В память о моих родителях,
Миле и Габи Бен-Ган

Содержание

Об авторе

Ицик Бен-Ган (Itzik Ben-Gan) – сооснователь и преподаватель образовательной компании SolidQ, с 1999 года является обладателем статуса Microsoft MVP в области платформ обработки данных. Ицик провел бесчисленное количество обучающих мероприятий по всему миру, делясь опытом написания и оптимизации запросов на языке T-SQL. Автор нескольких книг, включая *T-SQL Fundamentals* и *T-SQL Querying*. Также Ицик пишет статьи для платформ *sqlperformance.com*, *ITProToday* и *SolidQ* и участвует в различных конференциях, в числе которых *PASS Summit* и *SQLBits*. Является ведущим специалистом в области T-SQL и автором курсов *Advanced T-SQL Querying, Programming and Tuning* и *T-SQL Fundamentals* в компании SolidQ.

Введение

Для меня лично оконные функции представляются наиболее важным элементом, поддерживаемым как стандартом SQL, так и его диалектом для Microsoft SQL Server, называемым T-SQL. Они позволяют выполнять вычисления применительно к целым наборам строк в очень гибкой, понятной и эффективной манере. Искусность принципов реализации оконных функций трудно переоценить, и у традиционных альтернатив со всеми присущими им недостатками нет против них ни шансов, ни аргументов. Спектр задач, которые легко решаются при помощи оконных функций, столь широк, что стоит задуматься о том, чтобы изучить эту концепцию. С момента своего появления оконные функции получили большое развитие как в SQL Server, так и в стандарте SQL. В этой книге мы будем говорить как о специфических для SQL Server свойствах применения оконных функций, так и об особенностях их использования в стандарте SQL, включая те, которые еще не реализованы в SQL Server.

Для кого эта книга

Эта книга предназначена для разработчиков SQL Server, администраторов баз данных (DBA), специалистов в области обработки данных и бизнес-аналитики, а также для тех, кто пишет запросы и разрабатывает код на языке T-SQL. В книге мы будем предполагать, что у вас за плечами есть как минимум полугодовой опыт написания и отладки запросов на языке T-SQL.

Структура книги

В книге освещаются как логические аспекты оконных функций, так и вопросы их оптимизации и практического применения.

Глава 1. Работа с окнами в SQL. В данной главе мы опишем концепцию оконных функций в языке SQL. Здесь вы познакомитесь с назначением оконных функций, их типами и элементами, участвующими в их создании и применении, включая секционирование, упорядочивание и определение границ окна.

Глава 2. Детальное изучение оконных функций. В этой главе мы погрузимся в детали и специфику оконных функций, подробно разберем агрегат-

ные и ранжирующие оконные функции, функции смещения и статистические функции (функции распределения).

Глава 3. Функции упорядоченного набора. В третьей главе мы поговорим о поддержке в языке T-SQL и стандарте SQL функций для работы с упорядоченными наборами, включая конкатенацию строк, а также рассмотрим функции для работы с гипотетическими наборами, функции обратного распределения и др. Кроме того, для стандартных функций, пока не реализованных в языке T-SQL, мы приведем работающие решения.

Глава 4. Распознавание шаблонов в строках. Здесь мы коснемся продвинутой концепции анализа данных, именуемой *распознаванием шаблонов в строках* (row-pattern recognition – RPR), которую можно рассматривать как следующий шаг развития оконных функций. В языке T-SQL данный функционал пока не реализован, но, как уже было упомянуто ранее, в этой книге мы представим вам все важнейшие аналитические концепции, даже не воплощенные на данный момент в T-SQL.

Глава 5. Оптимизация оконных функций. В этой главе мы рассмотрим способы оптимизации оконных функций применительно к SQL Server и SQL Azure Database. Мы поговорим о применении индексации с целью ускорения выполнения запросов, затронем тему параллельных вычислений, сравним построчную и пакетную обработку запросов и узнаем другие способы оптимизации оконных функций.

