

Ростислав Михеев

MS SQL Server 2005 для администраторов

СПЕЦИАЛЬНЫЙ КУРС

Санкт-Петербург
«БХВ-Петербург»
2006

УДК 681.3.06

ББК 32.973.26-018.2

M69

Михеев Р. Н.

М69 MS SQL Server 2005 для администраторов. — СПб.: БХВ-Петербург, 2006. — 544 с.: ил.

ISBN 5-94157-796-6

Рассмотрены вопросы администрирования СУБД SQL Server 2005. Впервые на русском языке подробно рассматривается информация по использованию среды SSIS для передачи и преобразования данных, применение объектных моделей SMO, SQL-DMO и WMI для автоматизации администрирования, работа со встроенными средствами шифрования данных SQL Server 2005. В каждой главе приведена информация об отличиях новой версии SQL Server от предыдущей, а для начинающих администраторов предусмотрены задания для самостоятельной работы с подробными ответами. Книгу можно использовать и в качестве учебного пособия при проведении обучения по администрированию SQL Server 2005.

Для программистов и администраторов баз данных

УДК 681.3.06

ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	Екатерина Кондукова
Зам. главного редактора	Игорь Шишигин
Зав. редакцией	Григорий Добин
Редактор	Нина Седых
Компьютерная верстка	Ольги Сергиенко
Корректор	Зинаида Дмитриева
Дизайн серии	Инны Тачиной
Оформление обложки	Елены Беляевой
Зав. производством	Николай Тверских

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 07.07.06.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 43,86.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

Оглавление

Предисловие.....	1
Глава 1. Обзор новых возможностей SQL Server 2005	3
1.1. Новые возможности SQL Server 2005 для администраторов	3
1.1.1. Изменения в наборе административных средств.....	3
1.1.2. Новые возможности для оптимизации производительности	4
1.1.3. Новые возможности системы безопасности.....	7
1.1.4. Новые возможности для обеспечения отказоустойчивости.....	8
1.1.5. Новые возможности системы репликации	10
1.1.6. Новые объектные модели для автоматизации администрирования	12
1.2. Новые возможности SQL Server 2005 для программистов.....	12
1.3. Новые возможности SQL Server Integration Services (DTSX).....	20
Глава 2. Планирование и установка SQL Server 2005	23
2.1. Планирование установки SQL Server 2005	23
2.1.1. Оценка архитектуры приложения на основе SQL Server 2005.....	23
2.1.2. Выбор оборудования	26
2.1.3. Выбор операционной системы и ее компонентов	31
2.1.4. Выбор редакции SQL Server 2005.....	32
2.2. Установка SQL Server 2005	34
2.2.1. Несколько слов об установке	34
2.2.2. Начало установки. Выбор набора компонентов	35
2.2.3. Работа с именованными экземплярами.....	38
2.2.4. Выбор учетной записи для служб SQL Server.....	40
2.2.5. Выбор режима аутентификации SQL Server 2005	43
2.2.6. Выбор кодировки и порядка сортировки.....	46
2.2.7. Остальные параметры установки.....	48
2.3. Автоматизированная и удаленная установка	49
2.4. Обновление предыдущих версий SQL Server и миграция с Microsoft Access.....	50
2.5. Проверка установки и выполнение послеустановочных задач	53
2.5.1. Проверка результатов установки	53
2.5.2. Настройка сетевых библиотек для доступа клиентов	54
2.5.3. Другие послеустановочные задачи.....	57
Задание для самостоятельной работы 2.1. Установка именованного экземпляра SQL Server 2005.....	58

Глава 3. Средства администрирования SQL Server 2005	61
3.1. SQL Server Management Studio	61
3.1.1. Что такое SQL Server Management Studio	61
3.1.2. Окно <i>Registered Servers</i>	62
3.1.3. Окно <i>Object Explorer</i>	64
3.1.4. Окно <i>Document</i>	66
3.1.5. Окно <i>Solution Explorer</i>	66
3.1.6. Другие окна SQL Server Management Studio.....	68
3.1.7. Приемы работы со скриптами.....	70
3.2. Business Intelligence Development Studio	74
3.3. SQL Server Configuration Manager.....	75
3.3.1. Что такое SQL Server Configuration Manager.....	75
3.3.2. Службы SQL Server 2005	76
3.3.3. Настройка серверных сетевых библиотек средствами SQL Server Configuration Manager.....	78
3.3.4. Настройка клиентских сетевых библиотек средствами SQL Server Configuration Manager. SQL Native Client.....	79
3.4. SQLCmd	81
3.4.1. Применение SQLCmd.....	81
3.4.2. Специальный режим подключения Dedicated Administrator Connection.....	83
3.5. SQL Server Surface Area Configuration	84
3.6. SQL Server Profiler.....	86
3.7. Database Engine Tuning Advisor.....	88
3.8. Другие графические утилиты SQL Server 2005	90
3.9. Другие консольные утилиты SQL Server 2005	92
Задание для самостоятельной работы 3.1. Работа со скриптами в SQL Server Management Studio и SQLCmd.....	94
Задание для самостоятельной работы 3.2. Работа с серверными сетевыми библиотеками и псевдонимами	95

Глава 4. Создание баз данных и настройка параметров.....	97
4.1. Служебные и учебные базы данных SQL Server 2005	97
4.2. Создание пользовательских баз данных.....	99
4.2.1. Создание базы данных из SQL Server Management Studio.....	99
4.2.2. Создание базы данных при помощи команды <i>CREATE DATABASE</i>	100
4.2.3. Альтернативные способы создания базы данных.....	102
4.3. Файлы баз данных и журналов транзакций.....	104
4.4. Применение файловых групп	108
4.5. Режим восстановления базы данных	110
4.6. Режимы работы базы данных	112
4.7. Другие параметры базы данных.....	114
4.8. Расширенные свойства баз данных.....	119
4.9. Выполнение некоторых служебных операций с базами данных	120
4.9.1. Увеличение размера базы данных	121
4.9.2. Уменьшение размера базы данных.....	121

4.9.3. Перенос файлов базы данных	123
4.9.4. Переименование базы данных	124
4.9.5. Изменение владельца базы данных	124
4.9.6. Удаление базы данных	124
4.9.7. Проверка целостности базы данных.....	125
Задание для самостоятельной работы 4.1. Перенос баз данных с SQL Server 2000 на SQL Server 2005	125
Глава 5. Безопасность SQL Server 2005.....	129
5.1. Терминология и основы системы безопасности SQL Server 2005	130
5.2. Логины SQL Server 2005.....	131
5.2.1. Выбор типа логина	131
5.2.2. Создание логина и настройка его параметров	134
5.2.3. Режимы аутентификации. Аудит попыток входа	138
5.2.4. Логины, создаваемые по умолчанию	140
5.2.5. Серверные роли. Разрешения на уровне сервера.....	141
5.3. Пользователи баз данных	144
5.3.1. Что такое пользователь базы данных	144
5.3.2. Пользователь и схема	145
5.3.3. Создание, изменение и удаление пользователей базы данных	146
5.3.4. Встроенные пользователи базы данных	147
5.3.5. Роли баз данных.....	148
5.3.6. Предоставление прав на объекты в базе данных	150
5.4. Роли приложений	153
5.5. Изменение контекста выполнения. Выражение <i>EXECUTE AS</i>	155
5.6. Применение сертификатов и шифрование данных в SQL Server 2005.....	158
5.6.1. Основы применения сертификатов и шифрования данных	158
5.6.2. Защита сетевого трафика SQL Server 2005.....	160
5.6.3. Принципы шифрования информации в базах данных SQL Server 2005	168
5.6.4. Создание сертификатов и ключей и применение криптографических функций.....	170
Задание для самостоятельной работы 5.1. Назначение прав на объекты SQL Server 2005 и изменение контекста выполнения.....	173
Задание для самостоятельной работы 5.2. Шифрование информации в таблицах баз данных.....	176

Глава 6. Резервное копирование и восстановление баз данных SQL Server 2005

179

6.1. Основы резервного копирования SQL Server 2005	179
6.2. Планирование резервного копирования.....	180
6.2.1. Лента или жесткий диск?	180
6.2.2. Логические устройства или явно указанный путь?	182
6.2.3. Типы резервного копирования.....	183
6.2.4. Расписание резервного копирования	186

6.3. Проведение резервного копирования	187
6.3.1. Средства для выполнения резервного копирования.....	187
6.3.2. Параметры резервного копирования.....	188
6.3.3. Получение информации о резервном копировании и создание отчетов	194
6.4. Основы восстановления баз данных	195
6.5. Подготовка к восстановлению	197
6.6. Проведение восстановления	198
6.7. Специальные ситуации восстановления.....	203
6.7.1. Восстановление базы данных в оперативном режиме	203
6.7.2. Восстановление отдельных страниц базы данных	204
6.7.3. Восстановление системных баз данных.....	205
Задание для самостоятельной работы 6.1. Резервное копирование и восстановление базы данных	207

Глава 7. Средства обеспечения отказоустойчивости SQL Server 2005..... 209

7.1. Работа SQL Server 2005 в кластере.....	209
7.1.1. Преимущества кластеров	209
7.1.2. Терминология и варианты конфигурации кластера	210
7.1.3. Установка SQL Server 2005 в кластер	212
7.2. Автоматическая доставка журналов	213
7.2.1. Что такое "автоматическая доставка журналов".....	213
7.2.2. Терминология доставки журналов	214
7.2.3. Настройка доставки журналов	215
7.2.4. Мониторинг доставки журналов.....	221
7.2.5. Действия в случае сбоя основного сервера	224
7.2.6. Как отменить доставку журналов	226
7.3. Зеркальное отображение баз данных	227
7.3.1. Что такое "зеркальное отображение баз данных".....	227
7.3.2. Терминология зеркального отображения баз данных	229
7.3.3. Настройка зеркального отображения	229
7.3.4. Мониторинг зеркального отображения	232
7.3.5. Смена ролей серверов	233
7.3.6. Приостановка и отмена зеркального отображения.....	235
Задание для самостоятельной работы 7.1. Настройка доставки журналов	236

Глава 8. Автоматизация администрирования SQL Server 2005..... 241

8.1. Автоматизация административных операций средствами SQL Server Agent	242
8.1.1. Что такое SQL Server Agent.....	242
8.1.2. Параметры настройки SQL Server Agent	243
8.1.3. Работа с заданиями SQL Server Agent.....	248
8.1.4. Безопасность при выполнении заданий	257
8.1.5. Просмотр истории выполнения заданий.....	260
8.1.6. Мультисерверные задания.....	261
8.1.7. Работа с предупреждениями.....	264
8.1.8. Работа с операторами.....	270

8.2. Настройка электронной почты в SQL Server 2005	271
8.2.1. Обзор возможностей SQL Server 2005 для работы с электронной почтой....	271
8.2.2. Работа с Database Mail.....	273
8.2.3. Работа с SQMmail.....	278
8.2.4. Альтернативные способы работы с электронной почтой SQL Server и SQL Server Agent.....	281
8.3. Планы обслуживания баз данных	284
Задание для самостоятельной работы 8.1. Применение заданий, предупреждений и операторов	288

Глава 9. Выполнение административных операций

при помощи объектных моделей SMO, SQL-DMO и WMI 293

9.1. Применение скриптов для выполнения административных операций.....	293
9.2. Объектная модель SMO	295
9.2.1. Обзор объектной модели SMO	295
9.2.2. Общие приемы работы с объектами SMO.....	298
9.2.3. Объект <i>SMO.Server</i>	300
9.2.4. Объект <i>SMO.Database</i>	305
9.3. Объектная модель SQL-DMO.....	306
9.3.1. Обзор объектной модели SQL-DMO.....	306
9.3.2. Объект <i>SQLDMO.Application</i>	308
9.3.3. Объект <i>SQLDMO.SQLServer2</i>	308
9.3.4. Объект <i>SQLDMO.Database2</i>	311
9.4. WMI и SQL Server 2005	312
9.4.1. Что такое WMI.....	312
9.4.2. WMI-поставщики для работы с SQL Server	315
9.4.3. Программные средства для работы с WMI	316
9.4.4. Подключение к службе WMI	319
9.4.5. Язык WQL: подключение к объектам WMI	320
9.4.6. Работа с событиями в WMI	323
9.4.7. Объекты WMI Provider for Configuration Management	326
9.4.8. Работа с WMI Provider for Server Events	328
Задание для самостоятельной работы 9.1. Применение объектной модели SMO	330
Задание для самостоятельной работы 9.2. Применение объектной модели SQL-DMO.....	331
Задание для самостоятельной работы 9.3. Работа с WMI Provider for Configuration Management	332

Глава 10. Применение SQL Server Integration Services

335

10.1. Зачем нужны SQL Server Integration Services.....	335
10.2. Средства для работы с SSIS	336
10.3. Применение мастера импорта и экспорта данных	339
10.4. Пример работы с SSIS Designer	343
10.5. Менеджеры подключений	349

10.6. Работа с Data Flow Task	352
10.6.1. Что такое Data Flow Task	352
10.6.2. Элементы Data Flow Task	353
10.6.3. Источники и назначения Data Flow Task	354
10.6.4. Преобразования Data Flow Task	356
10.6.5. Пути и логика выполнение Data Flow Task	364
10.7. Script Task и ActiveX Script Task.....	366
10.8. Bulk Insert Task	368
10.9. Execute SQL Task.....	369
10.10. XML Task	370
10.11. Message Queue Task.....	371
10.12. Execute Package Task и Execute DTS 2000 Package Task.....	374
10.13. Transfer Database Task	375
10.14. Другие задачи копирования объектов SQL Server	376
10.15. File System Task и FTP Task	377
10.16. Send Mail Task	377
10.17. Execute Process Task.....	378
10.18. Web Service Task	378
10.19. WMI Data Reader Task и WMI Event Watcher Task	380
10.20. Задачи Analysis Services.....	383
10.21. Контейнеры SSIS.....	383
10.22. Задачи планов обслуживания	386
10.23. Ограничения предшественников.....	387
10.24. Протоколирование выполнения пакетов.....	389
10.25. Работа с конфигурациями.....	391
10.26. Хранение пакетов	393
10.27. Безопасность пакетов SSIS	395
10.28. Запуск пакетов SSIS на выполнение	398
Задание для самостоятельной работы 10.1. Создание пакетов SSIS для переноса данных	401

Глава 11. Мониторинг и оптимизация производительности SQL Server 2005

407

11.1. Введение в мониторинг работы и оптимизацию производительности SQL Server 2005	407
11.2. Мониторинг активности пользователей	408
11.2.1. Применение Activity Monitor.....	408
11.2.2. Другие средства мониторинга активности пользователей	409
11.2.3. Работа с профилявшником	410
Задание для самостоятельной работы 11.1. Сбор информации о запросах, выполняемых приложением	418
11.2.4. Применение триггеров DDL.....	419
11.2.5. Другие средства мониторинга активности пользователей и уведомления о событиях	421
11.3. Журналы SQL Server 2005.....	422

11.4. Мониторинг производительности SQL Server 2005	423
11.4.1. Основы мониторинга производительности	423
11.4.2. Терминология и общие принципы мониторинга производительности.....	424
11.4.3. Средства для мониторинга и анализа производительности	426
11.4.4. Нагрузочное тестирование	427
11.4.5. Приемы работы с Системным монитором	430
11.4.6. Основы работы с объектами и счетчиками	433
11.4.7. Счетчики для анализа загрузки процессора	434
11.4.8. Счетчики для анализа загрузки оперативной памяти.....	437
11.4.9. Счетчики для анализа производительности дисковой подсистемы	439
11.4.10. Счетчики для анализа производительности сетевой подсистемы	441
11.4.11. Объекты Системного монитора для мониторинга работы SQL Server 2005	444
Задание для самостоятельной работы 11.2. Приемы работы с Системным монитором	449
11.5. Оптимизация работы SQL Server	452
11.5.1. Общие вопросы оптимизации работы SQL Server	452
11.5.2. Оптимизация операционной системы для работы с SQL Server 2005	453
11.5.3. Общая оптимизация работы SQL Server 2005 с помощью Best Practices Analyzer	457
11.5.4. Оптимизация подключений к SQL Server 2005	459
11.5.5. Оптимизация системы индексов.....	462
Задание для самостоятельной работы 11.3. Оптимизация системы индексов	466
11.5.6. Дефрагментация индексов и таблиц.....	467
Задание для самостоятельной работы 11.4. Дефрагментация таблиц и индексов	474
11.5.7. Работа с блокировками	475
Задание для самостоятельной работы 11.5. Управление уровнем блокировок	479
11.5.8. Оптимизация запросов	480
Глава 12. Репликация в SQL Server 2005.....	487
12.1. Зачем нужна репликация	487
12.2. Новые возможности репликации SQL Server 2005	489
12.3. Терминология системы репликации	490
12.4. Типы репликации	492
12.5. Подготовка к настройке репликации	494
12.6. Настройка репликации.....	496
12.7. Средства администрирования и мониторинга репликации	501
12.7.1. Средства администрирования репликации.....	501
12.7.2. Применение Replication Monitor	504
12.7.3. Другие средства мониторинга репликации	507
Задание для самостоятельной работы 12.1. Настройка одноранговой репликации ...	508
Предметный указатель	513

Предисловие

7 ноября 2005 г. корпорация Microsoft представила новую версию своей флагманской системы управления базами данных — SQL Server 2005. С момента выхода предыдущей версии SQL Server прошло пять лет. Изменений в SQL Server 2005 по сравнению с SQL Server 2000 — множество. Появились совершенно новые возможности (например, подсистема Service Broker и встроенное шифрование), а некоторые подсистемы были практически созданы заново (DTS/SSIS). Даже специалистам с большим опытом работы с SQL Server 2000 многое приходится осваивать заново.

Автор этой книги — сертифицированный преподаватель Microsoft, который читает курсы по SQL Server уже в течение 8 лет (начиная с версии 6.5). Параллельно с преподаванием автору часто приходилось решать проблемы, связанные с SQL Server, на самых разных предприятиях. Много раз про свои найденные решения рассказывали слушатели. Таким образом, постепенно накапливался опыт решения различных проблем, множился список не всегда очевидных, но эффективных приемов.

С другой стороны, автору стал очевиден недостаток существующих источников информации для администраторов баз данных по SQL Server 2005.

Главный источник информации — это, конечно, встроенная документация по SQL Server 2005 (Books Online). Но на момент написания этой книги русской версии Books Online еще не было, что для многих специалистов, особенно в регионах, является большой проблемой.

Другой русскоязычной литературы по SQL Server 2005 на момент написания этой книги также практически не встречалось.

Еще один источник информации по SQL Server 2005 — это официальные курсы Microsoft (MOC — Microsoft Official Curriculum). Но методических материалов к ним на русском языке нет. Кроме того, у курсов Microsoft есть существенный недостаток — их маркетинговая направленность. В методических материалах по курсам активно продвигается информация о преимуществах и возможностях, и гораздо меньше говорится о проблемах SQL Server и его недостатках. Обычно администраторам приходится самостоятельно искать информацию о проблемах, столкнувшись с ними на практике.

Именно поэтому автор и решил написать эту книгу.

Автору хотелось бы сразу отметить некоторые моменты:

- эта книга посвящена только вопросам, связанным с администрированием SQL Server. Вопросы, касающиеся разработки баз данных и приложений SQL Server 2005, будут рассмотрены в другой книге, которая готовится к выходу в издательстве "БХВ-Петербург". Рабочее название книги — "Microsoft SQL Server 2005 для разработчиков";
- автор попытался сделать акцент на том, в какой именно ситуации, при возникновении какой проблемы нужно использовать то или иное средство. При этом некоторые технические подробности, не очень важные для понимания сути проблемы, опускались. Например, не приводится полное описание всех параметров, принимаемых хранимой процедурой. Зная, какую именно хранимую процедуру нужно использовать, описание параметров несложно найти в справке;
- автор постарался поместить в книгу ту информацию, которая, по опыту проведения курсов, очень востребована администраторами, но недостаточно рассматривается в официальных учебных курсах. Например, большое внимание уделяется вопросам, связанным с мониторингом и оптимизацией производительности баз данных SQL Server 2005, оптимизацией системы индексов и блокировок, автоматизацией выполнения административных операций средствами SMO и SQL-DMO, переносом и преобразованием данных средствами DTSX/SSIS, защитой информации в базах данных SQL Server 2005 и т. п.

Эту книгу можно использовать как учебное пособие для самостоятельного обучения или для проведения обучения специалистов. Для каждой главы предусмотрены задания для самостоятельной работы с подробными решениями. Эти решения можно также использовать как пошаговую инструкцию при выполнении различных операций.

Автор ждет вопросы от читателей этой книги и будет рад, если ему удастся помочь. Информация о каких-то неочевидных приемах работы с SQL Server или об ошибках в книге также будет принята с благодарностью. С автором можно связаться по электронной почте rostislavm@askit.ru.

Автор книги работает в учебном центре **Академии специальных курсов по информационным технологиям** (г. Санкт-Петербург). Учебный центр проводит обучение как по SQL Server 2005, так и по многим другим программным продуктам Microsoft и других производителей. Обучение производится как в учебных классах в Санкт-Петербурге, так и на территории предприятий по всей России. Информацию о некоторых учебных курсах можно найти в конце этой книги или на сайте www.askit.ru. Ваши вопросы по обучению направляйте по электронной почте info@askit.ru или по телефону **8 (812) 922 47 60**.



