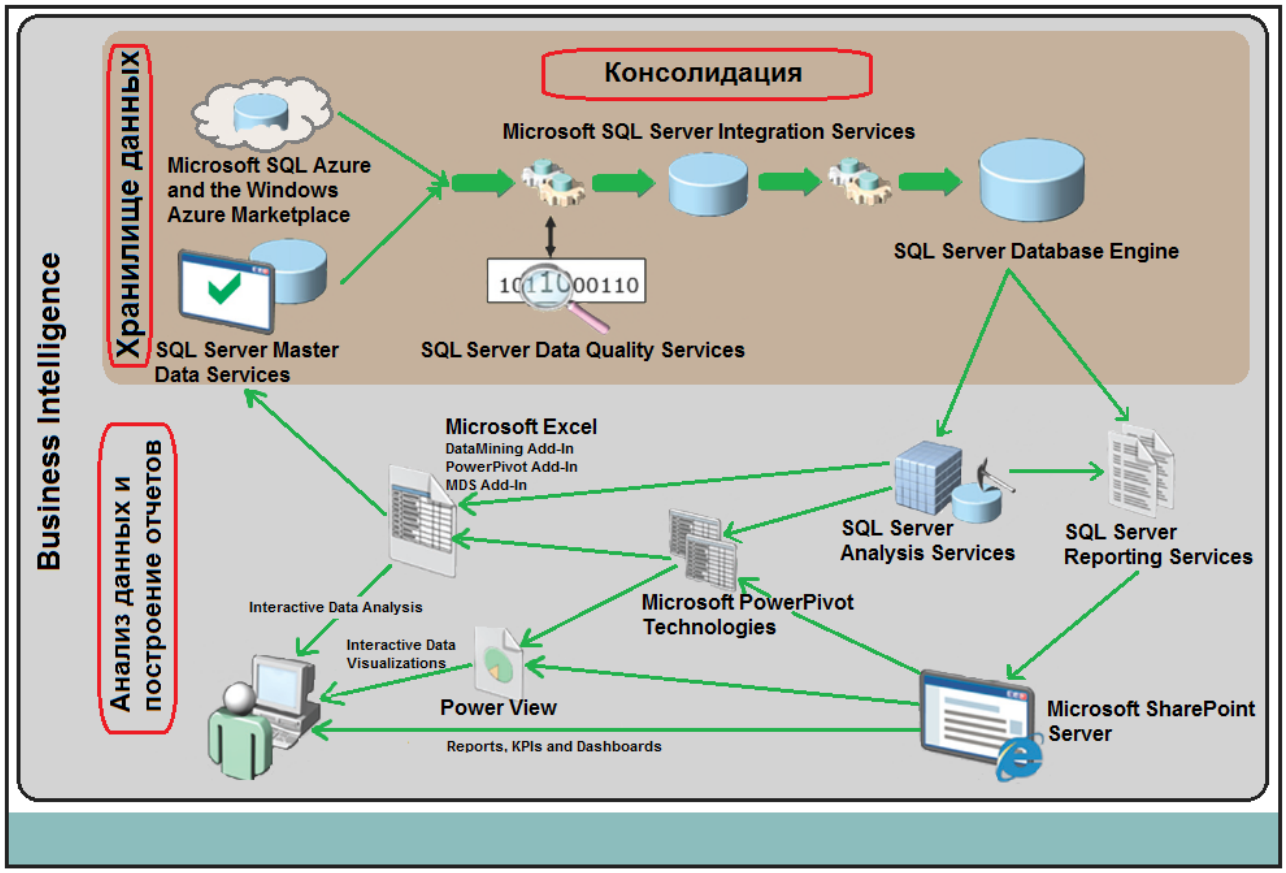

Рис. 15. Компоненты SQL Server для построения аналитических систем

Для реализации хранилищ данных Microsoft предлагает использовать следующие компоненты.

The SQL Server Database Engine. Высоко масштабируемая система управления реляционными базами данных, на которой можно реализовать хранилище данных. Редакция SQL Server Enterprise включает в себя механизмы, позволяющие оптимизировать запросы к типичному хранилищу данных.

SQL Server Integration Services. Комплексная и расширяемая платформа

для создания ETL-решений, включающая поддержку широкого спектра источников данных и многочисленные встроенные преобразования задач потока данных и потока управления для наиболее распространенных ETL потребностей.

SQL Server Master Data Services. Это решение SQL Server по управлению основными данными организации – Master Data Management. Цель управления основными данными – удостовериться в отсутствии повторяющихся, неполных, противоречивых данных в различных областях деятельности организации. Результатом успешно реализованного решения MDM становится получение надежных, централизованных данных, доступных для анализа, что влечет за собой принятие лучших бизнес-решений.

SQL Server Data Quality Services. Решение, которое обеспечивает автоматизированные и интерактивные способы очистки и/или устранения дубликатов в данных, загружаемых из источников. Служба DQS позволяет ИТ-специалисту поддерживать качество данных и обеспечивать их пригодность к использованию за счет обнаружения знаний о данных, строить наборы знаний и управлять ими.

Microsoft SQL Azure™. Облачная платформа баз данных, которая может использоваться как источник данных для хранилища.

Windows Azure Marketplace DataMarket. Облачное хранилище коммерчески доступных наборов данных, которые могут быть включены в хранилища данных или использованы для проверки и очистки данных в SQL Server Data Quality Services.

Для построения аналитических решений на основе хранилища данных используются компоненты SQL Server и некоторые другие продукты Microsoft, перечисленные ниже.

SQL Server Analysis Services. Сервис для создания многомерных и табличных аналитических моделей данных и для реализации моделей интеллектуального анализа данных, которые позволяют определить тенденции и закономерности в данных.

SQL Server Reporting Services. Решение для создания и распространения различных форм отчетов.

Microsoft SharePoint Server. Веб-портал, с помощью которого пользователи могут использовать отчеты и другие результаты работы аналитических приложений друг друга.

Microsoft Excel®. Популярный табличный процессор и инструмент для анализа данных.

Microsoft PowerPivot. Это приложение представляет собой средство анализа больших объемов данных в Excel и позволяет совместно использовать табличные модели данных в SharePoint Server.

Microsoft Power View. Представляет собой интуитивный и интерактивный инструмент визуализации данных при анализе неструктурированных данных в семантической модели бизнес-аналитики.

Контрольные вопросы

1. Какие службы и компоненты SQL Server используются для построения систем бизнес-аналитики.
2. Какие компоненты SQL Server используются для реализации хранилищ данных?
3. Для чего нужна служба SQL Server Integration Services?
4. Что представляет собой компонента SQL Server Analysis Services?
5. Для чего нужна служба SQL Server Reporting Services?
6. Что представляет собой Microsoft SharePoint Server?
7. Какие средства анализа и визуализации данных предоставляет Microsoft для аналитика?

Глава 3. Проектирование хранилищ данных

Высокоуровневый подход к реализации хранилищ данных обычно включает следующие шаги:

- ✓ Работа с аналитиками и другими заинтересованными лицами в сфере бизнеса для выявления круга вопросов, ответы на которые должно обеспечивать ХД.
- ✓ Определение данных, необходимых для ответа на эти вопросы. При этом мыслить нужно в терминах «измерений» и «фактов». Факты – это сведения об объектах и событиях, которые необходимо анализировать. Измерения представляют собой разные аспекты бизнеса, относительно которых анализируются факты.
- ✓ Определение необходимых источников данных.
- ✓ Определение приоритета каждого бизнес-вопроса для согласования масштабов будущего хранилища данных.

Далее строится логическая модель хранилища.

3.1. Логическое проектирование хранилища данных

Хранилище данных – это реляционная база данных, оптимизированная для операций чтения, поэтому модель хранилища должна быть построена с учетом минимизации операций объединения таблиц. Стандартной логической схемой применяемой для ХД является схема «Звезда» или её вариант «Снежинка». Одна схема охватывает какую-то отдельную сферу бизнеса и, как правило, ХД предприятия состоит из набора таких схем.

Таблицы в них делятся на два вида: *таблицы измерений* (или просто измерения) и *таблицы фактов*.

Таблица фактов является «центральной» таблицей, она связывается с таблицами измерений посредством внешних ключей, таким образом, они как бы окружают таблицу фактов, образуя схему «Звезды» (рис. 16).

В таблицу фактов помещаются данные, описывающие какое-либо бизнес-событие, подлежащие дальнейшему анализу. Например, факт продажи: «В филиале A продан товар B в определенный день C в количестве E на сумму D».

В основном аналитики строят отчеты на основе некоторых числовых показателей, называемых мерами, таких как выручка от реализации, стоимости, уровни запаса, объемы продаж и прочие показатели. Для построения отчетов данные в таблице фактов должны агрегироваться по различным ключевым аспектам бизнеса, например, по видам продукции, регионам, клиентам, сотрудникам, временным периодам и т.п., такая информация хранится в справочных таблицах, называемых таблицами измерений. Фильтрация данных в отчетах ведется по таблицам измерений.