Глава 6. Практическое применение оконных функций в T-SQL. В заключительной главе книги мы обратимся к практическому применению оконных функций для решения распространенных бизнес-задач.

Системные требования

Оконные функции являются составной частью ядра баз данных Microsoft SQL Server и SQL Azure Database, в связи с чем их поддержка реализована во всех версиях продуктов. Для запуска примеров из этой книги вам необходимо иметь доступ к экземпляру установки SQL Server 2019 и старше (любой версии) или SQL Azure Database с установленной базой данных с *уровнем совместимости* (compatibility level) 150 и выше. Также вам придется установить базу данных с названием TSQLV5, к которой мы будем обращаться в этой книге. Если у вас нет доступа к установленному экземпляру СУБД, на сайте Microsoft присутствуют бесплатные версии. Подробнее можно узнать по адресу *https://www.microsoft.com/en-us/sql-server/sql-server-downloads*.

В качестве клиентских инструментов для подключения к базам данных и выполнения представленных в книге примеров вы можете использовать SQL Server Management Studio (SSMS) или Azure Data Studio. Я использовал SSMS для создания графических представлений планов выполнения запросов, показанных в книге. Вы можете загрузить этот инструмент по ссылке *https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms*.

Azure Data Studio можно скачать по адресу *https://docs.microsoft.com/en-us/sql/azure-data-studio/download*.

Примеры из книги

Все примеры из этой книги, а также сопутствующие данные, ресурсы и многое другое вы можете загрузить со страницы книги на сайте издательства *www.dmkpress.com*.

На странице книги присутствует ссылка на архив, включающий в себя все примеры кода из книги, а также файл *TSQLV5.sql*, содержащий инструкции для создания и наполнения базы данных TSQLV5.

Благодарности

Многие люди прямо или косвенно повлияли на выход в свет этой книги, и они, конечно, достойны упоминания и благодарностей.

Спасибо Лилах за придание смысла всему, что я делаю, и за помощь с вычиткой текста.

Благодарю всех разработчиков Microsoft SQL Server (бывших и нынешних), а в особенности: Конора Каннингема (Conor Cunningham), Джо Сака (Joe Sack), Василиса Пападимоса (Vassilis Papadimos), Марка Фридмана (Marc Friedman), Крейга Фридмана (Craig Freedman), Милана Ружича (Milan Ruzic), Милана Стоича (Milan Stojic), Йована Поповича (Jovan Popovic), Борко Новаковича (Borko Novakovic), Тобиаса Тернстрёма (Tobias Ternström), Лубора Коллара (Lubor Kollar), Умачандара Джаячандрана (Umachandar Jayachandran), Педро Лопеса (Pedro Lopes), Архениса Фернандеса (Argenis Fernandez) и многих других. Я очень признателен вам за реализацию и поддержку оконных функций в SQL Server, а также за общение со мной и ответы на мои вопросы и просьбы что-то разъяснить.

Кроме того, я благодарен всей редакторской команде издательства Pearson. Лоретта Йейтс (Loretta Yates), спасибо за то, что поверила в этот проект и позволила ему реализоваться! Моя признательность Чарви Ароре (Charvi Arora) за приложенные огромные усилия. Также я благодарен Рику Кугену (Rick Kughen) и Трейси Крум (Tracey Croom). Спасибо Ашвини Кумар (Aswini Kumar) и ее команде за работу над PDF-версией книги.

Помимо прочих, хочу выразить отдельную благодарность Адаму Маханику (Adam Machanic) за согласие стать техническим редактором книги. Существует не так много людей, понимающих архитектуру SQL Server лучше тебя. Для меня вопрос выбора человека на эту роль вообще не стоял.