ГЛАВА 1

Обзор новых возможностей SQL Server 2005

В SQL Server 2005 появилось множество новых возможностей по сравнению с предыдущей версией этого программного продукта — SQL Server 2000. Задача этой главы — дать обзор новых возможностей SQL Server 2005 для специалистов, которые работали с более ранними версиями SQL Server. Подробнее большинство этих возможностей будет рассмотрено в следующих главах.

1.1. Новые возможности SQL Server 2005 для администраторов

1.1.1. Изменения в наборе административных средств

Первое, что бросается в глаза при знакомстве с SQL Server 2005, — это кардинальное изменение набора средств администрирования. В SQL Server 2005 вы не найдете ни Enterprise Manager, ни Query Analyzer, ни Service Manager, ни многих других привычных программ. Вместо них появились другие, совершенно новые средства. Конечно, многие возможности унаследованы из предыдущих версий, но добавлено и много нового. Приведем перечень главных средств администрирования SQL Server 2005.

- **SQL Server Management Studio** — главный инструмент администратора. Эта консоль администрирования, которая основана на Visual Studio, соединила в себе возможности Enterprise Manager, Query Analyzer, Analysis Manager, средств администрирования Reporting Services и Notification Services. Про SQL Server Management Studio будет рассказано в разд. 3.1.
- **Business Intelligence Development Studio** — еще одна консоль, основанная на Visual Studio. Она позволяет работать с проектами Analysis Services,

Integration Services (так называется новая версия Data Transformation Services, DTS), Reporting Services и Report Model. Проекты Integration Services и работа с ними средствами Business Intelligence Development Studio будут рассматриваться в гл. 10.

- **SQLCmd** — это утилита командной строки, призванная заменить `isql` и `osql` из предыдущих версий SQL Server. В ней появилось множество новых возможностей (например, Dedicated Administrator Connection — специальный выделенный административный режим подключения). Работа с этой утилитой будет рассматриваться в разд. 3.4.
- **SQL Server Configuration Management** — эта программа объединила и расширила возможности Service Manager, Server Network Utility и Client Network Utility из предыдущих версий SQL Server. Ее применение для настройки служб SQL Server, серверных и клиентских сетевых библиотек будет рассматриваться в разд. 3.3.
- **SQL Server Surface Area Configuration** — программное средство, в котором централизовано управление доступными возможностями SQL Server 2005 (например, из него можно включать и отключать возможность работы со сборками .NET из кода Transact-SQL, объектами для работы с электронной почтой, хранимыми процедурами автоматизации и т. п.). Про работу с этим программным средством будет рассказано в разд. 3.5.

1.1.2. Новые возможности для оптимизации производительности

Одной из основных целей, которые преследовала компания Microsoft при разработке SQL Server 2005, было повышение масштабируемости по сравнению с предыдущими версиями SQL Server. Надо сказать, что эта цель была во многом достигнута. Помимо общей оптимизации работы ядра базы данных, в SQL Server 2005 было реализовано множество новых средств, которые можно использовать для оптимизации производительности:

- **Возможность секционирования (*partitioning*) для таблиц и индексов.** Это очень важное нововведение позволяет производить более гибкое управление производительностью работы сервера, а также повышает удобство администрирования. Например, архивные данные в таблице (которые практически не изменяются, а обращения к ним производятся редко) можно поместить в одном разделе, а текущие данные, с которыми работают пользователи, — в другом. Второй раздел с текущими данными можно поместить на самый быстрый RAID-массив. Также можно производить различные операции по обслуживанию (например, дефрагментацию индексов) только для этого раздела.

- **Уровень изоляции моментальных снимков (*snapshot isolation*).** Этот новый план изоляции транзакций призван решить проблему с блокировками, которая доставляла специалистам немало проблем при работе с предыдущими версиями SQL Server. При использовании этого режима изоляции транзакций запросы, которые обращаются к данным только на чтение, не накладывают блокировки на записи в таблице.
- **Поддержка большего числа экземпляров SQL Server на одном компьютере.** Теперь на одном компьютере можно установить до 50 экземпляров SQL Server 2005 Enterprise Edition. Для других редакций SQL Server 2005 оставлено то же ограничение, что и для SQL Server 2000, — максимум 16 экземпляров SQL Server на компьютере. Про работу с именованными экземплярами будет рассказано в разд. 2.2.3.
- **Немедленная инициализация файлов (*instant file initialization*).** Это средство позволяет не заполнять пустое пространство файлов баз данных при их создании или увеличении двоичными нулями (как это было в предыдущих версиях SQL Server). В результате выполнение таких операций заметно ускоряется. Отметим только, что эта возможность доступна лишь при работе SQL Server 2005 на Windows Server 2003 или Windows XP (при работе под Windows 2000 она недоступна). Подробно про немедленную инициализацию файлов будет рассказываться в разд. 4.3.
- **Возможность отключения индексов.** Эта возможность может пригодиться, например, для диагностики проблем. Кроме того, перестроение отключенного некластеризованного индекса требует намного меньше места на диске. Отключение индекса производится при помощи команды ALTER INDEX ... DISABLE (см. разд. 11.5.6).
- **Ограничение количества процессоров для операций создания, изменения и удаления индексов (параметр MAXDOP).** Операции с индексами для больших таблиц могут оказаться очень ресурсоемкими. В результате работа пользователей может быть затруднена. Чтобы ограничить системные ресурсы, которые отводятся для выполнения операций с индексами, в SQL Server 2005 предусмотрен новый параметр MAXDOP (MAXimum Degree of Parallelism — максимальная степень распараллеливания). При помощи этого параметра можно определить максимальное количество процессоров, которые будут использоваться для выполнения операций с индексами.
- **Отложенное удаление и перестроение больших объектов.** Эта новая возможность автоматически используется SQL Server 2005 при удалении и перестроении таблиц индексов, которые занимают много места в базе данных (больше 128 экстентов, т. е. более 8 Мбайт). При этом в базе данных происходит только логическое удаление (т. е. для пользователя таблица

или индекс будут выглядеть как удаленные в обычном режиме). Физические же операции с экстентами, которые могут потребовать значительного времени, производятся в асинхронном режиме после завершения транзакции.

- **Динамические представления (*dynamic views*)**. Новый тип представлений SQL Server 2005 позволяет получать информацию о различных аспектах работы SQL Server 2005. Например, информацию о текущих подключениях пользователей можно просмотреть при помощи динамического представления `sys.dm_exec_sessions`, а информацию о блокировках — `sys.dm_tran_locks`. Динамические представления будут рассматриваться в разд. 11.2.2.
- **Триггеры DDL (*DDL triggers*)**. Эта новая возможность может использоваться для мониторинга выполнения команд DDL (Data Definition Language — язык изменения данных), т. е. команд, которые создают, изменяют или удаляют объекты в базах данных. Такие триггеры могут использоваться для аудита, дополнительных проверок и т. п. Подробнее про триггеры DDL будет рассказано в разд. 11.2.4.
- **Уведомления о событиях (*event notifications*)**. При помощи этого средства можно отслеживать выполнение команд DDL и события трассировки (те же события SQL Server, которые видны в профилировщике). После настройки уведомлений о событиях уведомления передаются в очередь программного модуля Service Broker в виде файлов XML.
- **Асинхронное обновление статистики (параметр `AUTO_UPDATE_STATISTICS_ASYNC`)**. Этот параметр предназначен для того, чтобы сделать время выполнения запроса более прогнозируемым. По умолчанию (значение параметра `FALSE`) запрос, выполняемый на SQL Server, может инициировать обновление статистики, если выяснится, что статистика, необходимая для выбора правильного плана выполнения этого запроса, устарела. Однако расчет новой статистики займет определенное время, в результате чего общее время выполнения запроса может сильно возрасти. Если мы установим для параметра базы данных `AUTO_UPDATE_STATISTICS_ASYNC` значение `TRUE`, то запрос также может инициировать обновление статистики. Но в этом случае статистика будет обновляться в асинхронном режиме, параллельно с выполнением запроса (при этом допускается, что для запроса будет выбран неоптимальный план).
- **Горячее добавление оперативной памяти (*hot-add memory*)**. Если ваше оборудование поддерживает добавление оперативной памяти "на лету", то SQL Server 2005 сможет использовать такую добавленную память без необходимости перезапуска сервера. Такое решение может оказаться очень удобным для серверов, которые работают круглосуточно и остановка которых может привести к проблемам для пользователей.

- **Динамическое управление памятью при использовании AWE (Address Windowing Extensions — оконное расширение адресации).** Это средство позволяет 32-разрядным компьютерам работать с оперативной памятью размером более 4 Гбайт. В предыдущих версиях размер памяти, которую мог использовать SQL Server при применении AWE, был статическим. В SQL Server 2005 его можно изменять динамически. Про работу с AWE будет рассказано в разд. 11.4.8.

1.1.3. Новые возможности системы безопасности

В систему безопасности SQL Server 2005 были внесены очень большие изменения. Она стала более зрелой и функциональной. Наиболее важные новые возможности представлены далее.

- **Большое количество новых разрешений.** Эти разрешения предусмотрены для самого сервера, для баз данных и для многочисленных объектов в базах данных. Возможности предоставлений разрешений были значительно расширены по сравнению с предыдущими версиями SQL Server. Про разрешения SQL Server подробно будет говориться в гл. 5.
- **Разделение владельцев и схемы (*separation of users and schemas*).** Эта новая и очень удобная возможность позволяет отделить именование и группировку объектов (которая производится при помощи схемы) от владения объектами. Кроме того, схему очень удобно использовать и в других ситуациях. Например, пользователю можно предоставить разрешения на схему (при этом он получит права на все объекты этой схемы), назначить ему схему по умолчанию и т. п. Про работу со схемами будет рассказываться в разд. 5.3.2.
- **Встроенные средства шифрования данных.** В SQL Server 2005 теперь можно шифровать данные в таблицах, используя встроенные возможности самого сервера. При этом предусмотрено множество способов шифрования. Для этой цели можно использовать сертификаты, асимметричные и симметричные ключи и просто пароли. Подробно про применение встроенных средств для шифрования данных будет рассказано в разд. 5.6.
- **Изменение контекста выполнения кода Transact-SQL.** Эта возможность позволяет менять контекст прямо в процессе выполнения этого кода. Для этого используется выражение `EXECUTE AS` (см. разд. 5.5).
- **Расширенные возможности работы с логинами SQL Server.** Например, теперь для логинов можно применять *парольные политики*, заставлять пользователей менять пароль при следующем входе в систему и т. п. Про работу с логинами речь пойдет в разд. 5.2.

1.1.4. Новые возможности для обеспечения отказоустойчивости

Одна из главных задач, которая была поставлена перед разработчиками SQL Server 2005, — это радикальное повышение отказоустойчивости SQL Server 2005 и снижение времени простоя сервера. Для решения этих проблем в SQL Server 2005 появились следующие возможности:

- **Возможность создания, изменения и удаления индексов в оперативном режиме (*online*).** Пользователи могут одновременно с выполнением этих операций работать с соответствующей таблицей. Эта возможность может оказаться очень удобной для серверов, которые должны работать в круглосуточном режиме. Она доступна только для SQL Server 2005 Enterprise Edition. Для того чтобы ей воспользоваться, предусмотрен новый параметр `ONLINE` для команд `CREATE INDEX`, `ALTER INDEX`, `DROP INDEX` и `ALTER TABLE`.
- **Поддержка большего числа узлов при работе в кластере.** Теперь количество узлов в кластере SQL Server 2005 ограничивается только возможностями операционной системы (это утверждение верно только для редакций Enterprise Edition и Developer Edition, поскольку SQL Server 2005 Standard Edition поддерживает кластер максимум из двух узлов). В предыдущих версиях SQL Server можно было использовать кластеры максимум из четырех узлов для 32-разрядных систем и максимум из восьми узлов для 64-разрядных систем. Про работу с кластерами будет рассказано в *разд. 7.1*.
- **Выделенное административное подключение (*Dedicated Administrator Connection*).** При запуске SQL Server 2005 обязательно резервируются ресурсы на одно подключение. Если на SQL Server 2005 возникли какие-то проблемы (например, некорректно написанный скрипт забрал себе все ресурсы сервера), вы можете использовать зарезервированные ресурсы для подключения в этом специальном режиме. Затем вы можете, к примеру, закрыть соединение, которое запустило этот скрипт. Выделенное административное подключение обладает приоритетом перед всеми остальными. Кроме того, подключившись в этом режиме, вы получаете дополнительные права на выполнение действий, которые при обычном подключении запрещены (например, на внесение изменений напрямую в таблицы базы данных `master`). Подробно про работу в режиме выделенного административного подключения будет рассказываться в *разд. 3.4.2*.
- **Зеркальное отображение баз данных (*database mirroring*).** Зеркальное отображение — очень важная новая возможность, которая позволяет поддерживать точную копию текущей базы данных в оперативном режиме

(*online*) на другом сервере, а также производить автоматическое переключение пользователей между серверами в том случае, если первый сервер вышел из строя. Возможности зеркального отображения баз данных очень похожи на возможности кластера, однако для его настройки вам не нужно никакого специального оборудования, а сами серверы, которые принимают в нем участие, могут находиться очень далеко друг от друга. К сожалению, достичь необходимого уровня зрелости этой технологии пока не удалось, поэтому Microsoft не рекомендует использовать зеркальное отображение для рабочих серверов. Кроме того, серверам, на которых настроено зеркальное отображение, отказано в поддержке со стороны соответствующих подразделений Microsoft. Подробно про зеркальное отображение баз данных будет рассказано в разд. 7.3.

- **Моментальные снимки баз данных (*database snapshots*).** С пользовательской точки зрения моментальные снимки представляют собой слепки базы данных по состоянию на определенный момент времени. Их можно использовать для того, чтобы вернуться к состоянию базы данных на определенный момент времени, например, на момент до начала выполнения рискованной операции (применение патчей, присланных разработчиками приложения), или для создания отчетов (на начало месяца, квартала и т. п.). Однако моментальные снимки кардинально отличаются от резервных копий баз данных, которые можно использовать для тех же целей. Изначально моментальный снимок базы данных — это просто набор пустых страниц (поэтому он создается очень быстро). При внесении любого изменения в базу данных старый вариант страницы, в которую вносится изменение, передается в распоряжение снимка, а изменения вносятся уже в новый вариант этой страницы. Для этого используются достаточно сложные возможности файловой системы NTFS. Снимки баз данных доступны только в редакции Enterprise Edition.
- **Контрольные суммы (*checksums*) для проверки целостности страниц базы данных.** В предыдущих версиях SQL Server для проверки целостности страниц базы данных использовались только контрольные биты. Такая технология называлась обнаружением поврежденных страниц (*torn page detection*). Она доступна и в SQL Server 2005, но кроме нее можно также использовать контрольные суммы для страниц базы данных (по умолчанию для создаваемых баз данных настроено именно использование контрольных сумм). Применение контрольных сумм позволяет более надежно обнаруживать любые повреждения в файлах данных. Подробно про использование контрольных сумм и альтернативных возможностей для проверки целостности страниц будет рассказываться в разд. 4.7 (параметр PAGE_VERIFY).

- **Зеркалирование носителей при резервном копировании (*mirrored backup media*)**. Теперь при проведении резервного копирования можно создавать одновременно несколько зеркальных копий создаваемых файлов. Эта возможность призвана повысить надежность резервного копирования. Подробно про нее будет рассказано в разд. 6.3.2.
- **Открытие базы данных для доступа пользователей в фазе отката при восстановлении работоспособности экземпляра (*instance recovery*)**. Теперь пользователи могут работать с базой данных еще до завершения отката всех незавершенных транзакций при восстановлении базы данных после сбоя. Эта возможность позволяет сократить время, необходимое для восстановления системы. Она доступна только в SQL Server 2005 Enterprise Edition. Подробнее с ней можно ознакомиться в разд. 6.4.
- **Игнорирование ошибок во время операций восстановления**. Если при восстановлении (или резервном копировании) базы данных обнаружилась какая-то ошибка, в SQL Server 2005 можно попытаться ее проигнорировать и попробовать продолжить восстановление или резервное копирование. Для этой цели предназначен новый параметр команд BACKUP и RESTORE — CONTINUE_AFTER_ERROR. Подробнее про него и про другие параметры будет рассказываться в разд. 6.6.
- **Восстановление открытой базы данных (*online database restore*)**. Эта возможность призвана сократить время простоя при сбоях. Однако для восстановления в таком режиме предусмотрено множество ограничений, например, в любом случае файл или файловая группа, для которой проводится это восстановление, должна быть переведена в автономный режим (*offline*), нельзя производить восстановление для первого файла базы данных и т. п. Такой режим восстановления будет рассматриваться в разд. 6.7.1.
- **Режим EMERGENCY для базы данных**. Этот режим предназначен для проведения диагностики при подозрении на наличие каких-то проблем. База данных в этом режиме доступна только на чтение, запись в журнал транзакций не производится, обращаться к ней могут только администраторы сервера. Подробно про этот режим будет рассказано в разд. 4.6.

1.1.5. Новые возможности системы репликации

В системе репликации SQL Server 2005 сохранены все возможности предыдущих версий. В добавление к ним появилось большое количество новых средств, которые позволяют расширить функциональность этой системы.

- **Возможность выбора учетных записей для агентов репликации**. Если в предыдущих версиях SQL Server репликация происходила практически

только от имени учетной записи, под которой работал SQL Server Agent, то в SQL Server 2005 у вас появились богатые возможности настройки безопасности для разных агентов репликации.

- **Новые возможности работы со столбцами идентификатора (*identity columns*).** Появилась возможность реплицировать столбец *identity* именно как столбец идентификатора, а не как обычный столбец с базовым типом данных (например, *int*). Кроме того, появилась возможность распределять диапазоны значений идентификатора между серверами, которые принимают участие в репликации слиянием.
- **Трассировочные маркеры (*tracer tokens*).** Такие маркеры представляют собой небольшие объемы служебных данных, которые предназначены специально для диагностики репликации. После отправки такого диагностического пакета можно проследить его путь стандартными средствами мониторинга репликации.
- **Инициализация данных на подписчике вручную при помощи резервных копий.** Эта возможность очень удобна при настройке репликации больших таблиц. Она позволяет не забивать каналы репликации большим объемом данных.
- **Возможность изменять структуру опубликованных таблиц (добавлять и удалять столбцы).** Эти изменения будут переданы подписчику стандартными средствами репликации.
- **Возможность использовать одноранговую (*peer-to-peer*) репликацию.** При использовании этого типа репликации изменения можно вносить на любом сервере, который принимает в ней участие, эти изменения отразятся на всех остальных серверах.
- **Расширение поддержки репликации с базами данных других производителей.** Появилась возможность напрямую подписываться на публикацию в базе данных Oracle. Кроме того, можно использовать репликацию моментальных снимков и транзакционную репликацию в случаях, когда в роли издателя выступает SQL Server 2005, а в роли подписчика — Oracle или IBM DB2.
- **Новый набор программных объектов RMO (Replication Management Objects — объекты управления репликацией).** Эти программные объекты используются для автоматизации управления репликацией.
- **Разрешение конфликтов при репликации слиянием при помощи собственных программных модулей.** Такие модули можно создавать на любом .NET-совместимом языке.

Подсистема репликации SQL Server 2005 и ее новые возможности будут рассматриваться в гл. 12.

1.1.6. Новые объектные модели для автоматизации администрирования

Одной из самых удачных и удобных для администраторов новых возможностей SQL Server 7.0 и SQL Server 2000 стала объектная модель SQL-DMO (Distributed Management Objects), при помощи которой можно автоматизировать любые операции по администрированию SQL Server (при помощи программного кода на языках VBScript или JavaScript или на любом другом COM-совместимом языке программирования — Visual Basic, VBA, C++, Java, Delphi и т. п.). Поддержка объектной модели SQL-DMO сохранена и в SQL Server 2005, но, к сожалению, она не развивается. При помощи SQL-DMO нельзя обращаться к новым объектам SQL Server 2005 — только к тем, которые были и в SQL Server 2000. Причина проста: объектная модель SQL-DMO не требует поддержки .NET, а именно .NET-совместимые объектные модели Microsoft старается развивать в первую очередь. Зато в SQL Server 2005 появились новые объектные модели для автоматизации администрирования.

- **Объектная модель SMO (SQL Management Objects).** Эта объектная модель применяется для выполнения административных операций на SQL Server 2005 и призвана заменить объектную модель SQL-DMO, которая использовалась в предыдущих версиях. Отличием модели SMO является то, что она реализована средствами .NET. Объектная модель SMO будет рассматриваться в разд. 9.2.
- **Новые поставщики (провайдеры) WMI (Windows Management Instrumentarium).** Они обеспечивают возможности обращения к SQL Server 2005 средствами стандартного интерфейса WMI (см. разд. 9.4). WMI Provider for Configuration Management обеспечивает средствами для работы со службами SQL Server и серверными сетевыми библиотеками (см. разд. 9.4.7), а WMI Provider for Server Events предоставляет возможности для работы с уведомлениями о событиях SQL Server (*event notifications*) (см. разд. 9.4.8).

1.2. Новые возможности SQL Server 2005 для программистов

В этом разделе кратко перечисляются те новые возможности, которые SQL Server 2005 предоставляет для программистов. Эти возможности подробно рассмотрены в другой книге автора, которая называется "SQL Server 2005 для программистов" и готовится к выходу в издательстве "БХВ-Петербург".