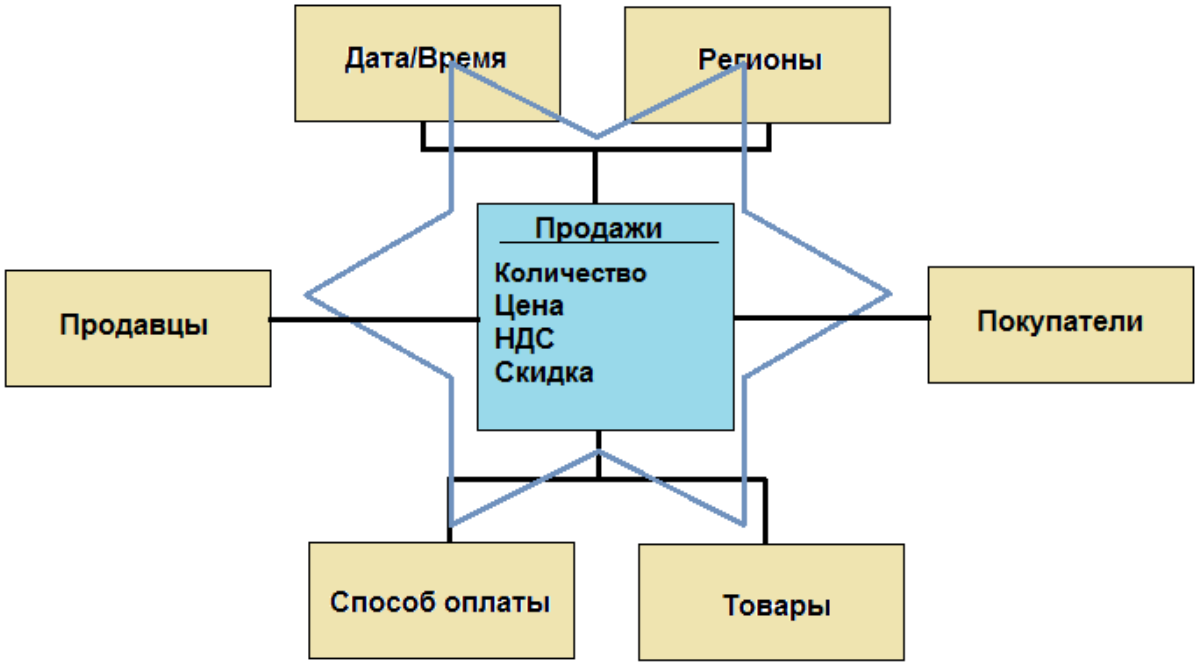

Рис. 16. Логическая схема «Звезда»

Общий подход на этапе логического проектирования хранилища заключается в выделении фактов и определении необходимых таблиц измерений.

Логическая схема «Снежинка»

В большинстве случаев схема «звезда» является оптимальной для модели хранилища данных. В этой схеме таблицы измерений, как правило, содержат избыточную информацию, т.е. являются *денормализованными* за счет присутствия функциональных зависимостей между неключевыми атрибутами. Например, если таблица «Товары» содержит, помимо прочего, атрибут

«Категория товара», то он будет функционально зависеть от атрибута «Товар», и данные о категориях будут многократно повторяться в таблице измерения. Если выполнить её нормализацию и вынести категории товаров в отдельную таблицу, называемую *таблицей подстановок*, то получим схему «Снежинка», в которой таблица фактов связывается с таблицами измерений, а те в свою очередь с таблицами подстановок посредством внешних ключей (рис. 17).

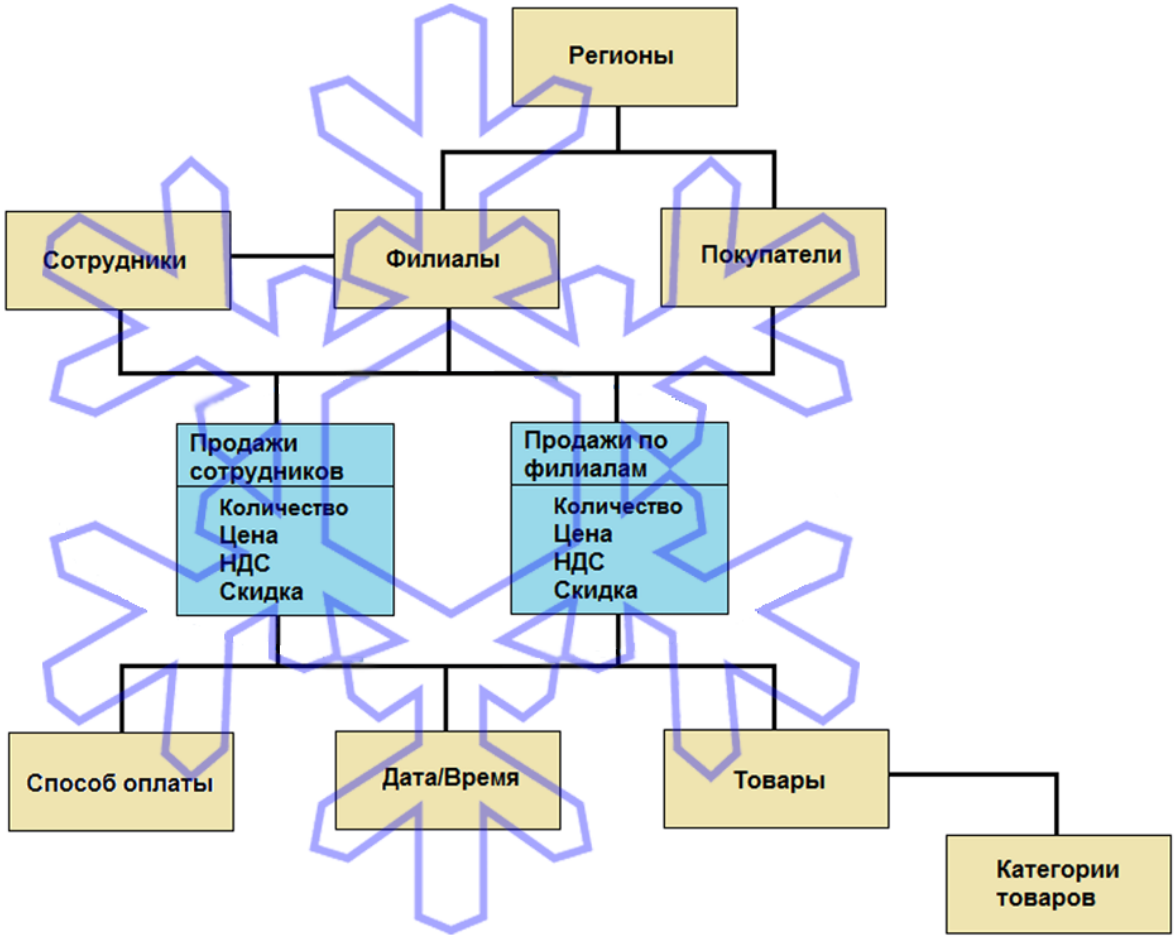

Рис. 17. Логическая схема «Снежинка»

Существует несколько причин, по которым следует перейти к схеме «Снежинка».

Если несколько измерений используют одни и те же атрибуты, то их перемещают в отдельную таблицу *подстановки*. Например, таблицы покупателей и филиалов фирмы могут включать атрибуты, отражающие их географическое расположение.

Если таблица измерения содержит небольшое подмножество данных,

которые могут часто меняться. Например, оптимальным для выполнения запросов будет хранение сведений о товарах и их категориях в одной таблице, но если категории товара часто меняются, то возможно следует выделить их в отдельную таблицу.

Если в хранилище есть несколько таблиц фактов с разной степенью детализации. Например, в хранилище есть таблица измерения с информацией о продавцах, включающая сведения о филиале, где они работают, и две таблицы фактов, в одной из которых ведется учет продаж по отдельным продавцам, а в другой продажи по филиалам в целом. В этом случае имеет смысл разделить сведения о продавцах и филиалах на две таблицы измерений, соединенных между собой посредством внешнего ключа в таблице продавцов.

Если таблица измерений сильно разрежена (содержит множество null – значений). Это происходит в случае, если таблица хранит информацию об объектах с различными свойствами. Например, в таблице товаров могут храниться сведения о товарах имеющих различные характеристики.

В большинстве случаев лучше стараться придерживаться схемы «Звезда».

3.1.1. Таблицы измерений

Измерения содержат контекстную информацию для мер. Типичный аналитический отчет представляет собой сводки по столбцам одного или нескольких измерений.

При проектировании таблиц измерений нужно рассмотреть два важных аспекта: денормализация и ключевые атрибуты.

Денормализация таблиц измерений

Таблицы измерений часто делают денорамализованными, т.е. содержащими избыточную информацию, для того чтобы избежать лишних операций соединения и тем самым повысить производительность запросов и предоставить пользователям широкий набор атрибутов, по которым можно проводить анализ. Например, рассмотрим таблицу измерения «dimGeography», в которой хранятся данные по городам и регионам продаж (рис. 18).

keyGeography	CodeRegion	NameRegion	PostalCode	NameCity	District
1	64	Саратовская область	410005	Саратов	Приволжский
2	64	Саратовская область	413100	Энгельс	Приволжский
3	64	Саратовская область	413840	Балаково	Приволжский
4	63	Самарская область	443000	Самара	Приволжский
5	63	Самарская область	445000	Тольятти	Приволжский
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Рис. 18. Пример денормализованной таблицы измерения «dimGeography»

В транзакционной базе данных эту таблицу нужно было бы нормализовать, чтобы устранить избыточность путем создания отдельных справочников регионов и округов. Это дало бы возможность оптимизировать обновляющие транзакции и сэкономить дисковое пространство, но привело бы к снижению производительности запросов на выборку данных.