Спасибо Q2, Q3 и Q4. Для меня большое счастье иметь возможность обсуждать идеи, связанные с книгой, с такими профессионалами в области SQL, как вы. Кажется, я могу делиться с вами всем без опасений за возможные последствия.

Также я хочу выразить благодарность моей компании SolidQ за последние два десятилетия. Быть частью такого коллектива и наблюдать за ростом компании – настоящее счастье. Для меня сотрудники этой компании – это больше, чем просто коллеги. Это партнеры, друзья и семья. Спасибо Фернандо Герреро (Fernando G. Guerrero), Антонио Сото (Antonio Soto) и многим другим.

Отдельную признательность я выражаю Аарону Бертрану (Aaron Bertrand) и Грегу Гонзалесу (Greg Gonzales). Мне очень приятно вести колонку на сайте *https://sqlperformance.com*. SentryOne – прекрасная компания, создающая для сообщества отличные продукты и сервисы.

Также я благодарен MVP в области SQL Server Алехандро Месе (Alejandro Mesa), Эрланду Соммарскогу (Erland Sommarskog), Аарону Бертрану (Aaron Bertrand), Полу Уайту (Paul White) и многим другим. Я очень рад быть частью этой великолепной программы. Уровень знаний этих людей просто поражает, и мне всегда приятно встречаться с ними, разговаривать на профессиональные темы или просто попить пива. Я уверен, что в Microsoft решили уделить отдельное внимание развитию оконных функций в SQL Server именно под влиянием сообщества, и не последнюю роль в этом сыграли наши MVP. Очень приятно наблюдать за тем, как совместные усилия выливаются в нечто большее.

Наконец, я хотел бы сказать спасибо своим студентам. Я обожаю преподавать SQL, это моя страсть, и я благодарен вам за возможность ее удовлетворять. А ваши бесконечные вопросы позволяют мне двигаться дальше, обретая новые знания.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте *www.dmkpress.com*, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу *dmkpress@gmail.com*; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу *http://dmkpress.com/authors/publish_book/* или напишите в издательство по адресу *dmkpress@gmail.com*.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу *dmkpress@gmail.com*. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Pearson Education очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты *dmkpress@gmail.com*.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Глава 1

Работа с окнами в SQL

Оконные функции (window function) способны помочь в решении огромного спектра задач посредством более простого, интуитивного и эффективного подхода к расчетам в рамках наборов данных. Оконные функции представляют собой вычисления, применяемые к набору строк, определенному при помощи инструкции *OVER*. Главным образом такие функции используются для проведения аналитики данных, включая вычисление *накопительных или нарастающих итогов* (running totals), *скользящих средних* (moving average), а также определение *пропусков* (gap), *островов* (island), *интервалов* (interval) и других показателей в ваших данных. Оконные функции базируются на продвинутой концепции, выраженной в стандарте SQL ISO/IEC и называемой *работой с окнами* (windowing). Идея, лежащая в основе этой концепции, позволяет выполнять вычисления применительно к наборам строк (окнам) и возвращать одиночные значения.

С момента появления оконных функций в SQL Server 2005 эта концепция получила существенное развитие в следующих версиях SQL Server и SQL Azure Database. Вскоре мы посмотрим внимательнее на достигнутый за это время прогресс. В актуальных на данный момент версиях продуктов все еще не реализована часть стандартного функционала этой технологии, но за последние годы были сделаны большие шаги в этом направлении. В данной книге мы будем рассматривать как функционал, реализованный в SQL Server, так и возможности, до сих пор не получившие развития в этой среде. Впервые упоминая о тех или иных средствах, я буду отдельно говорить об их поддержке в SQL Server.