- **CLR (Common Runtime Language) Integration — интеграция среды выполнения .NET.** Теперь можно обращаться к классам сборок .NET

прямо из кода Transact-SQL. Это, с одной стороны, ответ Oracle, где можно обращаться к классам Java, а с другой стороны, расширение возможностей, реализованных в предыдущих версиях SQL Server при помощи хранимых процедур автоматизации (`SP_OACreate`, `SP_OAMethod` и т. п.), которые использовались для обращения к классам COM. Обращаться к сборкам .NET очень удобно во многих ситуациях. Например, когда вы хотите использовать уже готовый программный код (для выполнения операций в файловой системе, для работы с электронной почтой и т. д.), когда вам нужно реализовать выполнение некоторых ресурсоемких операций (например, шифрование) во внешних программных модулях, когда вы хотите использовать программные конструкции обычных языков программирования и т. п. Отметим только, что для всех баз данных SQL Server 2005 CLR Integration по умолчанию отключена (как и возможность применения хранимых процедур автоматизации `SP_OACreate`, `SP_OAMethod` и др.). Пуще всего для включения этих возможностей использовать консоль SQL Server Surface Area Configuration (см. разд. 3.5).

□ **Возможность создавать свои собственные агрегатные функции.** В предыдущих версиях SQL Server можно было использовать только встроенные агрегатные функции (`SUM()`, `MAX()`, `MIN()`, `AVG()` и т. п.). В SQL Server 2005 можно создавать свои собственные агрегатные функции со своей программной логикой. Эти функции должны быть написаны на .NET-совместимых языках в виде классов с обязательным набором свойств и методов. Затем скомпилированную сборку нужно зарегистрировать в базе данных при помощи команды `CREATE ASSEMBLY`, а затем еще раз зарегистрировать ее в базе данных уже как пользовательскую агрегатную функцию при помощи команды `CREATE AGGREGATE`.

□ **Возможность создавать свои пользовательские типы данных .NET.** В предыдущих версиях SQL Server также можно было создавать пользовательские типы данных (на основе встроенных), но здесь имеется в виду совершенно другое. В SQL Server предыдущих версий пользовательские типы данных использовались в основном как дополнительное средство проверки вводимых пользователем значений (классический пример — пользовательский тип данных для почтового индекса). В SQL Server 2005 появилась возможность создавать пользовательские типы данных в виде классов .NET со своим набором свойств и методов. Затем эти типы можно использовать для столбцов таблиц, или для переменных в коде Transact-SQL, или для передачи параметров хранимым процедурам и функциям и т. п.

Для того чтобы можно было применять пользовательские типы данных, нужно вначале создать сборку .NET с соответствующим набором свойств и методов, затем зарегистрировать ее в базе данных при помощи команды

CREATE ASSEMBLY, а затем создать пользовательский тип данных в базе данных при помощи команды CREATE TYPE;

- **Создание программных объектов (хранимых процедур, триггеров, пользовательских функций) с применением кода .NET.** С использованием средств .NET можно создавать не только агрегатные функции и пользовательские типы данных. Программный код .NET можно использовать и в хранимых процедурах, триггерах и пользовательских функциях. Реализация этой возможности похожа на работу с пользовательскими типами данных: вначале создаем сборку .NET с требуемым программным кодом и стандартным набором свойств и методов, затем регистрируем эту сборку в базе данных при помощи команды CREATE ASSEMBLY, а затем используем ее при создании программного объекта. Например, при создании хранимой процедуры с использованием кода .NET нам потребуется команда CREATE PROCEDURE ... AS EXTERNAL NAME.
- **Возможность обращения к SQL Server 2005 как к Web-службе (Native XML Web Services).** При помощи этого средства клиенты могут обращаться на SQL Server напрямую по протоколу HTTP и передавать запросы в XML-совместимом формате SOAP. Результаты запроса возвращаются также в формате SOAP/XML. При этом SQL Server выглядит как стандартная Web-служба — с доступным стандартным определением на языке WSDL (Web Services Definition Language — язык определений Web-служб), со стандартными средствами аутентификации и шифрования данных и т. п.

Преимущества обращения к SQL Server как к Web-службе очевидны:

- упрощается интеграция с приложениями и средствами третьих фирм. Web-службы — это стандарт, который поддерживается практически всеми крупными производителями средств разработки;
- протокол HTTP, который используется для обращения к Web-службам, очень удобен для работы с низкоскоростными и ненадежными соединениями, а также для выполнения асинхронных запросов. В результате резко упрощается реализация удаленных и мобильных клиентов;
- упрощается решение вопросов, связанных с безопасностью. Например, чтобы разрешить доступ к SQL Server 2005 на брандмауэрах, при использовании Web-службы потребуется открыть минимум портов. Для целей аутентификации и шифрования данных также можно использовать стандартные средства, применяемые для защиты Web-серверов.

При настройке SQL Server 2005 как Web-службы необходимо определить точки подключения по HTTP (*HTTP Endpoints*). Такие точки подключения создаются в базе данных при помощи команды CREATE ENDPOINT. Отметим

только, что для работы с ними потребуется специальный драйвер HTTP.SYS, который поставляется только с Windows Server 2003 и Windows XP SP2. Поэтому такая возможность будет недоступна для SQL Server 2005, установленного на Windows 2000.

- **Новый тип данных xml.** Он может использоваться для хранения документов XML и фрагментов документов XML (т. е. документов, для которых не предусмотрен корневой элемент) в столбцах таблиц. Кроме того, этот тип данных можно использовать для переменных Transact-SQL, для параметров хранимых процедур и функций и т. п. Его можно использовать для хранения данных в формате XML размером до 2 Гбайт.

Отметим, что для этого типа данных в SQL Server 2005 предусмотрен специальный набор методов, который позволяет, например, выполнять для документа XML запросы на языке XQuery, вносить в него изменения при помощи команд на специальном языке XML DML (XML Data Modification Language — язык изменения данных XML) и т. п.

- **Параметры для массовой загрузки данных в виде документов XML.** В предыдущих версиях SQL Server можно было определять некоторые параметры массовой загрузки данных при помощи файлов форматирования. В этих файлах можно было указать соответствия между столбцами в текстовом файле и в таблице SQL Server, выбрать только нужные столбцы и т. п. В SQL Server 2005 для указания этих параметров можно использовать файлы XML (которые можно создать вручную или сгенерировать при помощи утилиты bcp). Про массовую загрузку данных средствами SSIS (SQL Server Integration Services) будет рассказано в разд. 10.8.

- **Возможность применения конструкции try ... catch для обработки исключений в коде Transact-SQL.** Теперь ошибки, которые возникают в ходе выполнения команд Transact-SQL, можно перехватывать и обрабатывать при помощи тех же синтаксических конструкций, что и в .NET-совместимых языках, например, в C#. За счет этого значительно повысилось как удобство, так и надежность обработки исключений.

- **Служебные представления каталога баз данных.** В SQL Server 2005 появилось множество новых служебных представлений, предназначенных для получения информации об объектах баз данных. Частично они повторяют служебные таблицы из предыдущих версий SQL Server (сама система служебных таблиц изменилась очень сильно), например, sys.sysobjects, но многие из них впервые появились в SQL Server 2005.

- **Новые функции ранжирования (rank(), dense_rank(), ntile() и row_number()).** Эти функции позволяют определить "ранг" для каждой записи в возвращаемом наборе по отношению к какому-нибудь значению, например, к сумме закупок для заказчика.

- **Выражение OUTPUT для команд INSERT, UPDATE и DELETE.** С его помощью можно вернуть набор записей, для которых были произведены операции вставки, изменения или удаления. Этот возвращаемый набор удобно использовать для протоколирования, дополнительных проверок, выдачи подтверждающих сообщений в приложениях и т. п.
- **Модификатор max для типов данных varchar, nvarchar и varbinary.** При использовании модификатора max эти типы данных могут использоваться для хранения информации размером до 2 Гбайт. Фактически, varchar(max) заменяет тип данных text, nvarchar(max) — ntext, а varbinary(max) — image. Традиционные типы данных text, ntext и image также оставлены, но только для обеспечения обратной совместимости. Преимуществом такого подхода является то, что с большими текстовыми и двоичными значениями теперь можно работать, как с обычными (получать их при помощи курсоров, передавать текстовые значения строковым функциям и т. п.)
- **Общие табличные выражения (Common Table Expression).** Это специальный набор записей, который определяется при выполнении запроса и может быть использован в этом запросе (например, для соединения — join). Работа с общими табличными выражениями очень похожа на работу с вложенными запросами, однако общие табличные выражения гибче. Например, к данным в общем табличном выражении можно обращаться несколько раз в рамках одного запроса.
- **Оператор APPLY.** Этот оператор позволяет вызывать функцию, возвращающую табличный набор записей, для каждой записи из первой (внешней таблицы) и передать ей значение из столбца этой таблицы. Оператор APPLY можно использовать в двух вариантах:
 - CROSS APPLY проверяет, возвращает ли эта табличная функция непустой набор значений, и если да, то записывает в возвращаемый набор результатов запись из внешней таблицы;
 - OUTER APPLY возвращает все записи из внешней таблицы (а не только те, для которых табличная функция вернула непустое значение). Но если табличная функция возвращает NULL, то в тот столбец, в который должно подставляться ее значение, также будет записываться NULL.
- Например, при помощи оператора APPLY можно пройти по всем записям таблицы MyTable, для каждой записи передать значение из столбца Column1 этой таблицы функции MyFunction и получить в результате только те записи из MyTable, для которых эта функция возвращает непустое значение:

```
SELECT * FROM MyTable CROSS APPLY MyFunction(MyTable.Column1);
```
- **Операторы PIVOT и UNPIVOT.** Эти операторы позволяют менять местами столбцы и строки в возвращаемых результатах, что может быть очень

удобно при создании отчетов (особенно отчетов в виде перекрестных таблиц — *cross-tabs*). В принципе то же самое можно было сделать и в предыдущих версиях SQL Server, но для этого потребовалось бы создать программный код на языке Transact-SQL. В SQL Server 2005 за счет возможности применения этих операторов выполнение такой операции значительно упростилось.

- **Уведомление об изменениях в базе данных (Query notification).** Это еще одна новая и очень удобная возможность SQL Server 2005, которая позволяет отслеживать изменения в таблицах базы данных и реагировать на эти изменения. Обычный пример использования такого уведомления выглядит так: предположим, что наше приложение должно показывать какую-то информацию из базы данных, например, список всех заказчиков с суммой продаж по каждому. Запрос, который генерирует эту информацию, является достаточно ресурсоемким, и мы заинтересованы в том, чтобы выполнять его как можно реже. Поэтому приложение, выполняя этот запрос, одновременно передает на SQL Server 2005 запрос на уведомление. Затем оно использует кэшированные данные выполненного запроса. Как только данные, которые использовались в запросе, изменятся на SQL Server, приложению сразу придет уведомление об изменении. Оно очистит кэш и выполнит запрос заново.

Для работы с уведомлениями необходимо настроить службу Service Broker для базы данных. Применения Notification Services (специального программного компонента SQL Server 2005) или уведомлений о событиях (*event notifications*) не требуется.

- **Выражение BULK для функции OPENROWSET().** С его помощью функция OPENROWSET(), которая традиционно использовалась для обращения к удаленным источникам данных, теперь может применяться и для массовой загрузки данных (*bulk load*) на SQL Server. При этом можно загружать как обычный табличный набор записей, так и большие двоичные и текстовые данные. Для этой цели в SQL Server 2005 предусмотрен специальный тип поставщика массовой вставки — BULK Provider.
- **Возможность оператором TOP принимать выражения и использоваться в командах INSERT, UPDATE и DELETE.** Раньше для этого оператора можно было применять только числовые значения. Теперь ему можно передавать выражения, которые будут вычисляться в момент выполнения (например, переменные). Теперь оператор TOP можно использовать и в командах INSERT, UPDATE и DELETE, например, чтобы вставить во временную таблицу записи о десяти самых крупных заказчиках.
- **Оператор AT для команды EXECUTE.** Этот оператор позволяет выполнить хранимую процедуру на подключенном (*linked*) сервере.

- **Выражение TABLESAMPLE.** Это выражение позволяет вернуть только определенное количество строк в наборе записей. Оно может использоваться, например, для отладки запросов, которые в обычном режиме возвращают большое количество данных. Можно указать процент от общего числа записей, которые нужно вернуть, или явно указать количество записей. В отличие от оператора TOP, возвращаются случайно выбранные значения.
- **Параметры SET NULL и SET DEFAULT для каскадного обновления данных.** Теперь в выражении REFERENCES при создании таблицы можно указать, кроме параметра CASCADE, еще и эти два значения. Параметр SET NULL означает, что при каскадном обновлении данных для столбцов в таблицах с внешними ключами будет устанавливаться значение NULL, а параметр SET DEFAULT — значение по умолчанию.
- **Снятие ограничения на максимальный размер записи в таблице в 8060 байт.** Если превышение размера в 8060 байт возникло из-за данных в столбцах nvarchar, varchar, varbinary и sql_variant, то ошибки при вставке или изменении данных не возникает. Механизм реализации этой возможности выглядит так же, как и при работе со столбцами text, ntext и image: на странице, принадлежащей таблице, помещается указатель, а сами данные перемещаются на другую страницу. Такое решение может снизить производительность, но, с другой стороны, в некоторых ситуациях может оказаться очень удобным.
- **Возможность изменения плана выполнения запроса без внесения изменений в текст запроса (руководства по запросам — *plan guides*).** Эта новая возможность SQL Server 2005 очень пригодится администраторам и специалистам, которые работают с уже готовыми приложениями. Для создания таких руководств на SQL Server 2005 предусмотрена хранимая процедура sp_create_plan_guide. Подробнее про нее и про работу с планами выполнения запросов будет рассказываться в разд. 11.5.7.
- **Форсированная параметризация (*forced parameterization*).** При использовании этой возможности любое явно определенное значение, которое передается команде SELECT, INSERT, UPDATE или DELETE, будет трактоваться как параметр. В некоторых ситуациях при применении такой форсированной параметризации можно получить выигрыш в производительности за счет снижения повторных компиляций планов выполнения команд Transact-SQL.
- **Параметр PERSISTED для вычисляемых столбцов.** Этот параметр позволяет рассчитывать значения для таких производных столбцов не в момент выполнения запроса, а при занесении данных в таблицу. Если расчеты в вычисляемом столбце могут потребовать значительных ресурсов, то такое решение может оказаться очень удобным. В предыдущих версиях SQL

Server заранее рассчитывать значения для вычисляемых столбцов можно было только при помощи индексированных представлений.

- **Множественные активные наборы результатов MARS (Multiple Active Result Set).** Эта новая возможность, которая обеспечивается средствами SQL Native Client, в некоторых ситуациях позволяет очень сильно повысить производительность работы клиентских приложений. Ее смысл очень прост: приложение может посыпать новый запрос, не дожинаясь возврата результатов предыдущего. Фактически запросы теперь можно производить в асинхронном режиме.
- **Управление блокировками для страниц индексов.** В SQL Server 2005 появилась возможность определять, как именно будут налагаться блокировки на элементы индекса: на уровне записей (`ALLOW_ROW_LOCKS`) или на уровне страниц (`ALLOW_PAGE_LOCKS`). Эти параметры определяются при создании или изменении индекса при помощи команд `CREATE INDEX` и `ALTER INDEX` соответственно. Более дробный режим блокировок (на уровне записей) обычно больше подходит для систем OLTP, а блокировки на уровне страниц — для хранилищ данных.
- **Индексы для столбцов с типом данных XML.** Сами столбцы с типом данных `XML` также являются новой возможностью SQL Server. Индексы для таких столбцов позволяют серьезно ускорить доступ к данным. Индексы, которые бывают двух типов — первичные и вторичные, позволяют проиндексировать имена элементов XML, пути к ним в документе XML, значения атрибутов и т. п.
- **Новые хинты оптимизатора (*optimizer hints*).** Они позволяют более точно определить план и особенности выполнения запросов. В SQL Server 2005 предусмотрено четыре новых хинта:
 - `RECOMPILE` — план выполнения такого запроса не будет использоваться повторно для аналогичных запросов. Обычно такой хint применяется для запросов, параметры которых изменяются очень сильно, и кэшированный план может оказаться неоптимальным;
 - `OPTIMIZE FOR` — позволяет оптимизировать план выполнения под конкретное значение передаваемого параметра для запроса;
 - `USE PLAN` — предписывает запросу использовать явно определенный план (передав его явно в виде строкового значения в формате XML);
 - `PARAMETERIZATION` — позволяет переопределить для запроса режим параметризации, если тот режим, который установлен на уровне всей базы данных, для данного запроса неоптimalен.

- **Библиотека SQL Server Native Client.** Этот новый программный интерфейс практически представляет собой надстройку над набором драйверов OLE DB, которая позволяет использовать новые возможности SQL Server 2005, например, множественные активные наборы результатов (Multiple Active Result Set, MARS), поддержку типа данных XML и т. п.

1.3. Новые возможности SQL Server Integration Services (DTSX)

Что появилось нового в новой версии DTS, которая в SQL Server 2005 называется SQL Server Integration Services (SSIS, используется также название DTSX), описать достаточно сложно, поскольку новой является вся подсистема. Изменился формат пакетов (теперь пакеты — это файлы в XML-совместимом формате), изменилась среда разработки пакетов (разработка пакетов производится из среды Business Intelligence Development Studio, т. е. фактически из Visual Studio.NET 2005), изменилась среда выполнения пакетов и т. п. Поэтому здесь мы отметим только некоторые новые наиболее важные возможности SSIS. Подробнее работа с SSIS будет рассматриваться в гл. 10.

- **Разработку пакетов SSIS теперь можно производить без подключения к SQL Server.** Фактически теперь пакеты SSIS — это специальный тип проекта Visual Studio, и разработку пакетов вполне можно производить в автономном режиме, без подключения к источникам данных.
- **Появились новые, очень мощные средства для целей отладки и протоколирования работы пакетов.** Теперь для этой цели можно использовать как стандартные средства Visual Studio, так и специализированные средства SSIS (Data Viewers — просмотрщики данных, Log Providers — набор драйверов для ведения протоколов выполнения пакетов и т. п.).
- **Появился новый набор элементов для управления логикой работы пакетов (Control Flow Tasks).** При помощи этих элементов очень удобно определять программную логику выполнения пакетов. Например, при помощи контейнера **Foreach Loop** можно выполнять определенные операции с каждым членом какой-то коллекции (например, с набором файлов в каталоге), при помощи контейнера **For Loop** — выполнять операции в цикле, пока выбранное вами условие не вернет FALSE, и т. п.
- **В SSIS предусмотрено большое количество новых задач.** В их число входят: Data Flow Task, Script Task, XML Task, File System Task, Web Service Task, WMI Data Reader Task, WMI Event Watcher Task и т. п. В результате функциональность пакетов SSIS значительно возросла.

- **Работу пакетов SSIS можно продолжать с определенной точки (после приостановки, возникновения ошибок и т. п.).** Такая возможность может оказаться очень удобной при работе с большим количеством данных.
- **Пакеты теперь можно защищать не только при помощи паролей, но и при помощи ролей базы данных MSDB.** Кроме того, появилась новая возможность — защита пакетов от изменения при помощи цифровых подписей.

При помощи подсистемы SSIS в SQL Server 2005 можно решать самые сложные задачи по перемещению и преобразованию данных.



ГЛАВА 2

Планирование и установка SQL Server 2005

2.1. Планирование установки SQL Server 2005

2.1.1. Оценка архитектуры приложения на основе SQL Server 2005

Эта книга предназначена для администраторов, которые обычно не могут принимать решения относительно того, на какой платформе будет работать приложение. Как правило, им приходится обслуживать уже готовое решение на основе SQL Server 2005. Однако при приеме приложения в эксплуатацию, т. е. перед тем, как производить развертывание SQL Server 2005, администратору настоятельно рекомендуется проверить некоторые моменты, связанные с архитектурой приложения. Такие вопросы намного проще решить с разработчиками до ввода приложения в эксплуатацию, чем пытаться внести изменения в работающую систему.

С первой проблемой, по наблюдению автора, сталкиваются, наверное, на большинстве предприятий. Суть проблемы очень проста. Предположим, что в эксплуатацию сдается обычная задача, например, по учету деятельности предприятия. Вначале объем информации в базе данных небольшой, и пользователи работают вполне комфортно. Однако проходит время, и когда объем информации достигает нескольких десятков гигабайт, пользователи начинают жаловаться: запросы выполняются очень долго, нужную информацию "на лету" не посмотреть, формирование отчетов может занимать несколько часов и т. п. Пользователь, конечно, всегда прав, но обычно какое-то время администраторы игнорируют их жалобы. Когда же не обращать внимания на жалобы становится невозможно, начинается оптимизация производительности системы. Обычные действия выглядят так:

- первым делом покупается мощный многопроцессорный сервер с хорошим RAID-массивом и большим объемом оперативной памяти;

- таблицы и индексы дефрагментируются (а иногда система индексов полностью переделывается);
- ресурсоемкие операции (загрузка и выгрузка данных, формирование больших отчетов и т. п.) переносятся на нерабочее время;
- оптимизируются подключения пользователей (например, подключения по ODBC заменяются на подключения по OLE DB);
- вручную настраиваются (например, при помощи хинтов оптимизатора) самые важные запросы пользователей.