Ключевые атрибуты таблиц измерений

Так же как и в OLTP-базе, в хранилище каждая строка в таблице измерения однозначно идентифицируется первичным ключом. Часто данные для измерений попадают в ХД из транзакционных баз, где уже имеются первичные ключи. В хранилище ключи из исходной системы принято называть *бизнес-ключами.* В каких-то случаях их можно использовать в качестве первичных ключей и в хранилище, но лучшей практикой является определение нового суррогатного ключа – счетчика для строк в таблице измерения. Это делается по следующим причинам:

- ✓ в таблицу измерений могут попадать данные из различных источников, и нет никакой гарантии, что ключи будут уникальными и совместимыми по типу данных;
- ✓ использование простых ключей с целочисленными значениями, как правило, приводит к улучшению производительности запросов, содержащих соединение таблиц фактов и измерений;
- ✓ необходимо предусмотреть возможность изменения атрибутов, при этом хранилище должно по-прежнему отражать правильные показатели до изменения данных и после. Чтобы достичь этого, в таблице измерения нужно хранить две версии записи с одним и тем же бизнес ключом, и если

он будет использован в качестве первичного, это приведет к нарушению уникальности ключа.

Медленно меняющиеся измерения

Значения неключевых полей некоторых таблиц измерений могут изменяться со временем, причем частота таких изменений небольшая. Например, смена имени контрагента, юридического адреса поставщика или изменение статуса клиента. Часто чтобы не потерять хронологическую целостность данных необходимо отслеживать эти изменения. Для этого используется механизм медленно меняющихся измерений (анг. *Slowly Changing Dimensions – SCD*).

Выделяют три типа медленно меняющихся измерений.

- ✓ К первому типу относятся те измерения, для которых не отслеживаются изменения данных во времени, т.е. значения полей просто обновляются. В этом случае невозможно объединить данные в разных временных периодах, если менялись атрибуты измерений.
- ✓ Для второго типа измерений отслеживание изменений осуществляется следующим образом: старая запись помечается как утратившая актуальность и добавляется новая запись с новым суррогатным ключом, но прежним бизнес-ключом, по которому можно определить, что эта один и тот же объект, но с обновленными полями. В этом случае к таблице можно добавить ряд служебных столбцов, позволяющих реализовать логику отслеживания изменений данных. Основная идея – указать временной диапазон, для которого действует определенное значение атрибута измерения и, возможно, дополнительно добавить логический атрибут, который указывает, какое значение в данный момент активно.
- ✓ К медленно меняющимся измерениям третьего типа относятся те измерения, в которых поддерживается отслеживание изменений данных во времени путем добавления в структуру таблиц полей, хранящих предыдущие значения.

На рис. 19 показаны примеры всех трех типов медленно меняющихся

измерений.

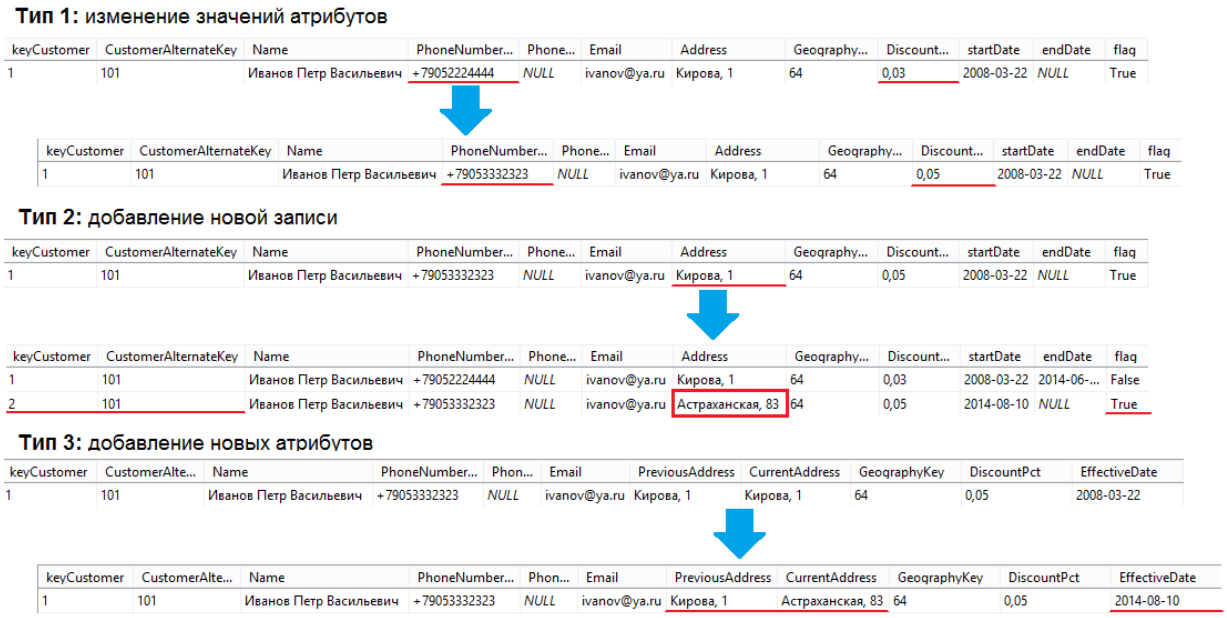

Рис. 19. Примеры различных типов медленно меняющихся измерений

В одной таблице можно использовать и комбинацию различных типов медленно меняющихся измерений, как правило, это комбинация первого и второго типов. В примере на рис. 19 для таблицы «dimCustomer» сохраняется история изменений для столбца «Address» и перезаписываются значения для столбцов «PhoneNumber» и «Discount». Но подобные комбинирования порождают новую проблему. При перезаписывании значения возникает вопрос: обновлять только последнюю версию или все предыдущие тоже? Возможны оба варианта решения проблемы.

Быстро меняющиеся измерения

Значения атрибутов измерений могут меняться часто за короткий период времени и, если эти изменения важны для последующего анализа данных, то необходимо фиксировать факт этих изменений. В этом случае применение механизма медленно меняющихся размерностей по второму типу возможно только при невысокой кардинальности таблицы, в противном случае нужны другие подходы к решению проблемы быстро меняющихся размерностей (анг. *Rapidly Changing Dimensions – RCD*).

Основным приемом в таком случае является перемещение этих атрибутов

в одну или несколько отдельных таблиц *мини измерений* (анг. *mini Dimension*), соединяющихся непосредственно с таблицей фактов. Это делается для повышения доступности данных в таблице фактов.

Таким образом, история изменений атрибутов будет храниться в таблице фактов, однако, для удобства можно в основной таблице измерения добавить внешний ключ для ссылки на текущие значения элемента, хранящиеся в таблице мини измерения, как показано на рис. 20.

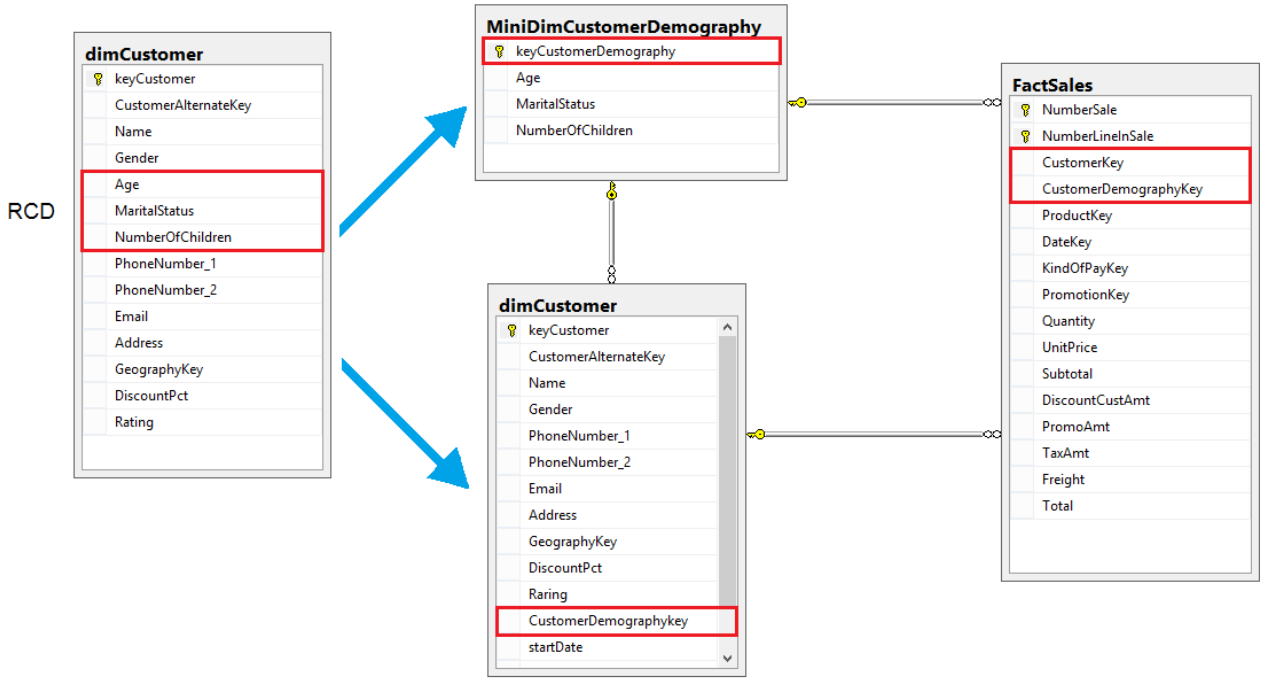

Рис. 20. Пример декомпозиции таблицы, содержащей быстро меняющиеся измерение

Иногда для быстро меняющихся атрибутов используют дискретный диапазон изменений, чтобы сократить объем данных в таблице. Если количество атрибутов и число их возможных различных значений не велико, то можно предварительно заполнить таблицу мини измерения всевозможными сочетаниями значений атрибутов, как показано на рис. 21. При этом соотнесение с каким-либо диапазоном значений происходит в момент загрузки данных в хранилище, на основании значений из транзакционной базы.

keyCustomerDemography	Age	MaritalStatus	NumberOfChildren
1	18-23	False	0
2	18-23	False	1
3	18-23	False	2
4	24-30	False	0
5	24-30	False	1
6	24-30	True	0
7	24-30	True	1
•••	•••	•••	•••

Рис. 21. Использование дискретного диапазона изменений для быстро меняющихся атрибутов

Поскольку схема «звезда», как правило, охватывает какую-то отдельную сферу бизнеса, то для корректного представления деятельности предприятия в целом необходимо иметь возможность связать разные схемы друг с другом. Для этого в разных схемах используют одни и те же измерения. Например, данные о заказчиках используются как в продажах, так и в системе бухгалтерского учета.