С момента появления оконных функций в SQL Server я стал все чаще применять их при решении самых разнообразных задач. Я до сих пор некоторые свои давние решения, опирающиеся на традиционные конструкции языка SQL, переписываю под поддержку оконных функций. И результат в плане эффективности и простоты чаще всего оправдывает приложенные усилия. Сейчас дошло до того, что практически в каждом моем решении так или иначе присутствуют оконные функции. Кроме того, в наши дни стандарт SQL и системы управления базами данных (СУБД), а с ними и вся индустрия данных в целом двигаются в сторону аналитических решений, и оконные функции являются одним из важнейших строительных элементов на этом пути. Упор на

средства оперативной обработки транзакций (online transaction processing – OLTP), который делался в 90-е, там и остался. Лично мне видится развитие технологии SQL в ближайшем будущем именно в сторону более эффективного использования оконных функций, а значит, время, потраченное на их изучение, не пройдет для вас даром.

В книге, которую вы держите в руках, я постарался наиболее полно и подробно рассказать обо всех аспектах применения оконных функций, их оптимизации и использовании в своих решениях. В этой главе мы начнем знакомство с этими функциями и обсудим следующие моменты:

- истоки оконных функций;
- беглый взгляд на решения, использующие оконные функции;
- описание основных элементов, входящих в состав оконных функций;
- обзор инструкций, поддерживающих использование оконных функций;
- разбор стандартного решения с повторным использованием определения окна.

Эволюция оконных функций

Как я уже говорил, с момента своего появления в SQL Server оконные функции проделали уже огромный путь, а в будущем их ждет еще большее развитие. На рис. 1.1 показаны все основные вехи эволюции оконных функций в SQL Server, включая возможности, которые, как мы все надеемся, будут реализованы в ближайшем будущем.

Рис. 1.1 Процесс эволюции оконных функций в SQL Server

На данном этапе многие из перечисленных возможностей могут вам ровным счетом ни о чем не говорить. По большей части этот графический экс-

курс в прошлое был сделан в качестве исторической справки. Гарантирую, что после прочтения книги все перечисленные здесь термины будут вам хорошо знакомы. А сейчас давайте кратко пройдемся по перечисленным на рис. 1.1 этапам:

- в версии SQL Server 2005 появилась поддержка агрегатных оконных функций, но без возможности задавать границы окна, а также были реализованы ранжирующие функции (*ROW_NUMBER*, *RANK*, *DENSE_RANK* и *NTILE*);
- в версиях SQL Server 2008 и 2008 R2 оконные функции затронуты не были;
- в SQL Server 2012 появилась поддержка агрегатных оконных функций с возможностью задавать границы окна, функций смещения (*LAG*, *LEAD*, *FIRST_VALUE* и *LAST_VALUE*), а также статистических или аналитических функций (*PERCENT_RANK*, *CUME_DIST*, *PERCENTILE_CONT* и *PERCENTILE_DISC*);
- в версии SQL Server 2014 оконные функции остались без изменений;
- в SQL Server 2016 оконные функции были оптимизированы за счет появления нового оператора пакетной обработки *Window Aggregate*. Однако, чтобы воспользоваться всеми его преимуществами, хотя бы для одной из участвующих в запросе таблиц должен был быть создан индекс *columnstore*;
- в версии в SQL Server 2017 появилась поддержка первой функции для работы с упорядоченными наборами – *STRING_AGG*, позволяющей конкатенировать строки;
- в SQL Server 2019 был реализован пакетный режим для данных *rowstore*, что позволило оставить в прошлом требование наличия индекса *columnstore* для участвующих в запросе таблиц;
- несколько важных возможностей, связанных с оконными функциями, на данный момент еще не реализованы в SQL Server. Сюда можно отнести создание границ окна с помощью инструкции RANGE с указанием параметра INTERVAL, технологию распознавания шаблонов в строках, вложенные оконные функции, опции *RESPECT | IGNORE NULLS*, инструкцию *WINDOW*, служащую для повторного использования определения окна, прочие функции для работы с упорядоченными наборами и многое другое.