Конечно же, все эти действия дадут только временный эффект. Пройдет еще какое-то время, эффект от них закончится, и база данных "встанет" окончательно. Главная проблема заключается в том, что в рабочей базе данных хранятся лишние данные по старым договорам, закрытым периодам и т. п. Единственное решение — это убирать их из рабочей базы данных на регулярной основе.

Наиболее удачное решение выглядит так: базу данных надо разделить на две. При этом каждая база данных должна храниться на своем сервере (они оптимизируются совершенно по-разному). Рабочая база данных, в которой пользователи выполняют текущие операции (заносят информацию о новых договорах, закрывают их, формируют текущие отчеты и т. п.), называется базой данных OLTP (OnLine Transaction Processing — оперативной обработки транзакций). На регулярной основе (раз в сутки, раз в месяц, раз в квартал и т. п.) информация о старых транзакциях переносится в другую базу данных — Data Warehouse (хранилище данных). Обычно для этой цели используются пакеты SSIS/DTS.

Информация в Data Warehouse, как правило, доступна только на чтение и используется для:

- формирования отчетов;
- получения аналитической информации (например, как выглядят тенденции по прибыльности такой-то линейки продуктов за продолжительный период). Часто для анализа на основе данных из Data Warehouse формируется еще одна база данных — OLAP (OnLine Analytical Processing — оперативная аналитическая обработка данных), где вместо таблиц используются кубы с преаггрегированными итогами (за период, по региону, по продуктовой линейке и т. п.).

На практике же вместо системы с базами данных OLTP/Data Warehouse чаще всего каждый год просто закрывают старую базу данных, и начинают новую базу, в которую переносятся остатки из старой. А иногда историю просто удаляют из рабочей базы данных, но это наименее удачный вариант, поскольку эти данные могут еще потребоваться в разных ситуациях.

Все эти способы вполне можно реализовать, если о них подумали заранее — на этапе проектирования приложения (или хотя бы при сдаче в эксплуатацию). Если же задача работает уже несколько лет и "обросла" системой отчетов и вспомогательных приложений, то менять ее будет очень сложно.

Вторая проблема, о которой тоже лучше подумать сразу: будут ли с вашей базой данных работать удаленные территориально пользователи, например, из филиалов, во время командировок, из дома и т. п. Если да, то лучше изначально предусмотреть решение. Обычно используются следующие варианты:

- **применение терминальных технологий** (Microsoft Terminal Server или Citrix). Преимуществами этих технологий являются защищенность, отсутствие необходимости устанавливать на пользовательские компьютеры клиентские приложения (кроме клиента Terminal Server), работа практически с любыми операционными системами (при использовании Citrix). К недостаткам можно отнести то, что придется открывать порты терминальных протоколов для обращения из-за пределов вашей сети, на что сетевые администраторы идут не всегда. Кроме того, не все клиентские приложения работают без проблем через терминальный сервер;
- **применение Web-интерфейса** (который можно использовать и для пользователей в локальной сети). К недостаткам этого способа можно отнести обязательную необходимость дополнительной защиты трафика HTTP, например, средствами SSL (Secure Socket Layer);
- **применение репликации**. Если пользователей в филиале много, а сетевое соединение с этим филиалом перегружено, то, возможно, есть смысл установить в филиале отдельный SQL Server и настроить **репликацию** (во внедневное время) с главным сервером. Подходит вам этот способ или нет — зависит от вашей задачи.

Еще один архитектурный момент, над которым стоит подумать — нужно ли разносить компоненты вашего приложения по разным компьютерам. Условно большинство приложений, которые построены на основе SQL Server, можно разделить на три уровня:

- **уровень данных** — сами таблицы и индексы, средства обеспечения целостности данных и прочие компоненты, которые непосредственно связаны с хранением данных;
- **уровень бизнес-логики** — то, как выполняются операции вашего приложения, например, добавление нового заказа или клиента. Чаще всего этот уровень реализуется при помощи хранимых процедур, хотя можно использовать и COM-компоненты, и реализацию бизнес-логики на клиенте;
- **уровень представления данных** — интерфейс для представления данных пользователю. Чаще всего используется Windows- или Web-приложение.

Во власти разработчика разнести эти компоненты по отдельным компьютерам или, наоборот, объединить на одном компьютере. При этом следует принимать во внимание:

- **соображения производительности.** Мы можем просто снять часть нагрузки с сервера данных путем переноса бизнес-логики на другой сервер;
- **соображения отказоустойчивости.** Серверов, которые обеспечивают бизнес-логику или представление данных (терминальные серверы, Web-серверы) может быть несколько, и в случае отказа одного из них пользователи могут продолжить работу с другим сервером;
- **соображения безопасности.** Например, если к данным SQL Server открыт доступ из Интернета через Web-приложение, то, конечно, безопаснее будет поместить Web-сервер на отдельном компьютере по отношению к SQL Server.

Все эти моменты перед вводом приложения в эксплуатацию администратору необходимо обязательно проверить. Конечно же, существует также множество других моментов, на которые следует обратить внимание, но о них речь пойдет в следующих разделах.

2.1.2. Выбор оборудования

Следующее, что предстоит сделать администратору, — выбрать подходящее "железо" для нашего сервера.

Надо сказать, что все зависит, конечно, от вашей конкретной задачи. Для каких-то целей вполне достаточно обычного персонального компьютера, который будет выступать в роли сервера (например, для разработки), а для других задач на сервер придется потратить десятки тысяч долларов. Самый правильный подход здесь — это измерить средствами Performance Monitor/System Monitor расход ресурсов (например, оперативной памяти) на работу типичного пользователя и умножить полученное значение на количество ожидаемых пользователей (при этом надо не забыть про то, что их количество в будущем может увеличиться). Можно также попробовать эмулировать нагрузку для большого количества подключений средствами специальных программ, например, SQL Hammer из SQL Server 2000 Resource Kit.

Далее будет рассмотрена каждая из подсистем оборудования сервера с указанием минимальных требований для SQL Server 2005 и с некоторыми комментариями.

Первая подсистема — **подсистема центрального процессора**. Минимальные требования для всех редакций SQL Server 2005 — Pentium III 500 МГц. При этом требование к типу процессора (не ниже Pentium III) является обязательным, а требование к частоте — желательным (будет выдано предупреждение,

но установку можно будет продолжить). Конечно же, работа с "минимальным" процессором никакого удовольствия вам не доставит, рекомендованные требования Microsoft — 1 ГГц и выше (для любых редакций SQL Server 2005).

Объективно измерить, хватает ли вам мощности процессора на вашем сервере, можно при помощи Системного монитора (меню **Пуск | Программы | Администрирование | Производительность**). Подробно про работу с Системным монитором и его счетчиками будет рассказываться в разд. 11.4. Нужный счетчик называется **Процессор: % загруженности процессора** (он выбирается по умолчанию в Системном мониторе). В соответствии с Microsoft значение этого счетчика в течение продолжительного промежутка времени (например, в течение рабочего дня) не должно превышать 80%.

И еще обратим внимание на два момента, которые связаны с процессорной подсистемой.

Существует 64-битная версия SQL Server 2005, которая, конечно, должна устанавливаться на 64-битную версию Windows. Эксперименты показывают, что использование этой версии действительно дает заметный выигрыш в производительности. Но следует учитывать, что 64-битная версия SQL Server 2005 — это фактически только ядро базы данных. Пока не предусмотрено 64-битных средств администрирования (пока администрировать такой сервер придется с 32-битной рабочей станции), будут недоступны многие возможности подсистемы SSIS, сильно ограничены возможности репликации, вообще не поддерживается Reporting Services, отсутствует поддержка многих сетевых библиотек и т. п. Так что при выборе платформы обязательно подумайте, устроит ли вас версия SQL Server с такими ограничениями.

Вторая подсистема — это **подсистема оперативной памяти**. Минимальные требования к компьютеру, на который устанавливается SQL Server 2005, — 512 Мбайт оперативной памяти, для SQL Server 2005 Express Edition (бывшая MSDE) — 192 Мбайт. Для большинства редакций SQL Server 2005 количество используемой оперативной памяти лимитируется только ограничениями операционной системы, но есть и исключения: SQL Server 2005 Express Edition использует не более 1 Гбайт оперативной памяти, а SQL Server 2005 Workgroup Edition — не более 3 Гбайт.

В SQL Server 2005 за счет активного использования платформы .NET требования к оперативной памяти сильно повысились как для ядра базы данных, так и для средств администрирования. Объективно оценить, достаточно ли у вас оперативной памяти, можно при помощи двух счетчиков Системного монитора:

- **Память: Обмен страниц в сек.** Этот параметр определяет, достаточно ли физической оперативной памяти для всего вашего компьютера. Физически

он показывает число обращений к файлу подкачки в секунду. Согласно Microsoft, для систем с SQL Server 2005 среднее значение этого параметра не должно превышать 10 для продолжительного промежутка времени;

- **SQL Server Buffer Manager: Buffer cache hit ratio** (Менеджер буфера SQL Server: процент попаданий в кэш). Это количество запросов пользователей, которые обслуживаются из кэша буферов базы данных без необходимости обращения к файлам данных на диске. Этот параметр показывает, достаточно ли оперативной памяти самому SQL Server 2005. Для систем OLTP, к которым относится подавляющее большинство баз данных, значение этого счетчика должно быть не меньше 90% (после нескольких часов работы при реальной нагрузке). Для хранилищ данных в связи с их спецификой (мало изменений и много операций полного сканирования таблиц) требования к оперативной памяти ниже, зато выше требования к дисковой подсистеме.

Третья подсистема — **дисковая**. Минимальные требования по дисковому пространству при установке всех компонентов — около 700 Мбайт в зависимости от выбранной версии SQL Server 2005. При этом большая часть дискового пространства (390 Мбайт) уйдет на учебные базы данных, которые, конечно, на рабочем сервере вам не нужны. С точки зрения оценки производительности дисковой подсистемы, главный счетчик в Системном мониторе для нее — **Логический диск: % активности диска**, т. е. сколько процентов от общего времени дисковой подсистеме приходится работать. В соответствии с Microsoft значение этого счетчика не должно приближаться к 100% на протяжении продолжительного промежутка времени.

В официальных учебных курсах Microsoft описывается идеальная конфигурация файловой системы для SQL Server. Опыт показывает, что эта конфигурация действительно является наиболее удобной. Выглядит она так:

- первый раздел (назовем его условно **C:**) отводится под **файлы операционной системы и программные файлы** (а также **служебные базы данных**) самого SQL Server 2005. Рекомендованный размер — не ниже 4 Гбайт (учитывая размеры современных пакетов обновлений, лучше позаботиться о дополнительном пространстве). Конечно, этот раздел должен быть отформатирован в NTFS. Этот раздел рекомендуется помещать на два зазеркальзованных жестких диска (большой нагрузки на них все равно не будет, а это самый простой способ защититься на случай выхода из строя одного диска);
- второй раздел (условно **D:**) отводится только под **файлы рабочей базы данных** (без журналов транзакций или каких-то других файлов). Рекомендуется использовать для него внешний RAID-массив не ниже 5-го уровня.

Внешний — потому, что в этом случае при отказе сервера вы сможете просто подключить этот RAID-массив к другому серверу и присоединить к нему вашу рабочую базу данных. Операцию присоединения проще всего выполнить из SQL Server Management Studio в окне **Object Explorer** (Проводник объектов), используя команду **Attach** (Присоединить) контекстного меню для контейнера **Databases** (Базы данных). Времени на такую операцию уходит намного меньше, чем на восстановление с ленты.

Если вы приобретете внешний RAID-массив, входящий в список совместимого оборудования (Hardware Compatibility List, HCL) Microsoft для Cluster Services, то вы можете при желании реализовать кластер для SQL Server 2005.

Дискового пространства на этом разделе должно быть много. Рекомендуется определить, сколько места потребуется вашей базе данных с учетом ее роста в ближайшие годы, и умножить полученную цифру на два. Дополнительное дисковое пространство может сэкономить вам множество сил при выполнении операций восстановления, перестроения индексов и т. п.

Конечно, этот раздел должен быть отформатирован в файловой системе NTFS. Для максимальной производительности он форматируется с размером блока в 64 Кбайт (чтобы соответствовать размеру экстента SQL Server). При форматировании средствами графического интерфейса такой размер не выбрать — придется использовать команду `FORMAT` из командной строки;

- третий раздел (условно Е:) отводится только под **журналы транзакций рабочей базы данных**. Существует четкое правило: файлы журналов транзакций должны находиться на разных физических дисках по отношению к файлам баз данных. Соблюдение этого правила позволит в случае отказа диска восстановить базу по состоянию на момент сбоя, а несоблюдение — только на момент создания последней резервной копии.

Выбор вариантов реализации этого раздела сильно зависит от разных условий. Например, в зависимости от требований к отказоустойчивости для вашей базы данных этот раздел можно создать на отдельном RAID-массиве, на двух зазеркальзованных дисках или просто на одном жестком диске (потеря журналов транзакций, если сохранились файлы данных, большой трагедией не является — их можно очень быстро сгенерировать заново, отключив и вновь подключив базу данных).

Размер этого раздела больше всего зависит от трех факторов: режима восстановления базы данных (**Full**, **Bulk-logged** или **Simple** — см. разд. 4.5), меняется на вкладке **Files** (Файлы) свойств базы данных), интенсивности

внесения изменений и частоты резервного копирования. Максимального размера журнала транзакций требуют специальные промежуточные базы данных (*staging databases*), в которые каждую ночь могут загружаться данные из разных источников, сводиться воедино, обрабатываться и затем, после записи результатов в файл или в хранилище данных, удаляться. Для таких баз данных требуемый размер журнала транзакций может приближаться к размеру самой базы данных. Минимальные требования к этому разделу предъявляют редкоизменяемые хранилища данных. Обычный размер файлов журналов транзакций в базах данных OLTP — около 10% от общего объема рабочей базы данных.

В обычных условиях операции чтения для раздела, на котором находятся журналы транзакций, почти не проводятся — только операции записи. Поэтому существует мнение, что этот раздел выгоднее форматировать в файловой системе FAT или FAT32. В этом случае операции записи будут производиться быстрее и будет проще дефрагментировать этот раздел. Однако Microsoft рекомендует использовать для SQL Server только разделы NTFS, и, по мнению автора, какого-то заметного выигрыша в производительности при использовании FAT или FAT32 не получить.

Автору не раз доводилось видеть такую картину: покупается мощный сервер с одним дорогим и надежным RAID-массивом. Этот RAID-массив становится одним разделом, на который помещается все: и программные файлы, и файлы баз данных, и файлы журналов транзакций. Конечно же, такой подход является неправильным. Самое лучшее "железо" RAID-массива не спасет вас, к примеру, от ошибок в драйвере SCSI-контроллера, что может привести к потере всех данных. Кроме того, при таком подходе могут возникнуть проблемы с фрагментацией диска.

Четвертая подсистема сервера — **сетевая**. С точки зрения SQL Server, автору почти не приходилось встречаться с ситуациями, когда эта подсистема стала бы узким местом. Наоборот, очень часто главным фактором, который требует перевода приложения с настольной системы (Access, FoxPro, Paradox) на SQL Server, является то, что сеть не справляется с потребностями по постоянному перемещению файлов MDB, DBF, DB с файл-сервера на клиентский компьютер и обратно. С SQL Server таких проблем не возникает.

Единственный момент — SQL Server 2005 не удастся установить на компьютер, на котором нет сетевого адаптера. Если вы хотите поэкспериментировать с сервером дома, а сетевой карты у вас нет, то потребности SQL Server 2005 вполне удовлетворит программный эмулятор сетевого адаптера от Microsoft — Адаптер Microsoft замыкания на себя (Microsoft Loopback Adapter). Он имеется в дистрибутиве Windows NT 4.0, Windows 2000, XP и 2003 (рис. 2.1).

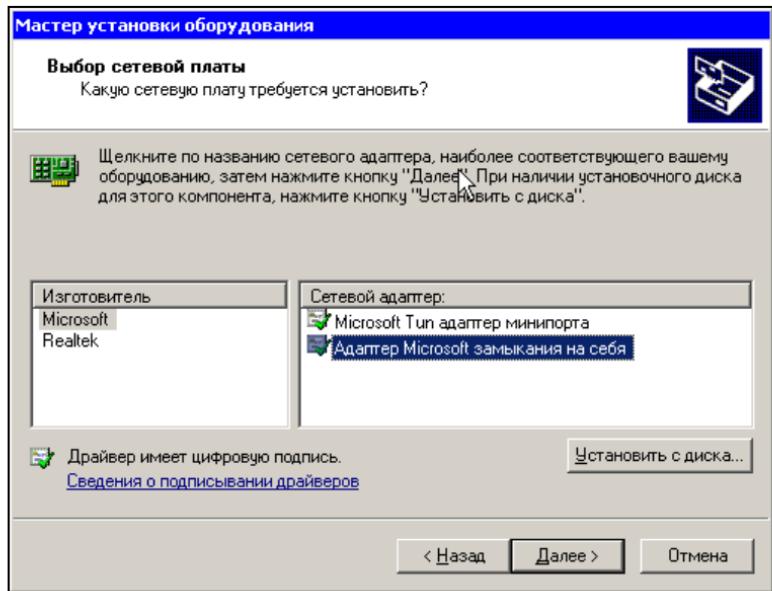


Рис. 2.1. Установка программного эмулятора сетевого адаптера

2.1.3. Выбор операционной системы и ее компонентов

Вы выбрали нужное оборудование под SQL Server 2005, заказали его, получили и установили в серверной. Следующая задача — выбрать операционную систему и установить в ней необходимый набор компонентов.

Сразу скажем, что реальный выбор у нас невелик. Фактически он ограничивается только Windows Server 2003. Теоретически можно установить SQL Server 2005 и на Windows 2000 Server, но делать этого не стоит, поскольку под Windows 2000 вам не будут доступны некоторые возможности SQL Server 2005, например, Native Web Services (другое название — HTTP Endpoints, Web-службы, создаваемые средствами самого SQL Server 2005).

Кроме того, SQL Server 2005 можно установить и на другие операционные системы. Всю таблицу совместимости разных версий SQL Server 2005 с различными версиями и сервис-паками операционных систем мы приводить не будем (ее можно посмотреть в документации), отметим только самые важные моменты:

- SQL Server 2005 Enterprise Edition можно установить не только на Windows Server 2003 Enterprise Edition, но и на Windows Server 2003 Standard Edition;

- все редакции SQL Server 2005, кроме Enterprise Edition (в том числе Standard Edition), можно устанавливать на клиентские операционные системы — Windows 2000 Professional и Windows XP.

Теперь поговорим о компонентах операционной системы, которые потребуются для установки SQL Server 2005. Как показывает опыт, в 90% случаях на отечественных предприятиях устанавливается SQL Server 2005 Enterprise Edition на Windows Server 2003 Enterprise Edition, поэтому рассмотрим компоненты именно для этой ситуации.

Первое, с чего нужно начать подготовку операционной системы, — установка **Service Pack 1** для Windows Server 2003.

Если вы планируете использовать Reporting Services, то следующее ваше действие — установка **Internet Information Server** (через **Пуск | Настройка | Панель управления | Установка и удаление программ | Установка компонентов Windows**). В Windows Server 2003 он входит в продукт под названием Application Server. Набор компонентов Application Server, предлагаемый по умолчанию при установке системы, вполне подойдет.

Для установки SQL Server 2005 на Windows Server 2003 больше ничего не нужно. Все остальные необходимые компоненты будут добавлены в процессе установки SQL Server 2005 автоматически.

2.1.4. Выбор редакции SQL Server 2005

Операционная система установлена, все необходимые компоненты добавлены, и все готово к началу установки SQL Server 2005. Последнее, что вам необходимо сделать, — это определить, какая редакция SQL Server 2005 вам нужна.

В вашем распоряжении большой выбор.

- **Enterprise Edition** — это наиболее полноценная версия SQL Server 2005 (и самая дорогая). Поддерживаются все возможности SQL Server 2005. Может устанавливаться как на Windows Server 2003 Enterprise Edition, так и на Windows Server 2003 Standard Edition.
- **Enterprise Edition 120-day Evaluation** — в этой редакции предусмотрены все возможности обычной Enterprise Edition. Отличается она тем, что работает только 120 дней, и, кроме того, ее можно установить на клиентские операционные системы (например, на Windows XP). Этую редакцию можно бесплатно скачать с Web-сайта Microsoft.
- **Developer Edition** — эта редакция также обладает всеми возможностями Enterprise Edition (но стоит значительно дешевле). Она, как понятно из названия, предназначена для разработки и тестирования приложений. При использовании ее в качестве рабочего сервера вы рискуете столкнуться с

ограничениями на количество одновременных подключений. Если сервер разработки у вас постепенно стал выполнять функции рабочего сервера, то вы можете обновить эту редакцию до Enterprise Edition.