Измерения, связанные с несколькими таблицами фактов, называют *общими* или *согласованными*, с одной таблицей – *частными*.

При использовании частных измерений теряется возможность сравнивать показатели из разных таблиц фактов по одним и тем же измерениям.

Измерение «дата/время»

Для обеспечения единообразия при сравнении показателей во времени большинство хранилищ данных включают в себя одну особую таблицу измерений, основанную на датах и времени, что позволяет учитывать факты в исторической ретроспективе.

При создании такой таблицы следует придерживаться следующих рекомендаций.

- ✓ Необходимо определить соответствующий уровень детализации данных. Чем выше уровень детализации, тем больше строк будет содержать таблица. В большинстве случаев детализация на уровне дней является целесообразной. Однако на некоторых предприятиях, где анализ в реальном времени имеет большое значение, можно выбрать детализацию

по часам или даже минутам, а на некоторых только по неделям или месяцам.

- ✓ Следует предусмотреть иерархию вдоль временной шкалы, поскольку аналитики часто строят отчеты с различной степенью детализации, например, по годам, по кварталам в рамках одного года, по месяцам и т.д. Таблица измерения «дата/время» должна включать в себя атрибуты для каждого уровня иерархии, на котором пользователи должны агрегировать меры.
- ✓ Возможно, необходимо включить некоторые специфические для данного бизнеса временные атрибуты.
- ✓ Продумать способ заполнения такой таблицы. В отличие от большинства других таблиц хранилища таблица измерения «дата/время» заполняется не из исходной системы. Строки обычно состоят из числового первичного ключа, который является производным от значения даты (например, 20140101 для 1 января 2014 года) и столбца для каждого атрибута размерности (например, дата, день года, название месяца, месяц года, год и т.д.). Для заполнения можно использовать один из следующих методов.
  - Создать скрипт Transact-SQL, включающий в себя множество функций для работы с датой и временем, которые можно использовать в цикле для создания требуемых значений атрибутов. Например, DATEPART(datepart, date) возвращает целое число, являющееся значением соответствующей части даты (день недели, день месяца, месяц, год и так далее). DATENAME (DatePart, дата) возвращает строковое значение, являющееся названием части даты (название дня недели или месяца).
  - Использовать Microsoft® Excel, включающий в себя функции, которые можно использовать для генерации значений даты и времени с помощью формул. Используя функцию автозаполнения можно быстро создать большую таблицу значений для последовательности временных интервалов.

- Использовать инструмент бизнес-аналитики для автоматического создания таблицы измерения «дата/время». Некоторые BI-инструменты включают в себя функциональность, позволяющую генерировать «временные» таблицы измерений.

Независимо от метода заполнения необходимо выбрать соответствующую начальную и конечную точку для последовательности временных интервалов, хранящихся в таблице. Если необходимо, то предусмотреть способ расширения спектра значений времени, хранящихся в таблице в будущем. Пример таблицы измерения «дата/время» представлен на рис. 22.

KeyDate	Date	Year	Quarter	Month	Week	Day	MonthName	DayName
20150301	2015-03-01	2015	1	3	9	1	Март	воскресенье
20150302	2015-03-02	2015	1	3	10	2	Март	понедельник
20150303	2015-03-03	2015	1	3	10	3	Март	вторник
20150304	2015-03-04	2015	1	3	10	4	Март	среда
20150305	2015-03-05	2015	1	3	10	5	Март	четверг
20150306	2015-03-06	2015	1	3	10	6	Март	пятница
20150307	2015-03-07	2015	1	3	10	7	Март	суббота
20150308	2015-03-08	2015	1	3	10	8	Март	воскресенье
20150309	2015-03-09	2015	1	3	11	9	Март	понедельник
20150310	2015-03-10	2015	1	3	11	10	Март	вторник
20150311	2015-03-11	2015	1	3	11	11	Март	среда
20150312	2015-03-12	2015	1	3	11	12	Март	четверг
20150313	2015-03-13	2015	1	3	11	13	Март	пятница
20150314	2015-03-14	2015	1	3	11	14	Март	суббота
20150315	2015-03-15	2015	1	3	11	15	Март	воскресенье
20150316	2015-03-16	2015	1	3	12	16	Март	понедельник
20150317	2015-03-17	2015	1	3	12	17	Март	вторник
20150318	2015-03-18	2015	1	3	12	18	Март	среда
20150319	2015-03-19	2015	1	3	12	19	Март	четверг
20150320	2015-03-20	2015	1	3	12	20	Март	пятница

Рис. 22. Пример таблицы измерения «дата/время»

Иерархии

В таблице измерения атрибуты могут быть функционально зависимы друг от друга и образовывать иерархии. Таким образом, появляется возможность строить отчеты с различным уровнем детализации, переходя от одного уровня иерархии к другому. Например, в таблице измерения «дата/время», представленной на рис. 22, атрибуты образуют иерархию и, создавая отчеты, можно пройти через следующие её уровни:

Year → Quarter → MonthName → Week → Date.

Механизм иерархий очень полезен и часто используется в аналитических приложениях, поэтому его следует применять везде, где это возможно.

Иерархии образуются на основе содержимого функционально зависимых столбцов одной таблицы измерений или на основе связи между таблицей измерения и таблицей подстановок в схеме «Снежинка».

Вырожденные измерения

Термин *вырожденное измерение* (анг. Degenerate Dimension) принадлежит Ральфу Кимбаллу, который наряду с Биллом Инмоном считается родоначальником технологии современных хранилищ данных.

Если таблица фактов содержит данные, детализированные на уровне операций внутри одной транзакции, то часто одним из её ключевых атрибутов является идентификатор транзакции, их содержащей. Это могут быть номера заказов, номера транзакций по кредитным картам, номера счетов, номера счетов-фактур и т.п. Эти идентификаторы присваиваются учетными системами, в которых данные по транзакциям в целом и по их отдельным операциям, как правило, хранятся в разных таблицах, и таблица с детальными данными содержит внешний ключ для ссылки на транзакцию их содержащую.

Таким образом, вырожденное измерение представляет собой ключевой атрибут в таблице фактов, с которым не связана ни одна конкретная таблица измерений, так как все данные распределены по различным измерениям.

Несмотря на то, что для таких атрибутов не существует связанной с ними отдельной таблицы измерения, они могут быть полезны для группировки родственных записей в таблице фактов. Например, на рис. 23 представлена таблица фактов «FactSales» с атрибутом «NumberSale», который является вырожденным измерением и по нему можно сгруппировать все товары, купленные в одной корзине.

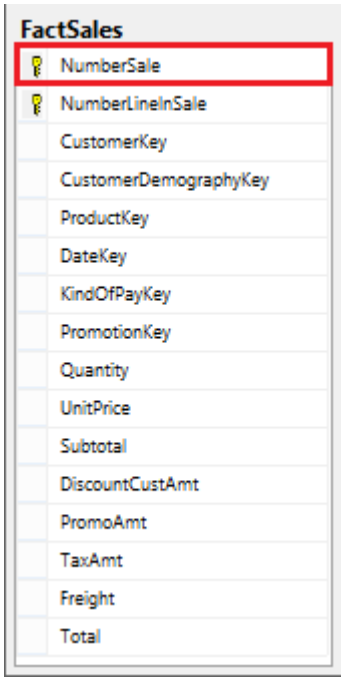

Рис. 23. Пример вырожденного измерения в таблице фактов

В таблице фактов может быть несколько вырожденных измерений. Например, таблица фактов по отгрузкам товаров может содержать как номер заказа, так и номер товарной накладной.

Вместо аутентичного значения из учетной системы возможно использование суррогатного ключа, но в этом случае измерение уже не будет вырожденным. Использование именно вырожденного измерения полезно для поддержания связи с учетными системами.

Если в измерении есть атрибут, за счет которого кардинальность таблицы растет пропорционально кардинальности таблиц фактов, то, возможно, следует перенести его в таблицу фактов и сделать вырожденным измерением [3].

3.1.2. Таблицы фактов

При проектировании таблиц фактов нужно рассмотреть следующие вопросы:

- ✓ перечень необходимых мер – числовых показателей, по которым будут анализироваться бизнес-процессы;
- ✓ уровень детализации данных, по которым в дальнейшем будет проводиться агрегация;

- ✓ перечень ключевых столбцов и любые дополнительные данные, которые должны быть сохранены в таблице фактов.

Грануляция данных

Уровень детализации данных, или *гранулярность,* является одним из наиболее важных вопросов для таблицы фактов. Например, в таблице фактов необходимо хранить данные о продажах. Одна продажа может включать в себя несколько позиций и в таблице фактов можно сохранять данные на уровне продажи или на уровне отдельных позиций по продажам. На рис. 24 показан пример таблицы фактов с гранулярностью на уровне продаж. Каждая строка отражает итоговую сумму продажи, налогов, скидки, стоимость доставки товаров. В этом случае невозможно проанализировать данные в разрезе отдельных товаров и посмотреть, например, какие товары были проданы со скидкой по промоакциям.