Основы оконных функций

Перед изучением оконных функций вам будет полезно узнать, что послужило поводом для их появления. Именно об этом мы будем говорить в этом разделе. Мы опишем разницу между подходами, основанными на работе с наборами данных и курсорами/итерациями, которые применяются в запросах, и покажем, как оконные функции позволяют перебросить мостик между ними. Наконец, мы обсудим недостатки альтернативных методов и расскажем, почему применение оконных функций в подавляющем большинстве случаев будет лучшим выбором.

Описание оконных функций

Оконная функция (window function) – это функция, применяемая к набору строк. Термином *окно* (window) в стандарте SQL описывается контекст, в котором выполняется функция. Спецификация создаваемого окна задается в SQL при помощи инструкции *OVER*. Рассмотрим в качестве примера следующий простейший запрос:

```
USE TSQLV5;
SELECT orderid, orderdate, val,
RANK() OVER(ORDER BY val DESC) AS rnk
FROM Sales.OrderValues
ORDER BY rnk;
```

О том, где скачать базу данных TSQLV5 и сопроводительные файлы, читайте в вводной главе.

Ниже представлен сокращенный вывод этого запроса:

```
orderid  orderdate  val       rnk
-------- ---------- --------- ----
10865    2019-02-02 16387.50  1
10981    2019-03-27 15810.00  2
11030    2019-04-17 12615.05  3
10889    2019-02-16 11380.00  4
10417    2018-01-16 11188.40  5
10817    2019-01-06 10952.85  6
10897    2019-02-19 10835.24  7
10479    2018-03-19 10495.60  8
10540    2018-05-19 10191.70  9
10691    2018-10-03 10164.80  10
...
```

Оконной функцией в приведенном выше примере является функция *RANK*. Для выполнения ранжирования нам необходимо выполнить упорядочивание. В данном случае мы применили сортировку по полю *val* в *убывающем* (descending) порядке. Эта функция вычисляет порядковый номер для строки в рамках заданного набора данных с учетом указанного порядка сортировки. При использовании сортировки по убыванию, как в нашем примере, ранг вычисляется путем добавления единицы к количеству строк в нашем наборе данных, в которых сортируемое значение больше, чем в текущей строке. Возьмите, к примеру, строку с рангом 5 из нашего запроса. Значение 5 получилось путем прибавления единицы к количеству строк, в которых в столбце *val* находится сумма, превышающая сумму в нашей текущей строке (11 188,40).

Характеристики набора данных, к которому относится наша строка, включая порядок упорядочивания и другие параметры, если они присутствуют, описываются при помощи ключевого слова *OVER*. Если в этой инструкции не указать границы окна, как в нашем примере, набором данных окна будет считаться весь результирующий набор запроса.

Если быть точными, окно определяется как набор строк, или *отношение* (relation), переданное на вход фазы *логической обработки запроса* (logical query processing), в которой представлена оконная функция. Однако такое определение на данном этапе мало что вам скажет. Так что с целью упрощения можно считать, что речь идет о результирующем наборе запроса. Более подробное описание этого процесса будет дано далее.

Гораздо важнее отметить, что в концептуальном смысле инструкция *OVER* определяет окно для функции относительно текущей строки. И это справедливо для всех строк в итоговом наборе данных. Иными словами, для каждой строки инструкция *OVER* определяет окно, независимое от окон, определенных для других строк. На самом деле это очень глубокая идея, понимание которой может прийти не сразу. Но когда вы осознаете всю полноту этого утверждения, вы сможете сказать, что постигли концепцию оконных функций, всю ее мощь и глубину. Если пока для вас это пустой звук, не отчаивайтесь. Я просто бросил семя в почву, ростка придется подождать.

Первое упоминание об оконных функциях в стандарте SQL появилось в расширенном документе к SQL:1999 в разделе «Функции OLAP». С тех пор каждая версия стандарта – SQL:2003, SQL:2008, SQL:2011 и SQL:2016 – расширяла возможности оконных функций, а в последней версии поддержка этой технологии достигла небывалого уровня. Кроме того, оконные функции приобрели такой аналитический механизм, как распознавание шаблонов в строках, что говорит о намерениях рабочей группы, ответственной за стандарт SQL, продолжать развивать это направление.