□ **Standard Edition** — по отношению к редакции Enterprise Edition это более дешевая и менее функциональная версия SQL Server 2005. По сравнению с Enterprise Edition и Developer Edition она не поддерживает:

- кластеры более чем из двух узлов;
- некоторые режимы зеркального отображения баз данных;
- снимки баз данных;
- динамическое увеличение пула памяти при использовании AWE (Address Windowing Extensions — специальное средство, которое позволяет 32-разрядным компьютерам работать более чем с 4 Гбайт оперативной памяти; для компьютеров с меньшим объемом памяти не применяется);
- добавление оперативной памяти и другие изменения оборудования без отключения сервера;
- зеркалированное резервное копирование;
- операции с индексами (перестроение, изменение параметров и т. п.) без отключения пользователей;
- восстановление базы данных без отключения пользователей (отдельных страниц и файлов);
- секционирование таблиц и индексов;
- применение распределенных представлений (это представления, которые обращаются к таблицам на разных серверах; используются для того, чтобы распределить части большой таблицы по разным серверам);
- распараллеливание служебных операций с индексами и операций DBCC.

Кроме того, эта редакция может работать не более чем с 4 процессорами. Ее также можно обновить до Enterprise Edition.

□ **Workgroup Edition** — это новая редакция, аналогов которой не было в предыдущих версиях SQL Server. Она еще дешевле, и в ней меньше функций, чем в Standard Edition. По сравнению со Standard Edition также отсутствует возможность использовать сборки .NET в среде выполнения Transact-SQL (самая разрекламированная возможность SQL Server 2005), и совсем не поддерживаются кластеры и зеркальное отображение баз данных. Эта редакция работает максимум с двумя процессорами и 3 Гбайт оперативной памяти. Во всем остальном она идентична Standard Edition.

При необходимости эту редакцию можно обновить до Standard Edition или Enterprise Edition.

- **Express Edition** — это наследница встроенной версии SQL Server, которая раньше называлась MSDE (Microsoft Database Engine). В основном эта редакция предназначена для установки вместе с "коробочными" приложениями. Она бесплатная, но ее применение ограничено рядом лицензионных ограничений. Она оптимизирована для минимального количества одновременных подключений. Сравнивать ее функциональность с другими редакциями нет смысла: фактически Express Edition — это только ядро SQL Server 2005. Вместе с этой редакцией не поставляются никакие графические средства администрирования. Обновить до Enterprise Edition невозможно, но можно перенести данные на "нормальный" сервер.
- **Mobile Edition** — эта версия предназначена для использования на наладонных компьютерах, смартфонах и Tablet PC. Обычно она вызывает определенное удивление — какой SQL Server может быть на слабеньких наладонниках? На самом деле, это очень нужное приложение для разработчиков, учитывая, что мобильной версии Access не существует. На нем реализовано множество коммерческих проектов. Конечно, этот продукт только отдаленно напоминает "обычный" SQL Server.

Из многолетнего опыта общения автора со множеством слушателей можно заключить, что на подавляющем большинстве предприятий используется только Enterprise Edition для любых версий SQL Server. Администраторы и разработчики выбирают эту версию по принципу "пусть будет все" (благо платить приходится не из своего кармана), а проконтролировать их обычно некому. Часто бывает так, что возможности Enterprise Edition, отличающие его от Standard Edition, потом не используются никогда.

Учитывая эти традиции, в данной книге будут рассмотрены возможности именно Enterprise Edition. Если не оговорено иное, будем считать, что речь идет именно об этой редакции.

2.2. Установка SQL Server 2005

2.2.1. Несколько слов об установке

Установка SQL Server 2005 очень проста и каких-то проблем не вызывает. Проблемы могут возникнуть позже, когда выяснится, что при установке были выбраны не те параметры. Некоторые из таких проблем можно решить очень быстро, для решения других может потребоваться переустановка сервера. В этом разделе мы пройдемся по основным этапам установки SQL Server 2005 и обсудим некоторые моменты, которые могут повлиять на принятие решений при выборе возможных вариантов.

2.2.2. Начало установки. Выбор набора компонентов

Первый этап установки подготовительный. Программа установки SQL Server 2005 ставит недостающие компоненты операционной системы (.NET Framework 2.0) и установочные файлы самого SQL Server 2005.

Далее выполняется запуск специальной программы System Configuration Checker. Она проверяет ваш компьютер по 12 параметрам: на соответствие требованиям к оборудованию, к операционной системе, к установленным компонентам операционной системы и т. п. Отнеситесь очень внимательно к тем ошибкам и предупреждениям, которые она генерирует, иначе вы рискуете получить совсем не тот результат, на который рассчитывали. При помощи кнопки **Report** (Отчет) можно просмотреть или сохранить файл с отчетом, который генерирует System Configuration Checker. После этого начинается сама установка программного обеспечения SQL Server 2005.

После ввода информации о пользователе и серийного номера первый важный момент — это выбор необходимых компонентов (рис. 2.2). Набор компонентов SQL Server 2005 существенно отличается от набора компонентов в дистрибутивах предыдущих версий SQL Server, поэтому мы прокомментируем каждый пункт (более точно выбрать те или иные вложенные компоненты вы можете при помощи кнопки **Advanced** (Дополнительно)).

- **SQL Server Database Services** — это само ядро базы данных с возможностью выбрать установку средств настройки репликации и полнотекстового поиска. Ядро, скорее всего, вам будет необходимо (если только вы не ставите одни лишь утилиты администрирования на рабочую станцию), средства репликации нужны далеко не всегда, а полнотекстовый поиск в реальных приложениях вообще используется крайне редко (поскольку поддержка русского языка для полнотекстового поиска в SQL Server 2005 не предусмотрена).
- **Analysis Services** — это ядро баз данных OLAP, которое предназначено для работы с многомерными кубами вместо двумерных таблиц (как в обычных базах данных), а также средства администрирования баз данных OLAP, тематическая документация и т. п. Если вы пока не собираетесь заниматься кубами OLAP, то, скорее всего, Analysis Services вам не нужны. В SQL Server 2000 Enterprise Edition компонент Analysis Services поставлялся на компакт-диске с дистрибутивом SQL Server, но устанавливать его надо было отдельно.
- **Reporting Services** — очень интересное и мощное средство организации корпоративной системы отчетов. Код отчета создается на специальном

XML-совместимом языке RDL (Report Definition Language — язык определения отчетов), при этом, конечно, отчеты можно создавать графическими средствами Report Designer. Пользователи обращаются к отчетам через Web-интерфейс. Конечно же, средствами Reporting Services можно создавать отчеты не только для данных, которые находятся на SQL Server, но и для любых других источников данных, доступных по OLE DB. Reporting Services — это очень функциональная, мощная и интересная система, но по сравнению с отчетами Access или Crystal Reports несколько непривычная.

Reporting Services не поставлялись с дистрибутивом SQL Server 2000, но для этой версии сервера их можно бесплатно скачать с сайта Microsoft.

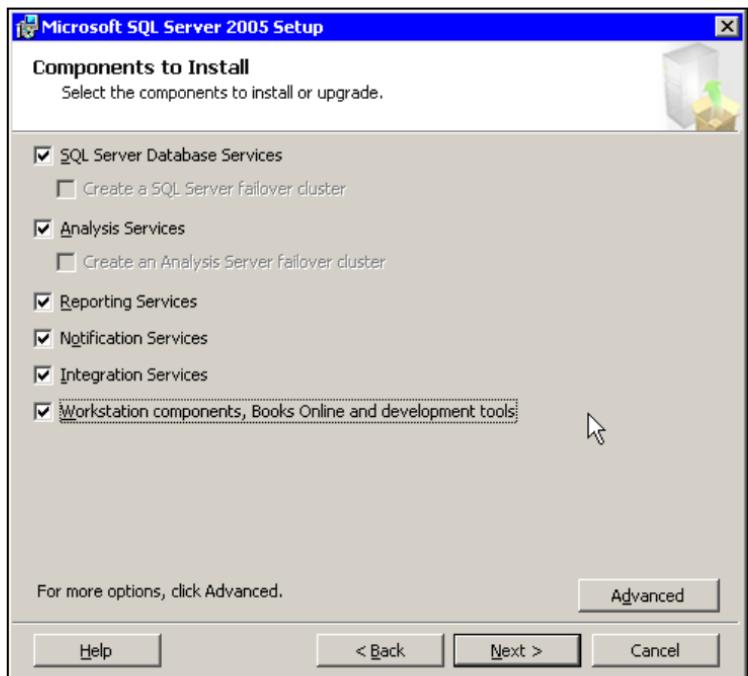


Рис. 2.2. Выбор компонентов SQL Server 2005 при установке

- **Notification Services** — это специальная надстройка над SQL Server 2005, основное назначение которой — уведомлять большой круг подписчиков (обычно по электронной почте) о каком-то событии, на которое они зарегистрировались: изменения в счете футбольного матча, котировки акций, результаты лотерей и т. п. Обычно событие, на которое реагируют Notification Services, — это изменение в таблице базы данных SQL Server. На практике Notification Services используются достаточно редко.

Notification Services не поставлялись с дистрибутивом SQL Server 2000, но, как и Reporting Services, их можно скачать с сайта Microsoft бесплатно. В SQL Server 2005 они просто добавлены в набор компонентов для установки.

- **Integration Services** — так в новой версии называются средства DTS из предыдущих версий SQL Server. Integration Services определяются как специальный набор графических и консольных утилит и программных объектов, главное назначение которых — перекачивать данные между источниками с возможным их преобразованием. Чаще всего Integration Services используются для регулярного переноса данных из базы данных OLTP в базу данных Data Warehouse, хотя ничего не мешает вам использовать их и для других целей (вплоть до генерации отчетов). При этом совсем необязательно, чтобы в качестве источника или получателя данных выступал SQL Server — средствами Integration Services, например, можно очень удобно реализовать перекачку данных с FoxPro на Oracle или между любыми другими источниками данных, к которым можно обратиться по OLE DB или ODBC. Integration Services очень сильно изменились по сравнению с DTS 2000. Функциональные возможности этой системы возросли многократно. Подробно работа с Integration Services будет рассматриваться в гл. 10.
- **Workstation components, Books Online and development tools** (Компоненты рабочих станций, справка и средства разработки) — это сетевые библиотеки для подключения в SQL Server 2005, графические и консольные утилиты, программные компоненты для разработчиков, документация и примеры, учебные базы данных. Как правило, все эти компоненты, кроме сетевых библиотек, не устанавливаются на рабочий сервер. Правильнее будет установить утилиты администрирования и документацию на рабочую станцию администратора, а учебным базам данных вообще не место на рабочем сервере.

Отметим, что набор средств администрирования очень сильно изменился по сравнению с предыдущими версиями SQL Server (например, вы уже не найдете ни Enterprise Manager, ни Query Analyzer). О новых утилитах администрирования речь пойдет в следующей главе. Также в новом SQL Server 2005 нет привычных учебных баз данных Northwind и Pubs (хотя их можно скопировать с серверов предыдущих версий или скачать с сайта Microsoft), вместо них используются базы данных AdventureWorks и AdventureWorksDW (учебное хранилище данных). Формат справки тоже не привычный CHM, а новый формат программы Document Explorer, так что справку будет трудно перенести, например, на наладонный компьютер.

Для подавляющего большинства рабочих приложений вам достаточно будет установить ядро (т. е. установить флажок **SQL Server Database Services** в

окне выбора компонентов) на сервер и утилиты администрирования с документацией на рабочую станцию администратора. Все остальные компоненты можно будет добавлять по мере необходимости.

Для ядра **SQL Server Database Services** и **Analysis Services** предусмотрен еще один параметр — **Create a failover cluster** (Создать отказоустойчивый кластер). Он предназначен для установки SQL Server 2005 или Analysis Services в кластер. Соответствующий флажок становится активным только тогда, когда программа установки обнаружит на компьютере работающий Cluster Server. Про работу SQL Server в кластере подробно будет рассказано в разд. 7.1.

2.2.3. Работа с именованными экземплярами

Следующий экран мастера установки SQL Server 2005 позволяет определить имя экземпляра по умолчанию или задать именованный экземпляр SQL Server (рис. 2.3).

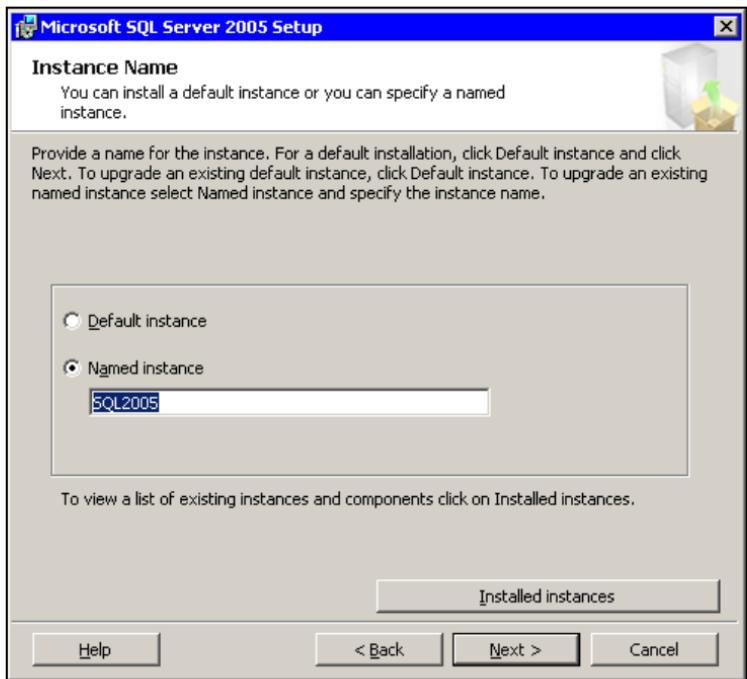


Рис. 2.3. Окно выбора имени экземпляра

Впервые концепция именованных экземпляров появилась в SQL Server 2000. Физически она позволяет установить несколько одновременно работающих экземпляров SQL Server (в том числе и разных версий) на одном компьютере.

Первый экземпляр всегда будет экземпляром по умолчанию (*default instance*). Его имя совпадает с именем компьютера, на который он установлен. Все остальные экземпляры будут именованными, и имена для них вам придется придумывать самостоятельно.

Например, если ваш компьютер называется MyComputer, а именованный экземпляр — MyInstance, то обращение к нему будет выглядеть как MyComputer\MyInstance (рис. 2.4).



Рис. 2.4. Подключение к именованному экземпляру SQL Server 2005

Не все старые клиентские приложения знают про именованные экземпляры и умеют обращаться к ним по имени со слэшем. Выйти из этой ситуации можно при помощи псевдонима (*alias*) для SQL Server, который можно настроить в SQL Server 2005 при помощи консоли SQL Computer Manager.

Уже имеющиеся экземпляры SQL Server на вашем компьютере можно просмотреть, воспользовавшись кнопкой **Installed Instances** (Установленные экземпляры) (см. рис. 2.3).

Для чего нужны именованные экземпляры? Вряд ли вы будете их использовать на рабочем сервере. Однако для целей разработки приложений и тестирования они очень удобны. Например, вы можете установить два экземпляра SQL Server 2005 на одном компьютере с совершенно разными настройками и сравнить, как они будут работать. Другой вариант (который наверняка придется по душе многим разработчикам) — установить в качестве экземпляра по умолчанию SQL Server 2000, а в качестве именованного экземпляра — SQL Server 2005 и работать с обеими версиями одновременно. Третий вариант — установить в качестве экземпляра по умолчанию SQL Server 7.0, в ка-

честве одного именованного экземпляра — SQL Server 2000, а в качестве третьего сервера — SQL Server 2005. В результате вы сможете работать одновременно с тремя версиями SQL Server!

Отметим только, что SQL Server 7.0 можно установить только в качестве экземпляра по умолчанию, а SQL Server 2000 не может быть именованным экземпляром, если экземпляром по умолчанию установлен SQL Server 2005.

Установка именованного экземпляра в добавление к уже работающим в терминологии SQL Server 2005 называется *side-by-side installation*. Новой возможностью SQL Server 2005 является то, что теперь именованные экземпляры можно использовать не только для самого SQL Server, но и для Analysis Services, Reporting Services и Notification Services. Именованные экземпляры предусмотрены даже для SQL Server 2005 Mobile Edition.

Кроме того, если SQL Server 2000 поддерживал максимум 16 именованных экземпляров во всех редакциях, то редакции Enterprise Edition и Developer Edition в SQL Server 2005 поддерживают до 50 именованных экземпляров на одном компьютере.

Если у вас достаточно мощный компьютер, и вы не хотите рисковать установленной операционной системой, то для тестирования SQL Server 2005 вполне можно использовать виртуальную операционную систему, работающую под VMWare или Virtual PC.

2.2.4. Выбор учетной записи для служб SQL Server

Следующий экран мастера посвящен выбору учетной записи для служб SQL Server (рис. 2.5). С его помощью вы можете выбрать учетные записи для самого SQL Server, SQL Server Agent, Analysis Services и Reporting Services.

В этом месте специалисты, которые производят установку SQL Server 2005, часто допускают ошибки.

Первый вопрос, с которым вы должны определиться, — будете ли вы использовать локальную системную учетную запись или доменную учетную запись (как еще один вариант, можно также использовать обычную, не системную, локальную учетную запись вашего компьютера)?

Локальная системная учетная запись — это специальная учетная запись, которая есть на каждом компьютере. От имени этой учетной записи работает сама операционная система и большинство служб Windows. Она заведомо обладает всеми необходимыми правами на локальном компьютере, на который устанавливается SQL Server 2005, и ее можно использовать для служб SQL Server 2005 в самых простых ситуациях. Однако помните, что при этом вы теряете возможность обращения SQL Server к любым внешним ресурсам

вашего домена. SQL Server 2005 не сможет взаимодействовать с Exchange Server, обращаться к другому экземпляру SQL Server при выполнении запроса, производить резервное копирование на сетевые диски, будет сильно затруднена настройка репликации. Поэтому на большинстве предприятий правильнее будет использование *доменной учетной записи*.

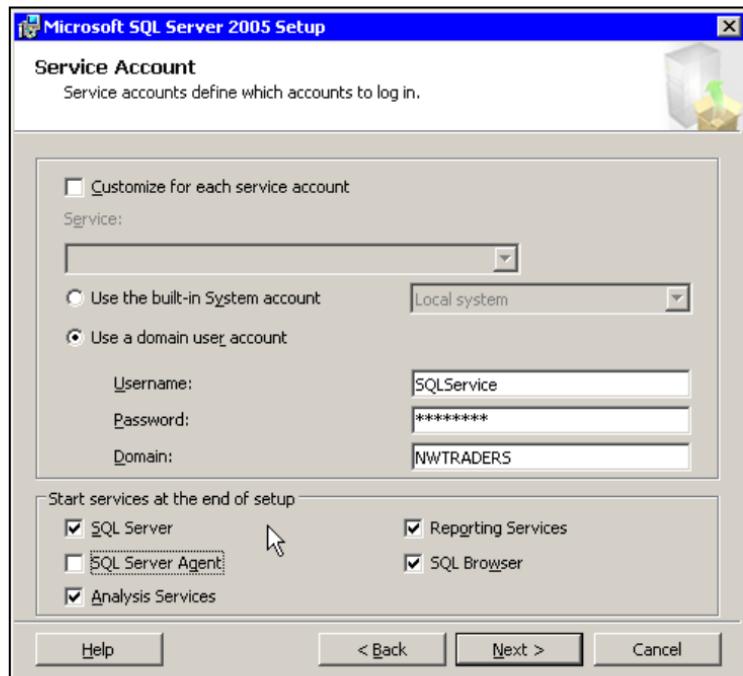


Рис. 2.5. Выбор учетной записи, от имени которой будут работать службы SQL Server 2005

Второй вопрос — какой должна быть эта доменная учетная запись? Иногда администраторы SQL Server в этом окне мастера прописывают параметры своей личной учетной записи (например, когда им не хочется обращаться с просьбами к администратору сети). Как правило, такое решение приводит к печальному результату. Типичная ситуация выглядит так: через несколько месяцев администратор SQL Server меняет пароль своей учетной записи и забывает его изменить для служб SQL Server. Внешне все продолжает работать нормально, поскольку SQL Server уже запущен и работает на сервере, который может не перезагружаться очень долго. Однако через какое-то время необходимость в перезагрузке все-таки возникает (например, устанавливается очередной патч), и после перезапуска SQL Server, конечно, не стартует. При этом выдается загадочное сообщение об ошибке, которое можно перевести примерно так: "При запуске службы возникла ошибка, но это не ошибка службы". Если вы не догадаетесь, в чем дело, все может закончиться переустановкой ни в чем не повинного сервера.

Поэтому правильное решение — это **доменная учетная запись, специально созданная для служб SQL Server 2005**. Для этой учетной записи необходимо снять флажок **User must change password at next logon** (Пользователь должен сменить пароль при следующем входе) и установить флажок **Password never expires** (Пароль никогда не устаревает). Пароль для нее должен быть сложным, его лучше записать и хранить в защищенном месте (поскольку уже через несколько месяцев сложный пароль вряд ли удастся вспомнить). Желательно также снабдить эту учетную запись комментарием, чтобы администраторы сети не забыли, что это такое и не отключили ее на всякий случай.