NumberSale	CustomerKey	DateKey	KindOfPayK...	Subtotal	DiscountCustAmt	PromoAmt	TaxAmt	Freight	Total
10001	1	22032014	1	300,00	9,00	0,00	54,00	0,00	345,00
10002	2	22032014	1	1000,00	50,00	0,00	180,00	0,00	1130,00
10003	3	22032014	2	450,00	0,00	0,00	81,00	0,00	531,00

Рис. 24. Пример таблицы фактов с гранулярностью по продажам в целом

Вариант таблицы фактов с гранулярностью на уровне отдельных позиций по продажам показан на рис. 25. В этом случае можно проводить анализ продаж по отдельным товарам.

NumberSale	NumberLineInSale	ProductKey	Customer...	DateKey	KindOf...	Promo...	Quantity	UnitPri...	Subtotal	Disco...	Promo...	TaxAmt	Freight	Total
10001	1	1	1	22032014	1	1	3	40,00	120,00	3,60	0,00	21,60	0,00	138,00
10001	2	2	1	22032014	1	1	2	80,00	160,00	4,80	0,00	28,80	0,00	184,00
10001	3	3	1	22032014	1	1	1	20,00	20,00	0,60	0,00	3,60	0,00	23,00
10002	1	14	2	22032014	1	1	6	100,00	600,00	30,00	0,00	108,00	0,00	678,00
10002	2	55	2	22032014	1	1	2	200,00	400,00	20,00	0,00	72,00	0,00	452,00
10003	1	12	3	22032014	2	1	1	450,00	450,00	0,00	0,00	81,00	0,00	531,00

Рис. 25. Таблица фактов с грануляцией на уровне отдельных позиций по продажам

Ключевые столбцы

Таблица фактов содержит, помимо числовых показателей, ссылки внешними ключами на таблицы, содержащие измерения, по которым проводится анализ. Как правило, внешние ключи, взятые все вместе,

однозначно определяют каждую строку в таблице фактов и таким образом сообща формируют уникальный ключ, поэтому все внешние ключи можно использовать как составной первичный ключ или можно добавить более простой ключ.

Меры

Для эффективной работы с ХД важно заранее произвести все скалярные операции над значениями в столбцах таблицы фактов и хранить вычисленные значения как отдельные меры. Например, помимо количества проданного товара и цены одной единицы товара, необходимо хранить в отдельном столбце заранее вычисленную итоговую сумму.

Меры в таблице фактов являются аргументами агрегатных функций, применяемых для вычисления различных показателей. По тому, как можно агрегировать меры, их можно разделить на три категории.

- ✓ Аддитивные меры. Это простейший тип мер, которые могут суммироваться по всем измерениям. Например, в таблице фактов продаж такая мера как объем продаж (поле «total») может агрегироваться по товарам, клиентам, временным периодам и т.п.
- ✓ Неаддитивные меры. Меры, которые не имеет смысла суммировать.
- ✓ Полуаддитивные меры. Меры, которые могут суммироваться только по некоторым измерениям.

На рис. 26 показана схема хранилища данных «MyStore» типа «снежинка». Это хранилище построено для гипотетической фирмы, занимающейся продажей товаров через интернет. Все примеры в этом учебном пособии демонстрируются на этом хранилище. Оно состоит из двух таблиц фактов и десяти таблиц измерений.

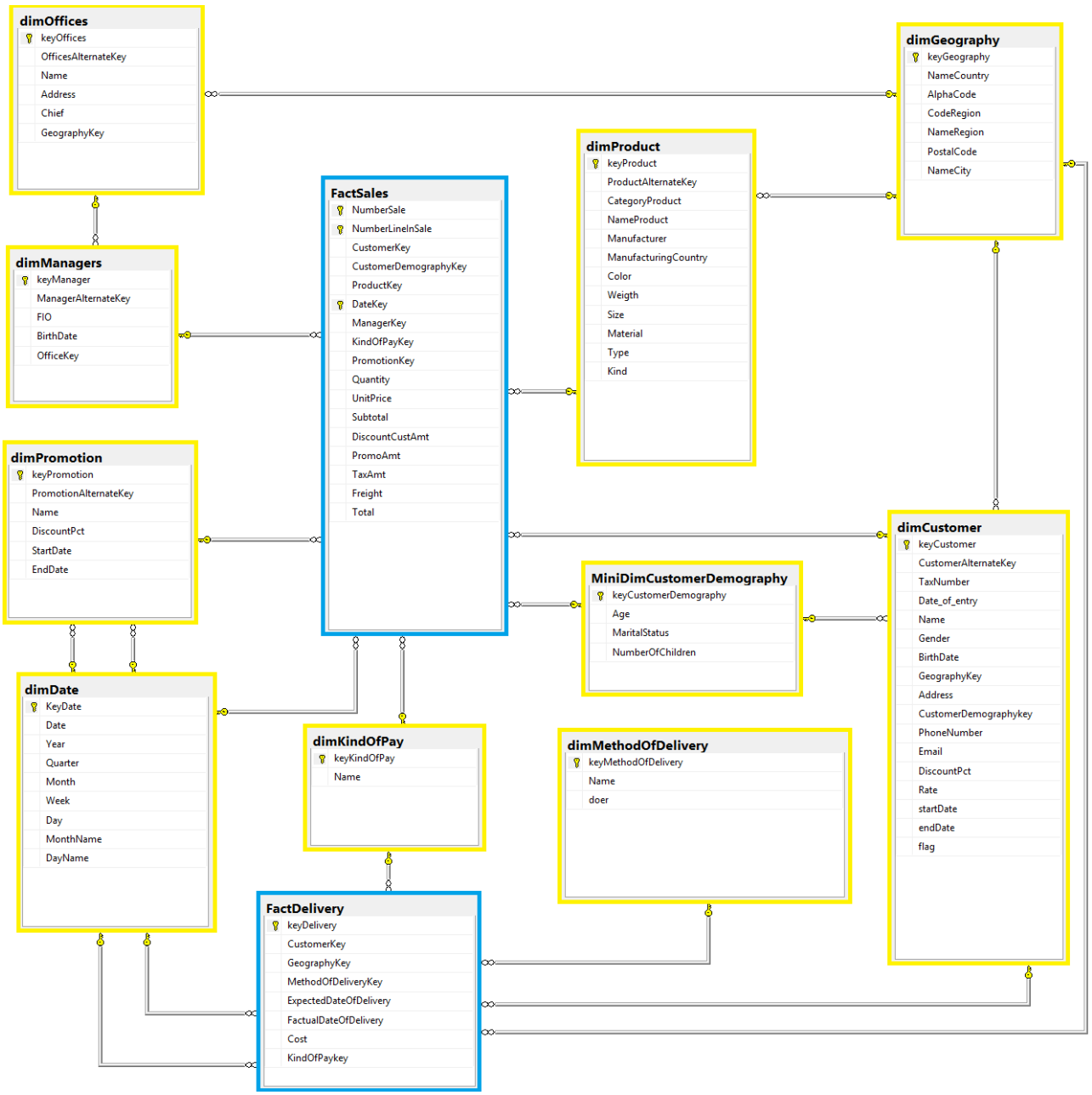

Рис. 26. Схема «снежинка» для хранилища данных «MyStore»

Таблицы фактов

«FactSales» – содержит данные об интернет-продажах каждому клиенту с детализацией по каждому отдельному товару в продаже.

Столбец	Тип данных	Допустимость Null-значений	Описание
NumberSale	int	Not Null	Поле первичного ключа. Идентификационный номер продажи
NumberLineInSale	int	Not Null	Поле первичного ключа Порядковый номер товара в

			продаже
CustomerKey	int	Not Null	Номер клиента, совершившего покупку. Внешний ключ к таблице измерения «DimCustomer»
CustomerDemographyKey	int	Null	Код текущего демографического статуса клиента. Внешний ключ к таблице измерения «MiniDimCustomerDemography»
ProductKey	int	Not Null	Код товара. Внешний ключ к таблице измерения «DimProduct»
DateKey	bigint	Not Null	Ссылка на дату совершения покупки. Внешний ключ к таблице измерения «DimDate». По этому полю задан кластеризованный индекс в таблице
ManagerKey	int	Null	Код менеджера, оформившего продажу. Внешний ключ к таблице измерения «DimManagers»
KindOfPayKey	int	Null	Код способа оплаты. Внешний ключ к таблице измерения «DimKindOfPay»
PromotionKey	int	Null	Код рекламной акции. Внешний ключ к таблице «DimPromotion»
Quantity	int	Not Null	Количество купленного товара
UnitPrice	money	Not Null	Цена единицы товара
Subtotal	money	Not Null	Промежуточный итог. Вычисляется как Quantity * UnitPrice
DiscountCustAmt	money	Not Null	Сумма клиентской скидки
PromoAmt	money	Not Null	Сумма скидки по промоакциям
TaxAmt	money	Not Null	Сумма НДС
Freight	money	Not Null	Сумма за доставку
Total	money	Not Null	Итоговая сумма

«FactDelivery» – содержит данные о совершенных доставках товаров по каждому клиенту.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyDelivery	int	Not Null	Поле первичного ключа. Идентификационный номер доставки
CustomerKey	int	Not Null	Номер клиента, кому осуществляется доставка. Внешний ключ к таблице измерения «DimCustomer»
GeographyKey	int	Not Null	Код территориальной расположенности адреса доставки. Внешний ключ к

			таблице измерения «DimGeography»
MethodOfDeliveryKey	int	Not Null	Код способа доставки. Внешний ключ к таблице измерения «DimMethodOfDelivery»
ExpectedDateOfDelivery	bigint	Not Null	Ссылка на дату предполагаемой доставки. Внешний ключ к таблице измерения «DimDate»
FactualDateOfDelivery	bigint	Not Null	Ссылка на дату фактической доставки. Внешний ключ к таблице измерения «DimDate»
Cost	money	Not Null	Стоимость доставки.
KindOfPayKey	int	Not Null	Код способа оплаты. Внешний ключ к таблице измерения «DimKindOfPay»

Таблицы измерений

«dimDate» – содержит временную шкалу с грануляцией по дням, заполняется с помощью SQL-скрипта.