Вы можете купить официальные документы стандарта SQL от ISO или ANSI. К примеру, базовый документ от ISO по стандарту SQL:2016, описывающий все ключевые конструкции языка, можно приобрести по адресу *https://www.iso.org/standard/63556.html*.

Стандарт SQL поддерживает несколько типов оконных функций: агрегатные, ранжирующие, аналитические или статистические (функции распределения) и функции смещения. Но помните, что работа с окнами в SQL представляет собой целую концепцию, так что в будущих версиях стандарта мы можем увидеть и новые типы оконных функций.

Агрегатные оконные функции (aggregate window functions) – это все те же функции агрегирования, с которыми вы давно и хорошо знакомы (*SUM*, *COUNT*, *MIN*, *MAX* и другие), но которые привыкли применять в контексте запросов с группировками. Агрегатные функции по определению работают с наборами данных, будь то наборы, определенные запросами с группировками или спецификацией окна.

К *ранжирующим функциям* (ranking functions) относятся следующие: *RANK*, *DENSE_RANK*, *ROW_NUMBER* и *NTILE*. При этом первые и последние две функции относятся в стандарте к разным категориям, и позже мы выясним причину этого. Я предпочитаю для простоты относить все четыре функции к одной категории по примеру официальной документации к SQL Server.

Статистические функции (statistical functions), или функции распределения, включают в себя функции *PERCENTILE_CONT*, *PERCENTILE_DISC*, *PERCENT_RANK* и *CUME_DIST*. Эти функции предназначены для расчета статистических показателей, таких как процентили, процентные ранги и накопительные распределения.

К *функциям смещения* (offset functions) относятся функции *LAG*, *LEAD*, *FIRST_VALUE*, *LAST_VALUE* и *NTH_VALUE*. SQL Server поддерживает первые четыре из этого списка. Функция *NTH_VALUE* в версии SQL Server 2019 не реализована.

В главе 2 мы подробно поговорим обо всех перечисленных здесь функциях.

Новые идеи, устройства или инструменты, даже если они очевидно превосходят своих предшественников в простоте использования, всегда являются определенным барьером. Все новое на первый взгляд кажется сложным. Так что если оконные функции для вас в новинку и вы ищете мотивацию в их освоении, я поделюсь несколькими наблюдениями из собственного опыта:

- оконные функции помогают в решении огромного спектра задач. Я даже не знаю, как еще лучше это акцентировать. Как я уже упоминал ранее, в настоящее время я применяю оконные функции практически во всех своих решениях. Заключительная, шестая глава этой книги будет посвящена практическим примерам использования этих функций. А пока я лишь перечислю типы задач, с которыми оконные функции справляются легко и непринужденно:
  - постраничный вывод;
  - удаление дубликатов;
  - возвращение верхних *n* строк по группам;
  - вычисление накопительных итогов;
  - выполнение операций над интервалами, таких как объединение интервалов и вычисление максимального количества пересекающихся интервалов;
  - определение пропусков и островов;
  - вычисление процентилей;
  - вычисление моды распределения;
  - сортировка иерархий;
  - сведение данных;
  - определение новизны данных;
- я пишу запросы на языке SQL вот уже почти три десятилетия и последние несколько лет очень активно использую оконные функции. Могу сказать, что, несмотря на время, которое потребовалось на доскональное освоение этой концепции, сейчас применение оконных функций при решении самых разнообразных задач кажется мне гораздо более эффективным и интуитивным по сравнению с традиционными методами;
- оконные функции поддаются оптимизации, и далее в этой книге мы поговорим об этом более подробно.