Третий вопрос — какими правами должна обладать эта учетная запись? Согласно рекомендациям Microsoft, учетная запись SQL Server Agent должна обладать административными правами на том компьютере, на котором устанавливается SQL Server, а учетная запись самого SQL Server 2005 должна работать с минимальными правами, которые должны быть предоставлены ей вручную. Практика, однако, показывает, что и учетную запись SQL Server 2005 лучше поместить в локальную группу администраторов на этом компьютере. Объяснения здесь простые:

- меньше ручной работы по предоставлению разрешений;
- на компьютере, на котором установлен SQL Server 2005, обычно нет другой ценной информации, кроме баз данных самого SQL Server. А на свои базы данных учетная запись, от имени которой работает SQL Server, будет обладать полными правами в любом случае;
- и самое главное — при использовании неадминистративной учетной записи вы все равно рано или поздно столкнетесь с ошибками, вызванными нехваткой прав (на ресурсы в файловой системе, на разделы реестра и т. п.). Удастся ли при этом быстро определить их причину — неизвестно.

Поэтому можно рекомендовать поместить в локальную группу Administrators (Администраторы) все учетные записи, используемые для работы службы SQL Server 2005 (обычно используется одна запись для всех). Если же это запрещено политикой безопасности вашего предприятия, то при выдаче разрешений руководствуйтесь статьей Books Online "Setting Up Windows Service Accounts" (Настройка учетных записей служб Windows). Найти эту статью можно по поиску в справке SQL Server — в Books Online.

Обратите внимание также на то, что учетной записи, от имени которой работают службы SQL Server 2005, программа установки автоматически предоставляет следующие системные права на локальном компьютере:

- Log on as a service (Входить в систему как служба);
- Act as part of the operating system (Работать как часть операционной системы);

- Log on as a batch job (Входить в рамках выполнения пакета команд);
- Replace a process-level token (Заменять маркер безопасности процесса);
- Bypass traverse checking (Проходить сквозь каталоги, на которые этой учетной записи не предоставлены разрешения);
- Adjust memory quotas for a process (Настраивать квоты на использование памяти для процесса).

Если вы меняете первоначальную учетную запись на другую (например, старая была случайно удалена), не забудьте предоставить эти разрешения вручную.

И последний вопрос, связанный с выбором учетной записи, — SQL Server предоставляет вам возможность использовать для всех служб SQL Server 2005 одну учетную запись или определить для каждой службы отдельную. Что же выбрать?

Очень редко встречаются ситуации, когда применение разных учетных записей для служб SQL Server давало бы какие-то преимущества. Поэтому можно рекомендовать использовать самый простой вариант — определить **одну доменную учетную запись с административными правами на данном компьютере для всех служб SQL Server 2005**. Усложнять себе жизнь и возиться с разными учетными записями имеет смысл только тогда, когда это требование разработчиков приложения или службы безопасности вашего предприятия.

Обратите также внимание на то, что для домена в окне мастера, показанном на рис. 2.5, необходимо указать имя NetBIOS (т. е. самую левую часть полного доменного имени). Например, если ваш домен называется NWTRADERS.MSFT, то в поле **Domain** (Домен) нужно ввести NWTRADERS.

В этом окне есть еще одна группа флажков — **Start services at the end of setup** (Запускать службы по окончании установки). Эти флаги определяют, будут ли службы SQL Server 2005 автоматически запускаться при загрузке компьютера. Выбирайте, как вам удобнее.

2.2.5. Выбор режима аутентификации SQL Server 2005

На следующем экране мастера установки (рис. 2.6) вам предоставляется возможность выбрать режим аутентификации SQL Server 2005.

Системе безопасности SQL Server 2005 (в которой очень много изменений по сравнению с предыдущими версиями) посвящена гл. 5. Однако поскольку решение нам нужно принимать уже в процессе установки сервера, кратко охарактеризуем причины, на основе которых вам придется сделать выбор.

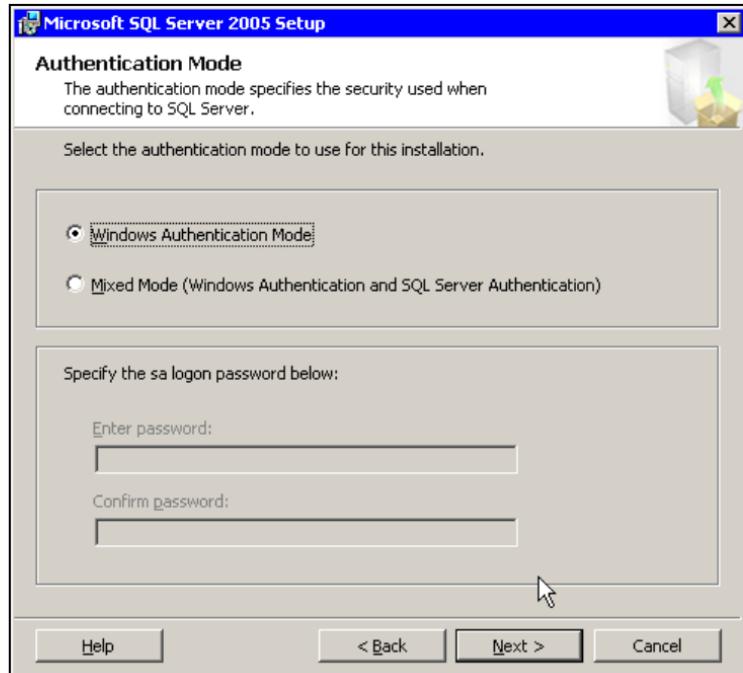


Рис. 2.6. Выбор доступных режимов аутентификации

В SQL Server 2005, как и в предыдущих версиях, предусмотрено два типа учетных записей для подключения SQL Server (*logins*, не путайте с *users* — пользователями базы данных!): **логины Windows** (Windows Authentication) и **логины SQL Server** (SQL Server Authentication). Логины Windows основаны на учетных записях Windows, которым предоставляется право подключаться к серверу SQL Server. Пользователям нет необходимости вводить какие-то пароли при подключении к SQL Server, если они уже вошли в сеть Windows.

В отличие от логинов Windows, логины SQL Server — это самостоятельные учетные записи со своими именами и паролями, информация о которых хранится в базе данных `master` на SQL Server. При подключении к серверу при помощи логина SQL Server вам придется явно указать имя логина и пароль.

Что лучше использовать — логины Windows или логины SQL Server? Microsoft однозначно рекомендует всегда стараться использовать только логины Windows (если у вас есть домен). Аргументы просты и бесспорны:

- пользователю нет необходимости запоминать дополнительные пароли, кроме пароля, под которым он входит в сеть;
- можно предоставлять разрешения при помощи групп Windows, что очень удобно на больших предприятиях;

- логины Windows лучше защищены (и с точки зрения возможности перебора паролей, и с точки зрения возможности перехвата парольных хешей по сети);
- при использовании логинов Windows подключение производится быстрее. Однако, по опыту автора, примерно на 80% установленных SQL Server на предприятиях используются логины SQL Server. Тех, кто выбирает этот способ, тоже можно понять:
- очень часто на предприятии за учетные записи домена и за SQL Server отвечают разные специалисты. Использование логинов SQL Server позволяет обойтись без обращения к администратору сети при появлении нового пользователя;
- разработчик может реализовать создание необходимых логинов SQL Server (а также назначить им пользователей баз данных, предоставлять права и т. п.) в установочном скрипте для своего приложения. В результате приложение будет уже готово к использованию — отпадет необходимости выполнять какие-либо дополнительные действия, зависящие от "местных" условий. Особенно это удобно для "коробочных" приложений;
- логины SQL Server будут работать и в случае, если у вас есть домен Windows NT/2000/2003, и тогда, когда домена у вас на предприятии нет.

Если вы производите развертывание приложения, созданного другими разработчиками, то выбора у вас нет — придется использовать тот тип логинов, который был предусмотрен разработчиками. Если вы сами разработчик, то решение вам придется принимать самостоятельно, основываясь на аргументах, представленных ранее.

В SQL Server логины Windows можно использовать всегда (поскольку этот тип подключения используется учетными записями служб SQL Server). Логины SQL Server можно использовать только тогда, когда режим аутентификации установлен в смешанный режим (Mixed Mode). Обычно на практике SQL Server при установке сразу переводится в смешанный режим. Выбирать использование только логинов Windows (Windows Authentication Mode) имеет смысл лишь тогда, когда у вас заведомо будут использоваться только логины Windows или этого требует политика безопасности вашего предприятия.

И еще один момент. Если вы выберете режим аутентификации Mixed Mode, вам потребуется определить пароль для учетной записи sa (системного администратора сервера). В отличие от предыдущих версий, в SQL Server 2005 к паролю для логина sa применяются те же требования, что и паролям для учетных записей Windows в вашем домене. Если вы устанавливаете SQL Server 2005 на компьютер, принадлежащий домену Windows 2003 с настрой-

ками по умолчанию, то вы не сможете ни оставить пароль пустым, ни использовать слишком простые пароли типа "password". Подробнее про парольные политики в SQL Server 2005 будет рассказано в гл. 5.

2.2.6. Выбор кодировки и порядка сортировки

Следующий экран мастера установки — выбор кодировки и порядка сортировки (рис. 2.7).

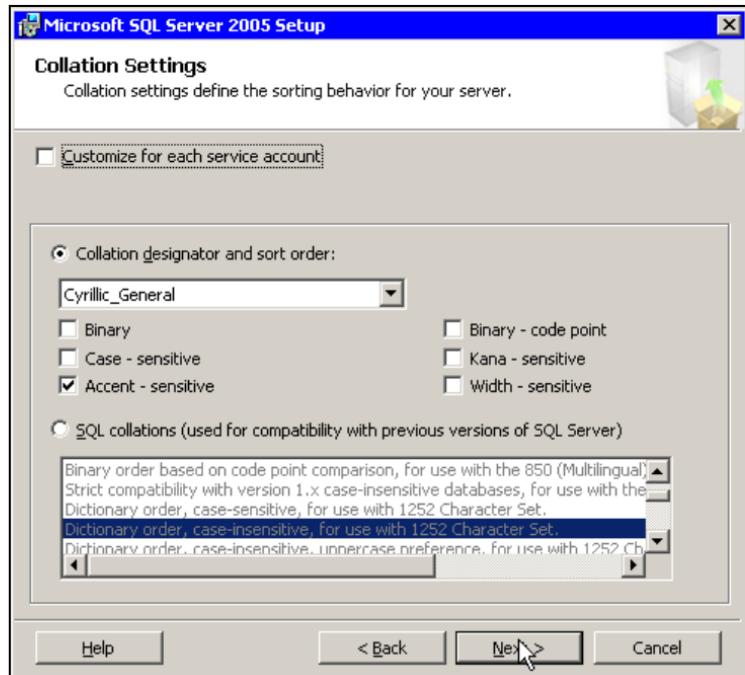


Рис. 2.7. Выбор кодировки и порядка сортировки

Именно на этом экране часто совершается неверный выбор, исправить который достаточно сложно.

Если вы производите установку SQL Server под готовое приложение, созданное сторонними разработчиками, то лучше не гадать, а обратиться к документации по этому приложению. Если же в документации ничего не сказано, то настоятельно рекомендуется остановить установку и обратиться за информацией к разработчикам. Как показывает опыт, угадать, какой кодировкой пользовались разработчики при создании своего приложения, очень сложно. Это зависит от того, какие версии SQL Server использовались для этого приложения, работало ли это приложение на других системах управления баз данных и т. п.

При этом помните о двух неприятных моментах:

- если вы выберите для сервера неверную кодировку и порядок сортировки, то вначале можно и не заметить каких-то проблем. Они появятся позже (при выполнении запросов с сортировкой, при соединениях и т. п.). Как правило, к этому моменту в базе данных уже накоплен определенный объем информации, что затрудняет исправление ситуации;
- в принципе, в SQL Server 2005 можно определить свою кодировку и порядок сортировки не только на уровне всего сервера, но и на уровне отдельной базы данных, таблицы и даже отдельного столбца таблицы. Однако если в вашем приложении используются системные таблицы, представления, хранимые процедуры из базы данных *master* (а в реальных приложениях это бывает очень часто), то проблемы все равно будут возникать. Единственный вариант здесь — перестройка базы данных (фактически переустановка сервера). В предыдущих версиях SQL Server для этого можно было использовать утилиту *rebuildm* (от англ. *rebuild master*). В SQL Server 2005 этой утилиты уже нет, и для перестройки снова придется использовать мастер установки. А перед этим в соответствии с инструкцией Microsoft необходимо будет выгрузить информацию из каждой таблицы в текстовые файлы и отскриптовать все объекты в базе данных, чтобы после переустановки сервера можно было воссоздать все заново.

Слушатели из одного крупного пивоваренного предприятия Санкт-Петербурга рассказывали автору, как им пришлось производить подобные процедуры для базы данных размером 7 Гбайт. Времени и сил на это у них ушло очень много.

Если же вы сами являетесь разработчиком и имеете возможность выбирать кодировку и порядок сортировки для сервера, то лучше всего посмотреть, какие кодировки используются на других SQL-серверах вашего предприятия. Единообразие в этом вопросе может во многих ситуациях оказаться очень удобным.

Если вы работаете с SQL Server предыдущих версий достаточно давно, то вероятнее всего, что для переноса существующих приложений лучшим выбором из списка **SQL Collations** (Кодировки SQL Server) будет **Dictionary order, case-insensitive, for use with 1251 (Cyrillic) Character Set** (Словарный порядок, нечувствительный к регистру, для использования с набором символов 1251 (Cyrillic)). Если вы создаете совершенно новое приложение, то самым удобным вариантом будет кодировка **Unicode Cyrillic_General** со снятым флагжком **Case-sensitive** (Чувствительность к регистру). Эта кодировка выбирается в списке **Collation Designator and Sort Order** (Определение сопоставления и порядка сортировки) в верхней части экрана.

То, что предлагает вам мастер установки по умолчанию, зависит от региональных настроек вашего компьютера. Если установлены русские региональ-

ные настройки, то по умолчанию будет предложена кодировка **Unicode Cyrilic_General**, если американские — **Latin1_General**.

Установленный флагок **Binary** (Двоичный) означает, что будет использоваться двоичный порядок сортировки вместо словарного (т. е. будут сравниваться числовые значения кодов символов). Флагок **Case-sensitive**, конечно, означает, будет ли учитываться регистр (как правило, это не требуется). Параметры **Accent-sensitive**, **Kana-sensitive** и **Width-sensitive** применяются только для дальневосточных языков.

Новой возможностью SQL Server 2005 является то, что кодировку и порядок сортировки можно выбрать не только для самого SQL Server, но и для Analysis Services. Принципы выбора кодировки для Analysis Services такие же, как и для самого SQL Server. При этом стоит учесть, что если основным источником данных для кубов OLAP у вас выступает хранилище данных, реализованное на SQL Server, то с точки зрения исключения преобразований и возможных ошибок наиболее выгодно будет установить на Analysis Services ту же кодировку и порядок сортировки, что и на этом SQL Server.

2.2.7. Остальные параметры установки

Если в числе устанавливаемых компонентов вы выбрали Reporting Services, то на двух следующих экранах мастера установки вам нужно будет выбрать параметры для этой службы. Reporting Services, как уже говорилось ранее, — это средства для организации системы корпоративных отчетов с возможностью создания отчетов, управления ими (размещение в папках, настройка разрешений и т. п.) и предоставления пользователям (через Web-интерфейс). Коротко перечислим те параметры, которые вы можете настроить.

- **Виртуальные каталоги Report Server и Report Manager** — это каталоги на сервере Internet Information Server, которые будут использованы для двух главных приложений Reporting Services — Report Server (среды выполнения отчетов) и Report Manager (отвечает за публикацию и администрирование отчетов). Эти параметры обычно имеет смысл менять только в том случае, когда виртуальный каталог с предлагаемым именем на компьютере уже существует.
- **Местонахождение базы данных Report Server** — Reporting Services в качестве источников для отчетов могут использовать любые доступные по OLE DB и ODBC источники данных. Однако служебная информация самого Report Server (настройки, определения отчетов в формате XML, разрешения и т. п.) обязательно должна храниться в базе данных SQL Server — уже существующего или устанавливаемого. В этом параметре вы можете определить сервер, на котором будет создана эта база данных, имя

базы данных Reporting Services и режим аутентификации или учетную запись, при помощи которой будет производиться обращение этой базе.

- **Имя хоста для сервера SMTP и почтовый адрес отправителя** — Report Server умеет генерировать отчеты по расписанию и отправлять их пользователям по электронной почте. Параметры электронной почты для Reporting Services устанавливаются на этом экране.

Последнее, что вы можете выбрать при работе мастера установки, — это отправлять автоматически информацию о возникновении серьезных ошибок в работе сервера на Web-сайт Microsoft или нет. Решение здесь принимать вам, но на большинстве предприятий из соображений безопасности эту возможность отключают.

При других вариантах установки (обновление предыдущих версий, установка SQL Server 2005 в кластер) набор экранов мастера установки будет отличаться. Об основных моментах, которые нужно принимать при создании кластера, будет рассказано далее в соответствующих разделах.

После того, как вы просмотрите выбранные вами параметры на экране **Ready to install** (Готовность к установке), начнется сама установка. Она производится медленнее, чем установка предыдущих версий SQL Server, но намного быстрее, чем, например, установка Oracle.

2.3. Автоматизированная и удаленная установка

Иногда может возникнуть необходимость провести установку SQL Server 2005 в автоматическом режиме или на удаленном компьютере. Например, разработчики "коробочного приложения" могут предусмотреть автоматизацию установки SQL Server 2005 при установке своего приложения. Другой случай — администраторам из центрального офиса предприятия может потребоваться провести единообразную установку SQL Server под какую-то задачу во всех филиалах. В этой же ситуации, возможно, пригодится и удаленная установка.

В SQL Server 2000 для автоматизированной установки можно было сгенерировать файл с расширением ini при помощи графического интерфейса мастера установки. Из того же графического интерфейса можно было провести и удаленную установку. В SQL Server 2005 эти возможности почему-то убрали, однако задача от этого не усложнилась.

Для автоматизированной установки необходимо использовать файл ответов со значениями параметров, которые при обычной установке выбираются на графическом интерфейсе. Самый простой способ создать файл ответов — скопировать файл template.ini из корневого каталога компакт-диска дистри-

бутива SQL Server 2005 и использовать его как шаблон. Этот файл хорошо прокомментирован, и проблем с его редактированием обычно не возникает.

Самый простой вариант команды на запуск автоматизированной установки может выглядеть так:

```
setup.exe /settings c:\setup\mysettings.ini
```

где `c:\setup\mysettings.ini` — это созданный вами файл шаблона. Можно использовать и более сложные варианты командной строки, когда часть параметров определяется не в `ini`-файле, а в самой командной строке. С полной справкой по параметрам команды `setup.exe` можно ознакомиться в статье "Running Setup from the Command Prompt" (Запуск установки из командной строки) из SQL Server Setup Help (этот файл справки открывается при нажатии на кнопку **Help** (Справка) на любом экране мастера установки). Отметим только два важных параметра, без которых провести полностью автоматизированную установку не получится:

- `/qn` — полное отключение любых диалоговых окон при выполнении установки;
- `/qb` — будут показываться только индикаторы процесса выполнения, при помощи которых можно следить за ходом установки. Никаких вопросов пользователю задаваться не будет.

Обратите внимание, что при помощи автоматизированной установки вы можете не только установить SQL Server 2005, но и удалить его, добавить компоненты, перестроить базу данных `master`, восстановить утраченные параметры в реестре, необходимые для работы SQL Server 2005, и т. п.

Удаленная установка выглядит точно так же, как и автоматизированная. Единственное отличие — в `ini`-файле необходимо обязательно указать параметры `TARGETCOMPUTER` (сетевое имя компьютера, на который будет производиться установка), `ADMINACCOUNT` (учетная запись с правами администратора на удаленном компьютере, которая будет использоваться для выполнения установки) и `ADMINPASSWORD` (пароль для этой учетной записи).

Удаленную установку можно выполнять только тогда, когда у вас в сети установлен домен, и только с использованием доменной учетной записи. Если удаленный компьютер не входит в домен, то провести удаленную установку SQL Server 2005 на него будет невозможно.

2.4. Обновление предыдущих версий SQL Server и миграция с Microsoft Access

Большинство администраторов и разработчиков стараются не использовать обновление старых серверов до более новых версий, и автор полностью раз-

деляет их мнение. Проблема заключается в том, что очень часто после обновления старое приложение начинает работать некорректно. Нередко такое бывает, если приложение напрямую обращается к системным таблицам SQL Server или использует какие-либо нестандартные возможности. Могут также перестать работать пакетные задания (*job*) SQL Server Agent, пакеты DTS, скрипты резервного копирования, репликация, расширенные хранимые процедуры и т. п. В SQL Server 2005 изменений и неподдерживаемых возможностей предыдущих версий очень много, поэтому риск еще больше.

На многих предприятиях в настоящее время продолжают работать решения на SQL Server 6.0 и SQL Server 6.5, и обычно это не вызывает никаких проблем. Переходить на SQL Server 2005 имеет смысл только одновременно с новой версией вашего приложения, которое изначально создавалось для использования с SQL Server 2005.

Если ваше приложение было создано на основе SQL Server 7.0 или SQL Server 2000, а для другой задачи вам нужно установить на тот же сервер SQL Server 2005, помните о возможности использования именованных экземпляров. Обычно это самый надежный и простой способ обеспечить сосуществование приложений, работающих с разными версиями SQL Server.