Столбец	Тип данных	Допустимость Null-значений	Описание
KeyDate	bigint	Not Null	Первичный ключ, являющийся производным от временного значения
Date	date	Not Null	Дата в формате ГГГГ-ММ-ДД
Year	int	Not Null	Год
Quarter	int	Not Null	Номер квартала
Month	int	Not Null	Номер месяца
Week	int	Not Null	Номер недели от начала года
Day	int	Not Null	День недели
MonthName	nvarchar(20)	Not Null	Название месяца
DayName	nvarchar(20)	Not Null	Название дня недели

«dimGeography» – содержит данные о городах, почтовых индексах, регионах и странах.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyGeography	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
NameCountry	nvarchar(50)	Not Null	Название страны
AlphaCode	char(3)	Not Null	Код страны
CodeRegion	int	Not Null	Код региона
NameRegion	nvarchar(45)	Not Null	Название региона
PostalCode	nvarchar(15)	Not Null	Почтовый код

NameCity	nvarchar(30)	Not Null	Название города

«dimCustomer» - содержит сведения о клиентах.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyCustomer	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
CustomerAlternateKey	nvarchar(12)	Not Null	Бизнес ключ клиента из транзакционной базы данных
TaxNumber	char(15)	Null	ИНН клиента. SCD тип 1
Date_of_entry	int	Null	Дата регистрации в интернет магазине
Name	nvarchar(50)	Not Null	ФИО клиента. SCD тип 2
Gender	char(1)	Null	Пол
BirthDate	date	Null	День рождения клиента
GeographyKey	int	Null	Код территориальной расположенности адреса клиента. SCD тип 2. Внешний ключ к таблице измерения «dimGeography»
Address	nvarchar(50)	Null	Адрес. SCD тип 1
CustomerDemographykey	int	Null	Код текущего демографического статуса клиента. Внешний ключ к таблице измерения «MiniDimCustomerDemography». SCD тип 1
PhoneNumber	varchar(12)	Null	Номер дополнительного телефона. SCD тип 1
Email	varchar(30)	Null	Адрес электронной почты. SCD тип 1
DiscountPct	float	Null	Накопительная скидка клиента. Задается как коэффициент, по умолчанию равный единице. SCD тип 1
Rate	int	Null	Текущий рейтинг клиента, зависит от количества совершенных покупок. SCD тип 1. Вычисляемый столбец
startDate	date	Null	Дата с которой данные действительны. Применяется для отслеживания медленно меняющихся размерностей
endDate	date	Null	Дата по которую данные действительны. Применяется для отслеживания медленно меняющихся размерностей
flag	bit	Not Null	Отметка об актуальности записи. Применяется для отслеживания

			медленно меняющихся размерностей

Таблица «MiniDimCustomerDemography» возникла в результате перемещения в неё часто изменяющихся атрибутов в измерении «dimCustomer».

Столбец	Тип данных	Допустимость Null-значений	Описание
keyCustomerDemography	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
Age	nvarchar(10)	Not Null	Возраст клиента
MaritalStatus	bit	Not Null	Семейное положение клиента
NumberOfChildren	int	Not Null	Количество детей у клиента

«dimPromotion» – содержит сведения о рекламных компаниях.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyPromotion	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
PromotionAlternateKey	nvarchar(15)	Not Null	Бизнес ключ рекламной компании из транзакционной базы данных
Name	nvarchar(50)	Null	Название рекламной компании.
DiscountPct	float	Not Null	Коэффициент скидки. По умолчанию значение 0
StartDate	bigint	Not Null	Ссылка на дату начала проведения акции. Внешний ключ к таблице измерения «DimDate»
EndDate	bigint	Not Null	Ссылка на дату окончания проведения акции. Внешний ключ к таблице измерения «DimDate»

«dimProduct» – содержит сведения о товарах.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyProduct	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
ProductAlternateKey	nvarchar(25)	Not Null	Бизнес ключ товара из транзакционной базы данных
CategoryProduct	nvarchar(50)	Null	Категория товара
NameProduct	nvarchar(50)	Not Null	Название товара

Manufacturer	nvarchar(50)	Null	Производитель
ManufacturingCountry	int	Null	Территориальная расположенность производства. Внешний ключ к таблице измерения «dimGeography»
Color	nvarchar(30)	Null	Цвет товара
Weigth	decimal(8,2)	Null	Вес товара
Size	nvarchar(10)	Null	Размеры товара
Material	nvarchar(30)	Null	Материал
Type	nvarchar(15)	Null	Тип механизма
Kind	nchar(1)	Null	Признак мужской или женский

«dimKindOfPay» – содержит сведения о способах оплаты.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyKindOfPay	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
Name	nvarchar(30)	Not Null	Наименование способа оплаты

«dimMethodOfDelivery» – содержит сведения о способах доставки товаров.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyMethodOfDelivery	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
Name	nvarchar(30)	Not Null	Наименование способа доставки
doer	nvarchar(30)	Not Null	Наименование исполнителя

«dimOffices» – содержит сведения об офисах продаж.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyOffices	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
OfficesAlternateKey	nvarchar(25)	Null	Бизнес ключ офиса из транзакционной базы данных
Name	nvarchar(50)	Null	Название офиса
Address	nvarchar(50)	Not Null	Адрес офиса
Chief	nvarchar(50)	Null	Директор офиса
GeographyKey	int	Null	Территориальная

			расположенность офиса. Внешний ключ к таблице измерения «dimGeography»

«dimManagers» – содержит сведения о менеджерах, оформляющих продажи.

Столбец	Тип данных	Допустимость Null-значений	Описание
keyManager	int	Not Null	Суррогатный первичный ключ со свойством IDENTITY(1,1)
ManagerAlternateKey	nvarchar(25)	Null	Бизнес ключ менеджера из транзакционной базы данных
FIO	nvarchar(50)	Not Null	ФИО менеджера
BirthDate	date	Null	День рождения менеджера
OfficeKey	int	Null	Офис, в котором работает менеджер. Внешний ключ к таблице измерения «dimOffices»

Лабораторная работа 1.

Разработать логическую схему хранилища данных для предметной области согласно варианту задания.

Для этого нужно выделить факты, которые будут анализироваться. Определить уровень их детализации. Спроектировать таблицы фактов с необходимым набором мер, проанализировав, какие скалярные операции над данными должны быть выполнены заранее, до загрузки в таблицу фактов, а также какие меры и по каким измерениям можно агрегировать.

В соответствии с набором мер в таблицах фактов необходимо продумать структуру таблиц измерений. Выбрать необходимый уровень детализации данных в них и рассмотреть целесообразность перехода к схеме «снежинка» в случае, если некоторые таблицы измерений будут содержать одинаковые данные или для различных отчетов требуется разная степень грануляции. В любом случае переход к схеме «снежинка» должен быть обоснован. Далее нужно подумать о необходимости добавления суррогатных ключей, помимо бизнес-ключей, наследуемых из транзакционных баз данных, об отслеживании

истории изменений для тех или иных атрибутов, составе временных меток в таблице измерения «дата/время».

Варианты предметных областей

Страховая компания, грузовые перевозки, кредитование, туристическая фирма, оптовая торговля (продуктовый, игрушки, косметика, спорттовары), розничный или интернет-магазин (книжный, продуктовый, игрушки, косметика, спорттовары, техника, компьютерная техника), торговля недвижимостью, аренда автомобилей, продажа автомобилей, кадровое агентство, аренда недвижимости, учебный центр, сервисный центр, частная клиника, учет нарушений ПДД, фармацевтическая компания, строительная компания, производство.

3.2. Физическое проектирование хранилища данных

Так как хранилище – это реляционная база данных, то основные приемы создания базы, таблиц, связей и т.п. для него в SQL Server те же, что и для OLTP-баз. Однако существует ряд особенностей, на которые следует обратить внимание при реализации хранилища на SQL Server.

Во-первых, данные в хранилище загружаются периодически по расписанию, а не в режиме реального времени, а значит, можно использовать простую модель восстановления (simple recovery model), в которой место в журнале зафиксированных транзакций освобождается автоматически и которая требует минимального места на диске для его хранения. Если же применяется модель полного восстановления (full recovery model) или модель с неполным протоколированием (bulk Logged recovery model), то необходимо регулярно создавать резервные копии журнала транзакций, так как он постоянно растет с каждой загрузкой данных. Подробные сведения о различных моделях восстановления базы данных можно узнать в электронной документации по ссылке: https://msdn.microsoft.com/ru-ru/library/ms189275(v=sql.110).aspx

Во-вторых, для облегчения процесса загрузки данных в хранилище,

возможно, потребуется создать промежуточные таблицы (staging tables). Они используются для временного хранения данных перед этапом очистки и объединения с данными из других источников, а также могут служить так называемым «буфером» между хранилищем и транзакционными базами данных. Например, если в исходной базе данных произошли изменения, требующие модификации способов загрузки данных в хранилище, то достаточно лишь изменить запрос, считывающий эти данные в промежуточные таблицы, а сам ETL-процесс будет выполняться обычным образом.