Декларативный язык и оптимизация

Вас может удивить то, что в случае использования декларативного языка SQL, в котором вы логически объявляете свой запрос, а не описываете, как именно должен быть получен результат, две разновидности одного и того же запроса (к примеру, с использованием оконных функций и без) могут давать разную производительность. Почему движок SQL Server, использующий диалект языка T-SQL, не всегда способен понять, что две записи означают одно и то же, а значит, должны приводить к созданию одинаковых планов выполнения?

На то есть свои причины. Во-первых, оптимизатор SQL Server не идеален. Я не хочу показаться неблагодарным, ведь оптимизатор запросов, использующийся в SQL Server, представляет собой настоящий шедевр искусства создания программного обеспечения. Но факт в том, что даже в нем не реализованы все без исключения правила оптимизации запросов. Во-вторых, движок ограничен во времени, которое отводится на оптимизацию. Иначе могло бы произойти так, что на саму оптимизацию тратилось бы больше времени, чем можно было бы сэкономить за счет нее. Для любого запроса можно придумать бессчетное количество планов выполнения, и на проверку каждого просто не хватит времени. На основании факторов, таких как размер таблиц, участвующих в запросе, SQL Server вычисляет два значения: *стоимость* (cost), которая может считаться приемлемой для данного запроса, и максимальное количество времени, которое может быть затрачено на выполнение оптимизации. По достижении любого из этих значений оптимизация прекращается, и SQL Server использует лучший из найденных на этот момент времени планов выполнения запроса.

При этом оконные функции, о которых мы будем говорить в нашей книге, зачастую могут быть оптимизированы гораздо более эффективно по сравнению с традиционными конструкциями при достижении одной и той же цели.

Здесь важно понимать, что вы должны сделать абсолютно осознанный выбор, переключаясь на использование оконных функций, поскольку это совершенно иная концепция, которую нужно использовать во благо. К работе с окнами в SQL привыкнуть бывает непросто. Но по прохождении периода адаптации вы осознаете всю легкость и интуитивность этой технологии. Вспомните, как долго вы привыкали к современным гаджетам, без которых сегодня просто не мыслите жизни.

Программирование на основе наборов данных и курсоров/итераций

Подходы, применяемые в диалекте T-SQL к решению задач, можно условно разделить на *декларативный* (declarative), использующий в своей основе наборы данных, и *итеративный* (iterative), базирующийся на курсорах. При этом в среде разработчиков T-SQL есть полное единодушие в пользу первого под-

хода. Но в то же время существует масса решений, в которых по-прежнему активно применяются итерации. Здесь возникает ряд интересных вопросов. Почему вариант с использованием наборов данных является более предпочтительным? И если это так, почему многие разработчики применяют итеративный подход? Что мешает им переключиться на рекомендованный декларативный стиль?

Чтобы докопаться до сути поставленных вопросов, сначала необходимо разобраться в основах T-SQL и понять, что из себя представляет подход, базирующийся на наборах данных. В процессе вы осознаете, что декларативные принципы для многих являются далеко не самыми интуитивными, тогда как итеративные кажутся простыми и понятными. Просто так устроен наш мозг, и мы позже поговорим об этом подробнее. Пропасть между программированием на основе наборов данных и курсоров просто огромна. Сократить ее можно, но сделать это очень непросто. И здесь на помощь приходят оконные функции. Лично я рассматриваю их как прочный мост между двумя концепциями, позволяющий обеспечить более плавный и легкий переход к мышлению на основе наборов данных.