Еще один "мягкий" вариант — установить SQL Server 2005 на другой компьютер (или как другой именованный экземпляр) и перенести на него базы данных с предыдущих версий с одновременным обновлением. При этом вы, по крайней мере, гарантируете, что старый сервер на всякий случай у вас останется в неприкосновенности. Перенос баз данных с обновлением проще всего произвести так:

- для баз данных SQL Server 7.0 и 2000 перенос можно выполнить при помощи **Copy Database Wizard** (Мастера копирования баз данных). Для этого достаточно подключиться к старому серверу из SQL Server Management Studio, щелкнуть правой кнопкой мыши по объекту переносимой базы данных к контейнере **Databases** (Базы данных) и в контекстном меню **Tasks** (Задачи) выбрать пункт **Copy Database** (Скопировать базу данных);
- для баз данных SQL Server 6.0 и 6.5 (а также баз данных Access, FoxPro, Oracle, книг Excel, текстовых файлов) рекомендуется использовать **SSIS Import and Export Wizard** (Мастер импорта и экспорта SSIS), который в предыдущих версиях назывался DTS Import and Export Wizard. Его можно запустить из Business Intelligence Development Studio, а можно и при помощи файла DTSWizard.exe, который нужно найти в программных файлах SQL Server 2005.

После такого переноса с обновлением для перенесенной базы данных можно настроить режим совместимости с предыдущими версиями: от уровня 60 (т. е. совместимость с SQL Server версии 6.0) до 90 (т. е. SQL Server 2005).

В зависимости от того, какой уровень совместимости выбран, в базах данных будут доступны разные возможности, и команды Transact-SQL тоже будут вести себя по-разному. Установить уровень совместимости баз данных можно при помощи графического интерфейса SQL Server Management Studio (из окна свойств базы данных на вкладке **Options** (Настройки)) или при помощи хранимой процедуры `sp_dbcmptlevel`. Полную справку о различиях в поведении баз данных с разными уровнями совместимости ищите в справке по этой хранимой процедуре в Books Online.

Однако предположим, что вам все-таки предписано произвести обновление вашего существующего сервера до SQL Server 2005. Что при этом стоит учесть?

Во-первых, напрямую можно обновить только SQL Server версий 7.0 и 2000 (с любым сервис-паком). Если вам необходимо произвести обновление SQL Server 4.2, 6.0 или 6.5, то придется вначале обновить эти версии до SQL Server 7.0 или 2000.

Во-вторых, с сайта Microsoft можно скачать приложение, которое называется Upgrade Advisor (SQLUASetup.msi). Эта утилита анализирует существующие серверы SQL Server 7.0 и 2000 и установленные на них базы данных на предмет потенциальных проблем, которые могут возникнуть при обновлении до SQL Server 2005. По итогам обследования генерируется отчет со списком проблем и рекомендациями по их исправлению. Альтернатива Upgrade Advisor — утилита Best Practices Analyzer, которую также можно скачать с сайта Microsoft. Эта утилита проверяет ваш сервер и установленные базы данных на соответствие определенным правилам и рекомендациям. Часть правил относится к совместимости с SQL Server 2005. Подробно про работу с Best Practices Analyzer будет рассказываться в *разд. 11.5.3*.

Рекомендуется также не полагаться только на эти утилиты, но и прочитать статьи из раздела Backward Compatibility (Обратная совместимость) в Books Online, особенно раздел "Breaking Changes to Database Engine Features in SQL Server 2005" ("Важнейшие изменения в ядре базы данных в SQL Server 2005").

После этого можно приступать к обновлению, не забыв до этого провести полное резервное копирование старого сервера, включая программные файлы, пользовательские и системные базы данных. Обновление выполняется при помощи того же мастера, что и обычная установка SQL Server 2005. Точно так же можно произвести обновление Analysis Services, Reporting Services и Notification Services.

После обновления уровень совместимости баз данных, которые обновились вместе с сервером, автоматически устанавливается в соответствии с версией старого сервера.

На предприятиях часто возникает ситуация, когда на SQL Server нужно перенести базу данных Access. Например, изначально база данных для простоты была создана на Access, постепенно она стала использоваться очень активно сразу большим количеством пользователей, и работать с ней в Access стало неудобно. Как можно выполнить перенос?

В принципе, для переноса таблиц вместе с данными можно использовать те же Integration Services (т. е. новые DTS), например, SSIS Import and Export Wizard, о котором говорилось ранее. Однако при его использовании возникает существенная проблема — он не переносит связи между таблицами (например, отношения первичный/внешний ключ). Если в вашей базе данных десятки и сотни таблиц, то этот недостаток может быть оказаться существенным: создавать руками каждую из связей долго и неудобно. Поэтому для переноса на SQL Server 2005 (или 2000) лучше всего использовать специальное средство, которое встроено непосредственно в Access. В русскоязычной версии Access оно называется "Мастер преобразования в формат Microsoft Access", а в английской — Upsizing Wizard. Кроме корректного переноса таблиц с данными, индексов, связей между таблицами, проверок и значений по умолчанию, этот мастер умеет автоматически переориентировать существующие формы и отчеты приложения Access на использование таблиц SQL Server. Мастер из Access 2003 вполне корректно работает с SQL Server 2005.

2.5. Проверка установки и выполнение послеустановочных задач

2.5.1. Проверка результатов установки

Если вы устанавливаете SQL Server 2005 средствами обычного мастера установки, и в процессе установки вам не встретилось ни сообщений об ошибках, ни предупреждений, то, скорее всего, все установилось нормально. Если же вы выполняете автоматизированную установку, то по ее окончании рекомендуется проверить, действительно ли SQL Server 2005 установлен правильно. Самый простой способ провести проверку — запустить SQL Server Management Studio и в базе данных `master` выполнить любой запрос к установленному экземпляру SQL Server, например:

```
SELECT * FROM sysdatabases;
```

Другие способы проверить результаты установки могут быть такими:

- убедиться, что в консоли Services появились службы Analysis Services, Report Server, SQL Browser, SQL Server, SQL Server Agent, SQL Writer и т. п., в зависимости от набора установленных вами компонентов, и что эти службы можно останавливать и запускать;

- найти программные файлы SQL Server 2005 (по умолчанию они помещаются в каталог C:\Program Files\Microsoft SQL Server\90) и просмотреть их;
- убедиться, что в меню **Пуск | Программы** в Windows Explorer появился новый раздел **Microsoft SQL Server 2005** с ярлыками для утилит администрирования SQL Server 2005.

Если же в процессе установки возникли какие-то проблемы, то можно обратиться к логам установки SQL Server 2005. Файл лога формируется для каждого компонента SQL Server 2005. Все эти файлы помещаются по умолчанию в каталог C:\Program Files\Microsoft SQL Server\90\Setup Bootstrap\LOG\Files.

После того как установка полностью завершена, рекомендуется выполнить несколько послеустановочных задач.

2.5.2. Настройка сетевых библиотек для доступа клиентов

Первое действие, которое обычно необходимо выполнить после установки SQL Server 2005, чтобы обеспечить возможность подключения к нему клиентов, — настроить серверные сетевые библиотеки. В SQL Server 2000 вы могли выбрать необходимые сетевые библиотеки во время работы мастера установки. В SQL Server 2005 настроить сетевые библиотеки можно только после установки сервера (или использовать специальные параметры в ini-файле при выполнении автоматизированной установки). При этом изменений по сравнению с предыдущими версиями в этом компоненте SQL Server очень много, и вмешательство администратора потребуется часто.

Что такое серверная сетевая библиотека? Это специальный программный модуль, который принимает от клиентов запросы на установку соединения с SQL Server 2005. Под клиентами подразумеваются как привычные клиентские приложения, так и утилиты администрирования, другие серверы SQL Server и т. п. Клиенты могут быть как локальными (т. е. находиться на том же компьютере, что и SQL Server), так и сетевыми (обращаться к SQL Server по сети).

Если обращение производится по сети, то на клиентском компьютере должна быть установлена клиентская сетевая библиотека, соответствующая установленной серверной. Установку клиентской сетевой библиотеки на клиентский компьютер можно произвести при помощи мастера установки SQL Server 2005 (в списке компонентов нужно выбрать **Client Connectivity** (Совместимость клиентов), а затем — нужную сетевую библиотеку). Однако чаще всего этим заниматься нет необходимости, поскольку клиентские приложения используют драйверы OLE DB, ODBC или BDE для подключения к

SQL Server. Эти драйверы подключаются к серверной сетевой библиотеке TCP/IP.

В SQL Server 2005 предусмотрено четыре сетевые библиотеки.

- **Shared Memory** (в списке в SQL Computer Management выглядит как **Sm**) — эта библиотека используется только для локальных подключений (т. е. подключений с того же компьютера, на котором установлен SQL Server 2005), как правило, утилитами администрирования. Это новая сетевая библиотека, которой не было в предыдущих версиях SQL Server. По умолчанию она включена во все редакции SQL Server 2005.
- **Named Pipes (Np)** — именованный канал в оперативной памяти, в который один процесс передает информацию, а другой — считывает. Может использоваться как для локальных, так и для удаленных подключений. Чаще всего он используется для локального подключения утилит администрирования предыдущих версий SQL Server, например, SQL Server 2000. По умолчанию для всех редакций SQL Server 2005 эта библиотека отключена.
- **TCP/IP (Tcp)** — самая популярная сетевая библиотека, которая используется в подавляющем большинстве случаев и почти всеми клиентскими приложениями. Как ни удивительно, но в SQL Server 2005 она по умолчанию включена только в редакции Enterprise Edition. Для всех остальных редакций ее придется включать вручную после установки.
- **VIA (Via, Virtual Interface Adapter — адаптер виртуального интерфейса)** — экзотическая сетевая библиотека, которая используется только со специальным сетевым оборудованием. Также по умолчанию отключена во всех редакциях SQL Server 2005.

Обратите внимание, что в SQL Server 2005 больше нет поддержки таких сетевых библиотек, как Multiprotocol, NWLink IPX/SPX, AppleTalk и Banyan VINES. Библиотека Multiprotocol, в которой реализованы специальные алгоритмы шифрования данных, широко используется на отечественных предприятиях, и отказ от ее поддержки может стать неприятным сюрпризом для тех, кто собирается использовать SQL Server 2005.

На всякий случай подчеркнем еще раз: **если вы используете любую редакцию SQL Server 2005, отличную от Enterprise Edition, то для обеспечения работоспособности многих клиентских приложений вам придется после установки вручную включить сетевую библиотеку TCP/IP!**

Как же физически настроить сетевые библиотеки? В SQL Server 2000 для этой цели использовалось специальное средство под названием Server Network Utility. В SQL Server 2005 его больше нет. Вместо этого необходимо использовать утилиту под названием SQL Server Configuration Manager. Ее

можно запустить из меню **Пуск | Программы | Microsoft SQL Server 2005** или из консоли Computer Management (контейнер **Services and Applications** (Службы и приложения)). Основное предназначение этой утилиты — управление службами SQL Server 2005, а также серверными и клиентскими сетевыми библиотеками. Работа с серверными сетевыми библиотеками производится из контейнера **Server Network Configuration** (Сетевая конфигурация сервера), а клиентскими — из контейнера **Client Network Configuration** (Сетевая конфигурация клиента). Вам необходимо выбрать нужную сетевую библиотеку (под контейнером **Protocols for имя_экземпляра**), в контекстном меню выбрать пункт **Properties** (Свойства) и установить нужные значения свойств (например, **Yes** (Да) для свойства **Enabled** (Включено)). Работа с SQL Server Configuration Manager будет рассматриваться в разд. 3.3.

И еще два момента, связанных с сетевыми библиотеками.

Во-первых, в протоколе TCP/IP используются порты — идентификаторы конечного процесса-получателя пакета. Очень часто администратору SQL Server необходимо предоставить администратору сети информацию о том, какие порты должны быть открыты на брандмауэре, или поменять используемый SQL Server порт (например, если этого требует политика безопасности вашей компании). По умолчанию SQL Server настроен так: экземпляр по умолчанию (*default instance*) занимает порт TCP 1433 для приема клиентских подключений, а именованные экземпляры работают с динамическими портами. Это значит, что выбирается любой порт TCP из свободных на компьютере. Клиентский компьютер обращается на порт 1433, и затем сетевая библиотека, занимающая этот порт, обнаружив, что клиент обращается к именованному экземпляру, перенаправляет его на нужный порт. Такое поведение не всегда удобно, поскольку требует открытия всех портов сервера на брандмауэре.

Поменять это поведение и настроить именованный экземпляр на использование определенного порта (или настроить экземпляр по умолчанию на другой порт) можно так:

- откройте SQL Server Configuration Manager;
- раскройте в нем контейнер **Server Network Configuration | Protocols for имя_экземпляра | Tcp**;
- в правой части экрана выберите нужный IP-адрес (они отображаются как **IP1**, **IP2** и т. п.; или **IPAll** — настройки для всех IP-адресов) и из контекстного меню откройте его свойства, а дальше определите значение свойства **TcpPort**.

Чтобы настройки вступили в силу, вам придется перезапустить SQL Server 2005.

Если для экземпляра SQL Server 2005 по умолчанию указать в качестве используемого порта порт 0, то он будет вести себя как именованный экземпляр, выбирая номер порта случайным образом из числа свободных на компьютере.

SQL Server 2005 использует еще один порт: UDP 1434. Этот порт используется службой SQL Browser для обнаружения установленных серверов SQL Server в сети и формирования их списка. Если вам необходимо получать список работающих в сети серверов SQL Server, на брандмауэрах необходимо открыть и этот порт.

И еще один важный момент. По умолчанию SQL Server 2005 обменивается с клиентами пакетами в формате TDS (Tabular Data Stream — табличный поток данных), которые совершенно не защищены от перехвата. Если кто-то в вашей сети перехватит эти пакеты, он сможет прочитать и запросы пользователя, и данные, которые возвратит сервер. В предыдущих версиях SQL Server могли шифровать трафик сетевого взаимодействия с SQL Server двумя способами: средствами сетевой библиотеки MultiProtocol и при помощи SSL (т. е. используя сертификаты). В SQL Server 2005 остался только один способ — применение SSL, но функциональность этого способа значительно расширена. Подробнее про применение SSL для защиты сетевого трафика SQL Server 2005 будет рассказано в разд. 5.6.2.

2.5.3. Другие послеустановочные задачи

К другим послеустановочным задачам можно отнести:

- создание пользовательской базы данных для поддержки вашего приложения, настройка ее параметров и создание в ней необходимых объектов;
- создание учетных записей для подключения к SQL Server, пользователей баз данных и настройка разрешений;
- обеспечение отказоустойчивости SQL Server 2005: настройка резервного копирования, зеркального отображения баз данных и т. п.;
- создание системы мониторинга и оптимизация производительности для вашего приложения;
- развертывание клиентских приложений, их настройка и тестирование.

Скорее всего, вам потребуются и другие действия, которые зависят от вашего приложения. Подробнее про них будет рассказано в следующих главах этой книги.

Задание для самостоятельной работы 2.1.

Установка именованного экземпляра SQL Server 2005

Подготовка компьютера:

На компьютере, который будет использоваться для этого задания, должен быть установлен Windows Server 2003 Enterprise Edition SP1 со следующими параметрами:

- региональные настройки — русские;
- в числе компонентов Windows должен быть установлен Application Server с параметрами по умолчанию (поскольку для работы Reporting Services нужен Internet Information Server);
- должен быть установлен SQL Server 2000 Enterprise Edition как экземпляр по умолчанию (он потребуется для выполнения задания по переносу существующих баз данных на SQL Server 2005). Для него должен быть выбран полный набор компонентов.

Все остальные параметры Windows Server 2003 Enterprise Edition оставлены по умолчанию. Для целей этого задания компьютер называется LONDON.

Задание:

Установите на компьютер SQL Server 2005 Enterprise Edition со следующими параметрами:

- должен быть установлен полный набор компонентов;
- SQL Server 2005 должен быть установлен как именованный экземпляр в дополнение к уже установленному SQL Server 2000. Именованный экземпляр должен называться SQL2005;
- все службы SQL Server 2005 должны работать от имени локальной системной учетной записи;
- режим аутентификации — **Mixed**;
- пароль для учетной записи SA — P@ssw0rd;
- кодировка по умолчанию — **Cyrillic_General**;
- не посыпать сообщения об ошибках в Microsoft.

Для остальных параметров оставьте значения по умолчанию.

После окончания установки подключитесь к SQL Server 2005 из консоли SQL Server Management Studio.

Решение:

1. Из корневого каталога компакт-диска с дистрибутивом SQL Server 2005 запустите файл setup.exe.
2. Согласитесь с лицензионным соглашением и на экране **SQL Server Component Update** произведите установку необходимых для установки SQL Server 2005 компонентов. По окончании установки нажмите кнопку **Finish** (Завершить).
3. На первом экране мастера установки SQL Server нажмите **Next** (Дальше).
4. На экране **System Configuration Check** (Проверка конфигурации системы) убедитесь, что нет ни ошибок (Error), ни предупреждений (Warning). Нажмите кнопку **Continue** (Продолжить).
5. На экране **Registration Information** (Информация о регистрации) введите ваше имя и название организации, а затем введите серийный номер (**Product Key**) и нажмите кнопку **Next**.
6. На следующем экране **Components to Install** (Компоненты для установки) установите флажки напротив всех компонентов SQL Server 2005 (флажки **Create a failover cluster** должны остаться неактивными). Затем нажмите кнопку **Advanced**, раскройте контейнер **Client Components** (Клиентские компоненты), щелкните по строке **Documentation, Samples and Sample Databases** (Документация, примеры и учебные базы данных) и выберите значение **Entire Feature will be installed on local hard drive** (Весь компонент будет установлен на локальном жестком диске), чтобы были установлены учебные базы данных и примеры. Нажмите кнопку **Next**.
7. На экране **Instance Name** (Имя экземпляра) переставьте переключатель в положение **Named Instance** (Именованный экземпляр) и введите имя именованного экземпляра **SQL2005**. Нажмите кнопку **Next**.
8. На экране **Service Account** (Учетная запись службы) установите переключатель в положение **Use the built-in System Account** (Использовать встроенную системную учетную запись) и в списке справа выберите **Local system** (Локальная системная). Установите флажки для всех служб в разделе **Start services at the end of setup** (Запускать службы по окончании установки) и нажмите кнопку **Next**.

Примечание

Как было сказано ранее, для рабочих серверов рекомендуется использовать только доменные учетные записи. Однако, поскольку это требует установки домена, то в этом задании для простоты мы используем локальную системную учетную запись для служб SQL Server.

9. На экране **Authentification Mode** (Режим аутентификации) установите переключатель в положение **Mixed Mode** и введите пароль для учетной записи sa. По условию задания пароль должен выглядеть как P@ssw0rd.
10. На экране **Collation Settings** (Настройки сопоставления) оставьте выбранное по умолчанию значение **Cyrillic_General**.
11. На экранах **Report Server Installation Options** (Параметры установки Report Server) оставьте переключатель в положении **Install the default configuration** (Установить конфигурацию по умолчанию) и нажмите кнопку **Next**.
12. На экране **Error Reporting** (Отчеты об ошибках) снимите оба флажка, а затем на экране **Ready to Install** (Готовность к установке) нажмите **Install** (Установить).
13. После окончания установки в меню **Пуск | Программы | Microsoft SQL Server 2005** выберите **SQL Server Management Studio**. Откроется SQL Server Management Studio с окном **Connect to Server** (Подключиться к серверу). Оставьте в поле **Server Type** (Тип сервера) значение **SQL Server**, в поле **Server Name** (Имя сервера) введите *имя_вашего_компьютера\имя_экземпляра* (например, LONDON\SQL2005), в списке **Authentification** выберите **Windows Authentification** и нажмите кнопку **Connect**. В окне **Object Explorer** откроется установленный вами SQL Server 2005.



ГЛАВА 3

Средства администрирования SQL Server 2005

Обычно следующее действие после установки SQL Server — это создание базы данных, с которой будет работать ваше приложение. Однако в SQL Server 2005 система средств администрирования поменялась настолько сильно, что вполне можно растеряться, столкнувшись с ней в первый раз. В SQL Server 2005 вы не найдете ни Enterprise Manager, ни Query Analyzer, ни Analysis Manager, ни многих других привычных программ и утилит. Поэтому в этой главе мы рассмотрим основные средства для работы с SQL Server 2005 и познакомимся с их возможностями.

3.1. SQL Server Management Studio

3.1.1. Что такое SQL Server Management Studio

SQL Server Management Studio — это главный рабочий инструмент администратора в SQL Server 2005. В нем объединены возможности Enterprise Manager, Query Analyzer (с возможностью создания запросов MDX и XQuery), Analysis Manager, средств администрирования Reporting Services и Notification Services, а еще Visual Studio (поскольку при создании скриптов теперь используется проектный подход). Собственно говоря, в основу SQL Server Management Studio легла именно среда разработки Visual Studio, что хорошо видно по структуре его окон. Предложение разработчиков Microsoft администрировать SQL Server 2005 из Visual Studio выглядит несколько необычным, но привыкнуть к новому интерфейсу можно достаточно быстро.

В первых бета-версиях SQL Server 2005 вместо SQL Server Management Studio использовалось название "SQL Workbench", что осталось в названии исполняемого файла (sqlwb.exe) и в некоторых служебных сообщениях.

Запустить SQL Server Management Studio можно, конечно, из системного меню **Пуск | Программы | Microsoft SQL Server 2005**. Второй вариант — воспользоваться командой `sqlwb` в командной строке.

Рассмотрим еще несколько моментов, связанных с применением Management Studio.