Промежуточные таблицы являются внутренним инструментом организации ETL-процесса и никогда не показываются конечным пользователям. Для удобства их можно хранить в отдельной схеме базы данных (Schema), что упростит администрирование. В типовом хранилище данных достаточно двух схем: одна с обычными таблицами ХД и другая с промежуточными [4].

Еще один момент, который следует обязательно учитывать при создании хранилища данных – это механизм управления дисковым пространством. По умолчанию в SQL Server применяется динамическое управление, т.е. база данных может автоматически увеличиваться или сжиматься по мере необходимости, но операции эти происходят в момент загрузки хранилища и замедляют её. Также многочисленные операции по незначительному автоматическому увеличению или сжатию базы могут вызвать фрагментирование данных, а это влияет на производительность запросов, считывающих большой объем данных. Поэтому режим **автоматического сжатия** (Auto Shrink) для хранилищ должен быть отключен, а ростом всех файлов данных и журналов следует управлять вручную, заранее предусмотрев достаточный объем диска для хранения данных и регистрационных журналов.

Ниже приведен скрипт создания базы данных для хранилища «MyStore».

```
USE Master;
IF DB_ID(N'MyStore') IS NOT NULL
DROP DATABASE [MyStore];
GO
```

```
CREATE DATABASE [MyStore]
ON  PRIMARY
( NAME = N'MyStore', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyStore.mdf' , SIZE = 51200KB ,
FILEGROWTH = 10240KB )
 LOG ON
( NAME = N'MyStore_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyStore_log.ldf' , SIZE = 10240KB ,
FILEGROWTH = 10%)
 COLLATE Cyrillic_General_100_CI_AI
GO
ALTER DATABASE [MyStore] SET RECOVERY SIMPLE WITH NO_WAIT;
GO
ALTER DATABASE [MyStore] SET AUTO_SHRINK OFF
GO
```

Изначально для хранилища создается один файл данных MyStore.mdf с начальным размером 50 Мбайт и приращением в 10 Мбайт и файл журнала транзакций MyStore_log.ldf с начальным объемом в 10 Мбайт и приращением на 10%. После создания базы данных модель восстановления меняется на простую.

Существенное влияние на производительность, масштабируемость и управляемость хранилища данных оказывает выбор того или иного способа хранения и индексирования данных, поэтому, уделив должное внимание проектированию физической модели хранилища, можно значительно улучшить эти аспекты.

В этом разделе будут описаны основные механизмы SQL Server, используемые при физическом проектировании хранилищ данных.

3.2.1. Файловые группы

В первую очередь следует продумать стратегию хранения данных на дисках, и хотя конкретные детали того, как данные размещаются на физических носителях, могут варьироваться от одного аппаратного решения к другому, существуют некоторые общие принципы, которые необходимо учитывать.

Прежде всего, это распределение данных в соответствии с их свойствами между физическими носителями. Компонент Microsoft SQL Server Database Engine может параллельно обрабатывать потоки при чтении данных с нескольких физических устройств, что может значительно сократить время на

извлечение данных для запроса. В большинстве хранилищ распределение данных между физическими устройствами осуществляется через избыточный массив независимых дисков – RAID, как правило, в сети хранения данных (анг. Storage Area Network – SAN) или выделенном сервере хранения. В отсутствии решения RAID, можно предусмотреть распределение таблиц хранилища по нескольким физическим дискам, в зависимости от свойств данных этих таблиц. Например, если объем данных в таблице быстро растет, то возможно, для неё следует предусмотреть диск большего объема, или поместить таблицу на более быстрый диск, если её данные часто запрашиваются.

Для распределения данных по различным физическим дискам в SQL Server существует механизм файловых групп.

По умолчанию при создании базы данных создается два системных файла: файл данных и файл журнала транзакций. Файлы данных содержат данные и объекты, такие как таблицы, индексы, хранимые процедуры и представления. Файлы журнала содержат сведения, необходимые для восстановления всех транзакций в базе данных.

Если база содержит несколько файлов данных, то нельзя напрямую указать в каких именно файлах следует хранить данные из таблиц, но для таблиц можно указать файловые группы, к которым они будут относиться, а каждая файловая группа, в свою очередь, является своеобразным контейнером для файлов, хранящихся на различных дисках (рис. 27).

По умолчанию существует одна файловая группа PRIMARY, и все создаваемые файлы относятся к ней. Помимо самих данных эта файловая группа содержит еще и метаданные, то есть всю информацию о структуре базы данных. Удалить эту файловую группу нельзя. В случае если файловая группа PRIMARY единственная, сервер автоматически распределяет данные по файлам базы данных.

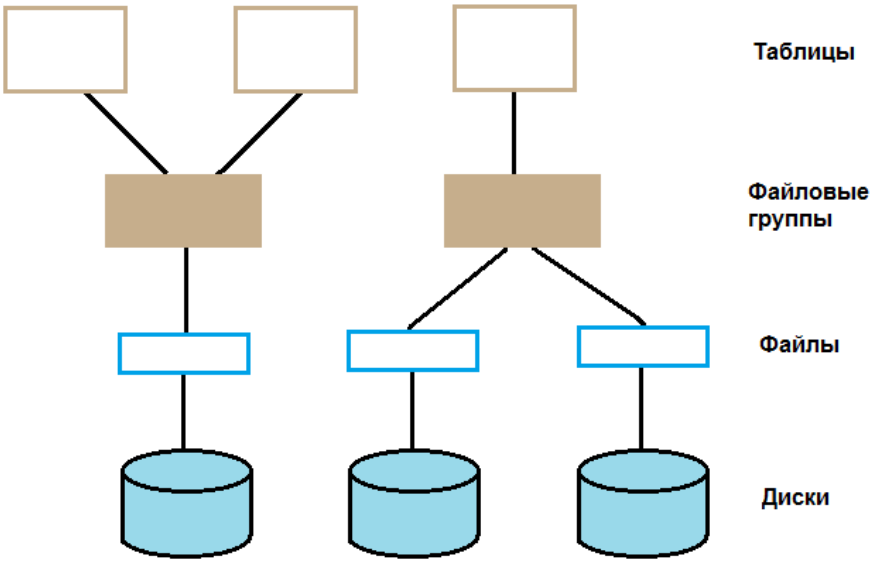

Рис. 27. Распределение данных по физическим носителям

Разработчик хранилища данных не может знать заранее, сколько будет дисков, но он может заранее предусмотреть инфраструктуру файловых групп, исходя из свойств данных, хранящихся в таблицах. Нет общих правил о распределении таблиц по различным файловым группам и, соответственно, количестве этих групп, но можно руководствоваться следующими рекомендациями:

- ✓ создать отдельные файловые группы для быстрорастущих таблиц;
- ✓ таблиц с часто запрашиваемыми данными;
- ✓ таблиц с редко запрашиваемыми данными;
- ✓ таблиц, данные в которых не изменяются со временем (для таких таблиц не нужно каждый раз создавать резервную копию);
- ✓ создать отдельную файловую группу для индексов;
- ✓ предусмотреть файловую группу по умолчанию, в которую будут попадать таблицы, для которых не указана явно файловая группа;
- ✓ создать файловую группу для промежуточных и временных таблиц.

Желательно предусмотреть как можно больше файловых групп, это не скажется на производительности ХД, но существенно повысит управляемость его данными.

Для управления файловыми группами нужно зайти в свойства базы данных и слева в списке страниц выбрать страницу **Файловые группы** (Filegroups). Здесь можно создать файловые группы и указать их свойства (рис. 28).

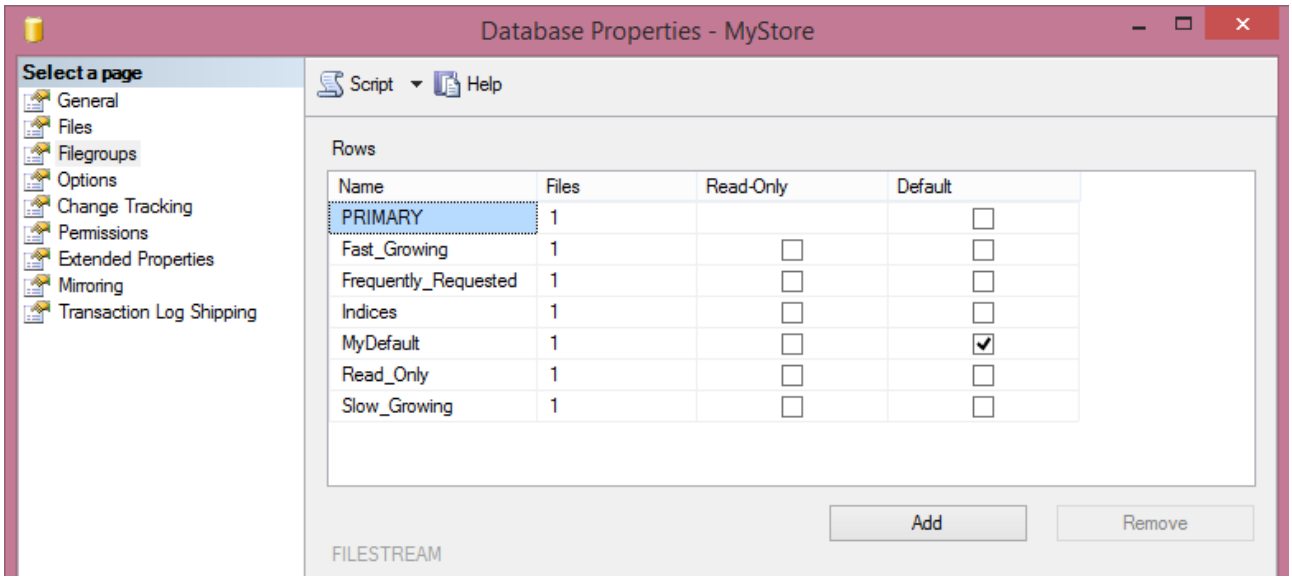

Рис. 28. Управление файловыми группами

Если для таблицы явно не указана файловая группа, то будет использоваться файловая группа, заданная по умолчанию, а это группа PRIMARY. Поскольку в этой группе хранится файл .mdf со всей важной информацией о хранилище, то возможно, следует переопределить файловую группу по умолчанию на другую.