Но для начала нужно ясно проговорить, что из себя представляет декларативный подход, базирующийся на выполнении операций с наборами данных, применительно к запросам на языке T-SQL. *T-SQL* – это диалект стандарта SQL (обоих стандартов – ISO/IEC и ANSI). SQL основывается (вернее, пытается это делать) на *реляционной модели* (relational model). Реляционная модель представляет собой математическую модель для управления и манипулирования данными, которая была сформулирована и предложена Эдгаром Франком Коддом (E. F. Codd) в конце 1960-х. Реляционная модель базируется на двух математических принципах: *теории множеств* (set theory) и *логике предикатов* (predicate logic). Многие аспекты вычислений были разработаны на основе чистой интуиции, и они до сих продолжают меняться достаточно динамично – до такой степени, что порой кажется, что ты носишься за собственным хвостом. Реляционная модель – это остров в этом мире вычислений, поскольку она основывается на гораздо более строгих математических постулатах. Многие считают, что с математикой не поспоришь. И, будучи основанной исключительно на математических принципах, реляционная модель получила славу очень логичной и надежной субстанции. Эта модель продолжает развиваться, но не так стремительно, как другие аспекты вычислений. На протяжении последних нескольких десятилетий реляционная модель прочно удерживает свои позиции и по-прежнему является основой всех современных лидирующих *систем управления реляционными базами данных* (relational database management system – RDBMS).

SQL стал своеобразной попыткой создать язык на основе реляционной модели. Но SQL не идеален и в некоторых аспектах отклоняется от принципов реляционной модели. В то же время он предоставляет достаточный выбор инструментов, чтобы при должном понимании концепции реляционной модели можно было использовать язык SQL на основе реляционных принципов. По большому счету SQL удерживает безоговорочное первенство и фактически является единственным языком данных.

Однако, как я уже упоминал, мыслить категориями реляционной модели для многих очень непривычно и даже противоестественно. Основной блок со-

держится в различиях между подходом, основывающимся на наборах данных, и итеративным подходом. И особенно трудно приходится тем, у кого есть за плечами опыт программирования на процедурных языках, в которых привычным способом чтения данных из файлов являются итерации, как показано на примере псевдокода ниже:

```
открываем файл
извлекаем первую запись
пока не достигнут конец файла
начало
    обрабатываем запись
    извлекаем следующую запись
конец
```

Данные в файлах, или, если быть более точными, в *индексно-последовательном методе доступа* (indexed sequential access method – ISAM), хранятся в заданном порядке, и в процессе чтения информация извлекается в точности в том же порядке, в котором была записана. Кроме того, данные извлекаются порциями – по одной за раз. Таким образом, наш мозг запрограммирован воспринимать работу с данными именно так: порядок строго задан, получаем по одной записи. Это аналогично тому, как в T-SQL работают *курсоры* (cursor), и программисты с опытом работы с процедурными языками склонны использовать именно их или другие техники выполнения итераций в SQL, воспринимая это как естественное продолжение того, что они уже знают.

Реляционный подход, основанный на наборах или множествах данных, проповедует совершенно иные принципы работы с информацией. Чтобы проникнуть в самую их суть, давайте для начала посмотрим на определение *множества* (set), данное основателем теории множеств Георгом Кантором (Georg Cantor):

> Под «множеством» мы подразумеваем любое объединение M в одно целое, состоящее из определенных различающихся объектов m (называемых «элементами» M) в области наших мыслей и восприятий.
>
> –*Джозеф Даубен* (Joseph W. Dauben), *Георг Кантор* (Georg Cantor) (Princeton University Press, 1990)

В этом определении множества столько всего, что можно было бы не одну страницу посвятить его объяснению. Но я предпочел сфокусировать внимание на двух ключевых аспектах, один из которых присутствует в определении явно, а второй подразумевается:

- **целое.** Обратите внимание на использование в определении множества термина *целое*. Множество должно восприниматься и управляться как единое *целое*. И вы должны работать с ним как с общностью данных, а не разрозненным набором элементов. В процессе выполнения итеративного перебора это правило нарушается, поскольку записи из файла или курсора обрабатываются по одной. Таблица в SQL представляет (хотя и не совсем успешно) отношение в реляционной модели. *Отношением* (relation) называется общность похожих элементов, т. е. обладающих одинаковым