Таблицы-справочники, данные в которых не изменяются, лучше поместить в отдельную файловую группу и затем после начальной загрузки данных установить для этой файловой группы свойство **Только для чтения** (ReadOnly). Это позволит в дальнейшем не включать её каждый раз в резервные копии ХД.

Для каждой новой файловой группы необходимо указать хотя бы один файл. Создать файл, указать файловую группу, к которой он будет относиться и все необходимые свойства можно, зайдя на страницу **Файлы** (Files) (рис. 29).

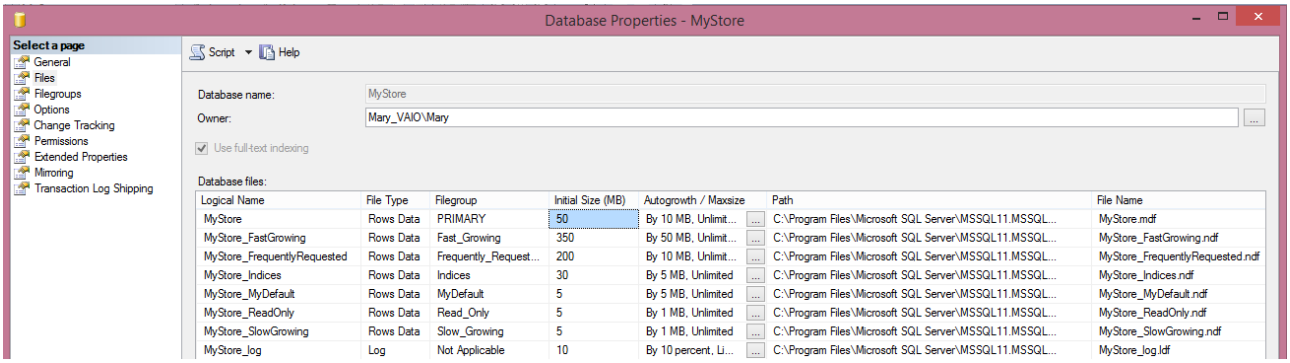

Рис. 29. Создание файлов для базы данных

Можно установить следующие свойства.

- ✓ **Начальный размер файла** (Initial Size).
- ✓ **Автоувеличение** (Autogrowth), если существующий размер файла недостаточен для хранения добавляемых данных, сервер автоматически запросит систему выделить файлу дополнительное дисковое пространство. Объем дополнительного дискового пространства (в процентах или мегабайтах) указывается в поле **Увеличение размера файла** (File Growth), а в разделе **Максимальный размер файла** (Maximum File Size) можно ограничить максимальный размер файла, установив переключатель **Ограничен** (Limited to). Для автоматического увеличения размера базы данных нужно установить флажок **Разрешить авторасширение** (Enable Autogrowth).

Управлять файловыми группами и файлами можно с помощью кода T-SQL. В следующем примере создается файловая группа «Fast_Growing» для базы данных «MyStore» и к ней добавляется файл с логическим именем «MyStore_FastGrowing».

```
USE [master]
GO
ALTER DATABASE [MyStore] ADD FILEGROUP [Fast_Growing]
GO
ALTER DATABASE [MyStore] ADD FILE
( NAME = N'MyStore_FastGrowing',FILENAME = N'C:\Program Files\Microsoft
SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\MyStore_FastGrowing.ndf',SIZE =
358400KB, FILEGROWTH = 51200KB)
TO FILEGROUP [Fast_Growing]
GO
```

Подробный синтаксис инструкции можно посмотреть в электронной

документации по SQL Server 2012 в статье «Параметры инструкции ALTER DATABASE для файлов и файловых групп (Transact-SQL)» по адресу https://msdn.microsoft.com/ru-ru/library/bb522469(v=sql.110).aspx.

Также создавать и управлять файловыми группами можно на странице свойств **Общие** (General) при добавлении нового файла к базе данных, выбрав для него уже существующую файловую группу или создав новую. Для этого в столбце **Файловые группы** (Filegroup) из выпадающего списка нужно выбрать опцию **<Новая файловая группа>** (new filegroup) и задать необходимые параметры.

Следует также помнить, что распространенной практикой является размещение файлов данных и журналов транзакций на различных дисках.

После создания хранилища данных с набором необходимых файловых групп, можно создавать таблицы с привязкой к ним.

Таблицы создаются обычным образом. При использовании конструктора таблиц в Management Studio задать файловую группу можно в окне **Свойства** (Properties), которое вызывается по кнопке F4. В пункте **Спецификация регулярного пространства данных** (Regular Data Space Specification) нужно выбрать из списка необходимую файловую группу (рис. 30). При этом BLOB (анг. Binary Large OBject — двоичный большой объект) поля таблиц в SQL Server можно выносить в отдельные файловые группы, указав их в пункте Text/Image Filegroup.

Если таблицы создаются с помощью кода, то в конце дописывается инструкция *ON {Имя файловой группы}*.

Далее приведен пример кода создания таблицы измерения «dimDate» с привязкой к файловой группе «Frequently_Requested».

```
CREATE TABLE dimDate
(KeyDate BIGINT NOT NULL,
[Date] DATE NOT NULL,
[Year] INT NOT NULL,
[Quarter] INT NOT NULL,
[Month] int NOT NULL,
[Week] int NOT NULL,
[Day] int NOT NULL,
```

```
[MonthName] NVARCHAR(20) NOT NULL,
[DayName] NVARCHAR(20) NOT NULL,
CONSTRAINT PKDW_1 PRIMARY KEY (KeyDate)
) ON [Frequently_Requested]
```

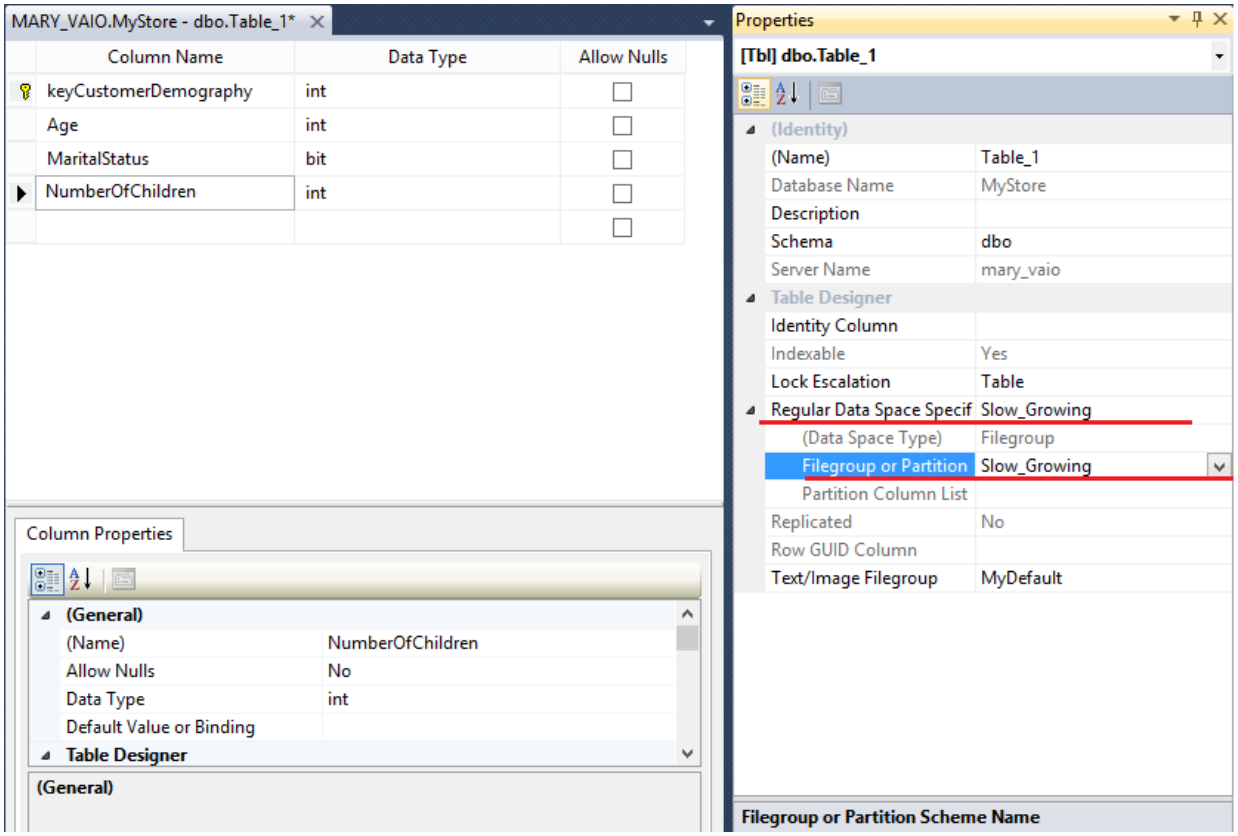

Рис. 30. Создание таблицы с привязкой к файловой группе

В прил. 1 целиком приведен скрипт создания хранилища данных «MyStore», с набором необходимых файловых групп и файлов.