Если вы привыкли пользоваться клавиатурными комбинациями в Enterprise Manager, то вы будете удивлены, узнав, что в SQL Server Management Studio они сильно изменены. Это было сделано специально для того, чтобы сделать набор клавиатурных комбинаций максимально похожим на используемый в Visual Studio. Если вы хотите вернуться к тому набору, который был в Enterprise Manager, необходимо в меню **Tools** (Сервис) выбрать **Options** (Настройки), раскрыть контейнер **Environment** (Рабочая среда), выделить строку **Keyboard** (Клавиатура) и вместо раскладки **Standard** (Стандартная), которая соответствует Visual Studio и установлена по умолчанию, выбрать **SQL Server 2000**.

Еще один способ — это назначить вручную удобные вам клавиши для выполнения различных команд SQL Server Management Studio (можно даже назначить горячую клавишу хранимой процедуре из базы данных SQL Server). Обычно для этой цели используется меню **Tools | Customize** (Сервис | Настроить) (а затем в открывшемся окне кнопка **Keyboard**), но есть и другие способы. Подробнее о них написано в статье "Customizing Menus and Shortcut Keys" (Настройка меню и клавиатурных комбинаций) в Books Online.

Окна в SQL Server Management Studio можно прятать, менять их поведение (например, делать плавающими или пристыкованными) и т. п. Чтобы отобразить закрытое окно, можно воспользоваться меню **View** (Вид). Если в процессе настроек вы пришли к совсем неудобному варианту, можно восстановить конфигурацию окон, определенную по умолчанию, при помощи меню **Window | Reset Window Layout** (Окно | Вернуть исходное расположение окон).

Далее рассказывается о главных окнах SQL Server Management Studio.

3.1.2. Окно *Registered Servers*

В окне **Registered Servers** (Зарегистрированные серверы) (рис. 3.1) представлены, как и в Enterprise Manager предыдущих версий SQL Server, те серверы SQL Server, о которых знает Management Studio (как версии SQL Server 2005, так и предыдущих версий).

Экземпляры SQL Server, которые расположены на том же компьютере, где запущена Management Studio, появляются в этом окне автоматически. Для

серверов SQL Server, расположенных на других компьютерах, сначала необходимо создать регистрацию (**New | Server Registration** (Новый | Регистрация сервера) в контекстном меню Microsoft SQL Servers). Возможностей при создании регистрации намного больше, чем в SQL Server 2000. Вы можете для сервера указать свое имя, под которым он будет виден в окне **Registered Servers**, описание, базу данных, к которой вы будете подключаться по умолчанию, сетевой протокол, размер пакета и тайм-ауты. Кроме того, можно установить принудительное шифрование всего сетевого трафика Management Studio к этому серверу.

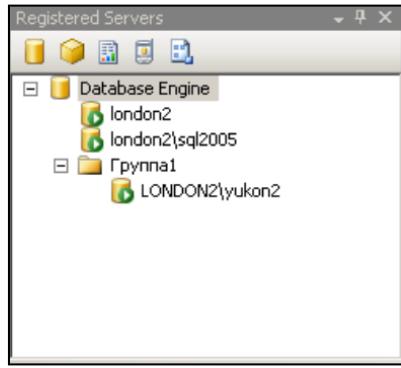


Рис. 3.1. Окно Registered Servers в SQL Server Management Studio

Новой возможностью этого окна является то, что вы можете добавлять регистрации не только для самих серверов SQL Server, но и для их компонентов: серверов Analysis Server, Report Server, среды выполнения DTS (DTS Server) и для серверов SQL Server, установленных на наладочных компьютерах и смартфонах (SQL Mobile). При создании регистрации вы можете выбрать тип сервера, к которому будет производиться подключение. Чтобы отобразить нужный тип серверов, необходимо воспользоваться кнопками на панели инструментов окна **Registered Servers**.

Из окна **Registered Servers** очень удобно открывать выбранный вами сервер в **Object Explorer** (Проводник объектов) или открывать для него уже подключенное к серверу окно редактора кода скриптов. Выполняются эти операции из контекстного меню сервера, например, **Connect | New Query** (Подключиться | Новый запрос) или **Connect | Object Explorer** (Подключиться | Проводник объектов).

Еще одной новой возможностью этого окна является то, что вы можете экспортить информацию о всех зарегистрированных в этом окне серверах (со всеми параметрами подключения, включая настройки сетевых библиотек, пароли и т. п.) в файл XML, а затем импортировать эту информацию в

Management Studio, например, на другом компьютере, для переноса всех настроек. Эта операция выполняется при помощи команд **Import** и **Export** из контекстного меню элементов верхнего уровня (например, для **Database Engine**) в этом окне.

3.1.3. Окно *Object Explorer*

По умолчанию сразу под окном **Registered Servers** располагается окно **Object Explorer**. Фактически это окно представляет собой Enterprise Manager, который встроен в Management Studio. Основной объем административных операций с серверами, базами данных и объектами баз данных производится при помощи этого окна.

По сравнению с Enterprise Manager, дерево объектов серверов SQL Server в этом окне стало глубже: теперь в нем можно дойти до отдельных столбцов в таблице, ограничений целостности, триггеров, индексов и даже статистики. Например, информация для таблицы `Sales.Store` учебной базы данных AdventureWorks выглядит так, как представлено на рис. 3.2.

Контейнеры базы данных также существенно изменились, чтобы отразить новую функциональность SQL Server 2005. Например, контейнер базы данных AdventureWorks представлен на рис. 3.3.

Если в дереве **Object Explorer** отображается очень много объектов, а вам нужны только некоторые из них, вы можете отфильтровать отображаемые объекты при помощи кнопки **Filter** (Фильтр) на панели инструментов этого окна. При помощи другой кнопки вы можете выбрать режим сортировки — по типам объектов или по схеме.

Различные мастера (копирования базы данных, создания плана обслуживания и т. п.) теперь можно запускать только из контекстного меню в **Object Explorer**. Специального пункта в меню **Tools** в SQL Server Management Studio уже не предусмотрено.

Параметров, которые можно настроить из **Object Explorer**, очень много. Мы рассмотрим их в соответствующих разделах (например, свойства баз данных приводятся в следующей гл. 4). Отметим еще одну особенность **Object Explorer**: это наследник не только Enterprise Manager, но и **Object Explorer** из Query Analyzer в SQL Server 2000. Поэтому в этом окне вы можете не только выполнять различные операции на графическом экране, но и создавать скрипты разных типов для самых разных объектов. Например, для таблиц можно создавать скрипты на создание таблицы, ее удаление, выборку, вставку, изменение и удаление данных, для ограничений целостности — на создание, удаление и т. п.

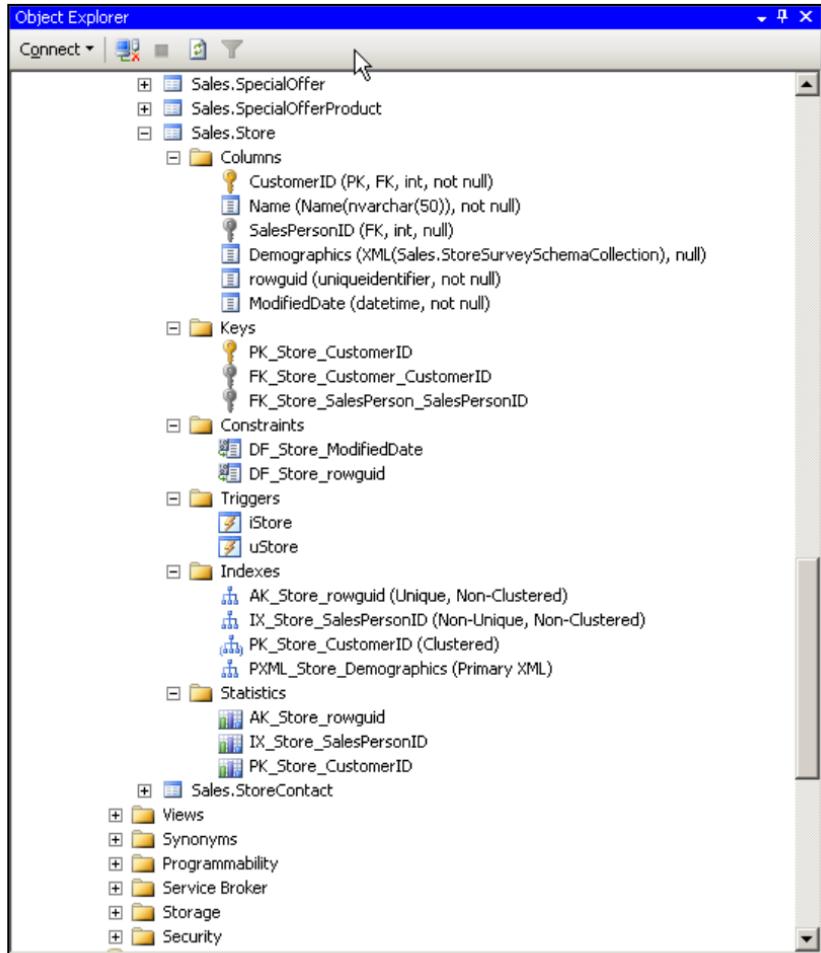


Рис. 3.2. Информация о таблице в окне Object Explorer

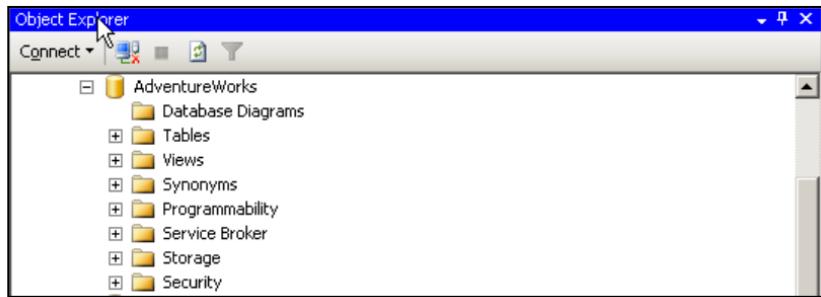


Рис. 3.3. Контейнер базы данных AdventureWorks в Object Explorer

3.1.4. Окно *Document*

Это главное окно SQL Server Management Studio. Оно используется для просмотра сводной информации для объектов, выбранных в **Object Explorer**, для написания скриптов и просмотра их результатов, отображения статей справки, вывода информации о статистике работы сервера и т. п.

Это окно может работать в двух вариантах:

- Tabbed** — когда это окно разделено на вкладки. На каждой вкладке отображается своя информация. Этот режим используется по умолчанию;
- MDI** — когда для каждого документа (скрипта, статьи справки и т. п.) открывается свое отдельное окно, которое можно перемещать по экрану Management Studio.

Настроить удобный вам режим отображения информации можно при помощи меню **Tools | Options**, контейнера **Environment | General** (Рабочая среда | Общие), группы элементов управления **Environment Layout** (Схема рабочей среды).

Работа в окне **Document** — это фактически работа с элементами, которые в нем отображаются: сводная информация объектов, скрипты, статьи справки. Работа со скриптами будет рассмотрена *разд. 3.1.7*. Отметим только, что просмотреть сводную информацию для выделенного объекта в **Object Explorer** можно при помощи клавиши **<F7>**.

3.1.5. Окно *Solution Explorer*

Ничего похожего на это окно еще не было в предыдущих версиях SQL Server. Окно **Solution Explorer** (Проводник проектов) (рис. 3.4) пришло из Visual Studio.

Его появление отражает важную новую возможность: скрипты (например, на языке SQL или MDX) теперь можно организовывать в проекты — точно так же, как это делается с модулями кода и формами в Visual Studio. Проекты, в свою очередь, можно организовывать в решения (*solutions*). Это дает ряд преимуществ:

- логически связанные между собой скрипты (например, относящиеся к одной задаче) можно объединять вместе;
- можно настроить свойства решения и отдельного проекта;
- вместе с файлами скриптов, которые обычно составляют основное содержание проектов SQL Server Management Studio (поддерживаются скрипты SQL, MDX и SQL Server Mobile), можно хранить дополнительную инфор-

мацию, например, информацию о подключениях, которая будет использоваться для этих скриптов по умолчанию, или любые другие файлы (например, XML, BCP, TXT и т. п.). Эти "посторонние" файлы будут видны из контейнера **Miscellaneous** (Разное). Добавить их можно, воспользовавшись командой **Add | Existing Item** (Добавить | Существующий элемент) контекстного меню проекта;

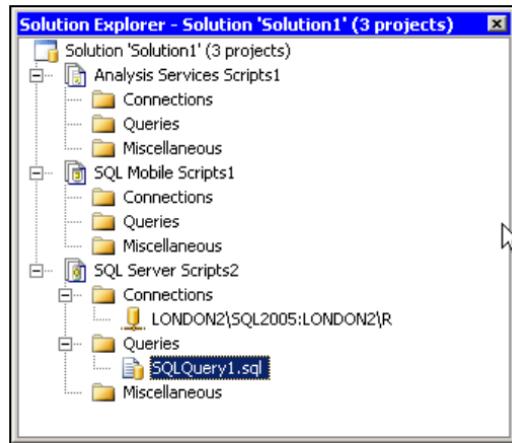


Рис. 3.4. Окно **Solution Explorer**

- решения и проекты можно сохранять в базе данных Visual Source Safe (это специальное программное средство от Microsoft, предназначенное для организации совместной работы над кодом многих программистов). Это средство обычно использовалось вместе с Visual Studio. Теперь появилась возможность использовать его и для скриптов SQL и MDX.

Конечно же, оставлена и возможность работать с отдельными скриптами, не входящими в решения и проекты, как это было в предыдущих версиях SQL Server.

Сама работа с **Solution Explorer** выглядит очень просто. Это окно можно сделать видимым при помощи меню **View** или клавиатурной комбинацией <Ctrl>+<Alt>+<L>. Затем при помощи меню **File | New** (Файл | Новый) создайте новый проект. По умолчанию для создаваемого проекта будет создано новое решение. Если вы хотите добавить проект к уже существующему решению, проект нужно создавать из контекстного меню этого решения в окне **Solution Explorer**.

После того как решение и проект будут созданы, при помощи контекстного меню этих объектов в **Solution Explorer** можно добавлять новые скрипты и другие элементы проекта.

Информация решения сохраняется в двух файлах:

- **с расширением ssmsln** — файл, содержащий в основном информацию о проектах, которые входят в решение, а также о файлах информации проектов с указанием пути к ним. Это текстовый файл, который можно изменять самостоятельно;
- **с расширением sqlsuo** — двоичный файл, определяющий различные пользовательские параметры SQL Server Management Studio, настроенные при работе с данным решением.

Информация о проектах скриптов SQL сохраняется в файлах с расширением ssmsqlproj (для проектов скриптов MDX — ssmsasproj, для проектов SQL Server Mobile — ssmsmobileproj). В этих XML-совместимых файлах хранится информация о параметрах, настроенных вами для свойств проектов, а также о всех элементах каждого проекта (скриптах, подключениях и прочих файлах).

По умолчанию информация решений и проектов помещается в каталог Мои документы\SQL Server Management Studio. Для перемещения решения или проекта на другой компьютер достаточно просто скопировать весь каталог средствами Windows Explorer.

3.1.6. Другие окна SQL Server Management Studio

Остальные окна SQL Server Management Studio пришли или из Visual Studio, или из средств администрирования предыдущих версий SQL Server.

Окно **Template Explorer** (Проводник шаблонов) пришло из Query Analyzer в SQL Server 2000. Это окно позволяет найти нужный шаблон, т. е. заготовку скрипта для выполнения определенной операции. В шаблоне достаточно изменить лишь необходимые параметры (значения в угловых скобках) — и скрипт готов. Как правило, это удобнее, чем копировать примеры из справки.

Обратите внимание, что вручную заменять значения в угловых скобках вам совсем не требуется. Намного удобнее воспользоваться для этой цели окном **Specify Values for Template Parameters** (Указать значения для параметров шаблонов) (рис. 3.5). Эта окно можно открыть при помощи одноименной команды из меню **Query** (Запрос).

Библиотека шаблонов, которая поставляется с SQL Server 2005, является расширяемой. Вы легко можете добавлять в нее свои шаблоны. Для этого достаточно просто создать новый файл с расширением sql и перенести его в нужный каталог. Каталог, в котором находятся файлы шаблонов, по умолчанию доступен по пути C:\Program Files\Microsoft SQL Server\90\Tools\Binn\VSShell\Common7\IDE\sqlworkbenchnewitems.

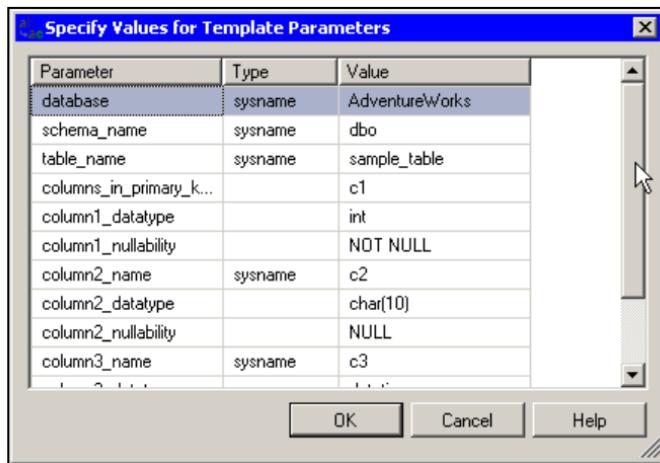


Рис. 3.5. Окно подстановки значений параметров

В отличие от библиотеки шаблонов Query Analyzer в SQL Server 2000, с SQL Server Management Studio поставляются, кроме шаблонов скриптов SQL, также шаблоны для скриптов MDX и SQL Server Mobile.

По умолчанию окно **Template Explorer** закрыто. Открыть его можно при помощи меню **View** или клавиатурной комбинацией <Ctrl>+<Alt>+<T>.

Окна **Properties** (Свойства) в SQL Server Management Studio бывают двух видов:

- окна свойств, которые предназначены для административных целей. Эти окна открываются из контекстного меню объектов в **Object Explorer**, например, баз данных, таблиц или представлений. Структура и наполнение таких окон полностью зависят от типа выбранного объекта. Содержание этих окон мы будем рассматривать в главах, посвященных соответствующим объектам;
- окна свойств, которые предназначены для разработчиков и сделаны по образцу окон свойств в Visual Studio. Эти окна доступны, например, для решений, проектов или скриптов. Их можно открыть из контекстного меню этих объектов или при помощи меню **View**. Большая часть параметров в этих окнах доступна только на чтение, и работа с ними не вызывает каких-либо проблем.

Окно **Toolbox** (Элементы управления) делает то же самое, что и в Visual Studio — позволяет перетаскивать в нужное место графические элементы управления. Однако поскольку в SQL Server Management Studio никаких графических форм не предусмотрено, оно используется очень редко. Например, при помощи этого окна можно добавить новые элементы в планы обслуживания баз данных (*database maintenance plans*).

Окно **Bookmark** (Закладки) также пришло из Visual Studio. В этом окне отображается список закладок, которые могут относиться к самым разным скриптам. Расставлять закладки можно при помощи панели инструментов **Text Editor** (Редактор текста) (этот панель можно сделать видимой при помощи меню **View | Toolbars | Text Editor** (Вид | Панели инструментов | Текстовый редактор)).

При необходимости в SQL Server Management Studio открываются и другие окна: Internet Explorer, результат поиска, результат выполнения запросов и т. п. Они вполне очевидны, и рассматривать их мы здесь не будем.

3.1.7. Приемы работы со скриптами

Любому администратору и разработчику приходилось писать сотни скриптов SQL. Кажется, что может быть проще? Однако опыт общения со слушателями на курсах показывает, что даже опытные специалисты часто не знают о всех средствах, которые можно использовать при создании скриптов в Query Analyzer в SQL Server 2000. А в SQL Server Management Studio появились новые возможности в этой области.

Первое, о чём необходимо сказать, — что часто совершенно нет необходимости создавать скрипт с нуля. Можно сэкономить время, если воспользоваться средствами автоматической генерации кода скриптов.

□ Первая возможность, о которой мы уже говорили ранее, — воспользоваться готовыми шаблонами (встроенными или добавленными вами) при помощи **Template Explorer**. Во встроенной библиотеке шаблонов предусмотрены скрипты для самых разных ситуаций. Особенно удобно то, что даже исправлять код шаблона, подстраивая его под вашу ситуацию, можно в автоматическом режиме, используя диалоговое окно **Specify Values for Template Parameters** (см. рис. 3.5), которое открывается при помощи команды **Specify Values for Template Parameters** из меню **Query**.

□ Второй вариант — воспользоваться средствами автоматической генерации скриптов из **Object Explorer** (рис. 3.6).

Например, для таблицы можно сгенерировать скрипты на её создание (для создания похожей таблицы), удаление, выборку данных (будут перечислены все столбцы таблицы), добавление новых данных, изменение существующих записей и удаление старых. Если столбцов в таблице много и вы не помните наизусть все их имена, такой подход может сэкономить много времени.

□ Третий вариант (с точки зрения автора, самый удобный) — воспользоваться графическим построителем запросов **Query Designer** в SQL Server Management Studio. Это средство особенно удобно в тех ситуациях, когда

вам нужно создать большой запрос со множеством соединений, условий и сортировок. С его помощью можно создавать очень сложные запросы, вообще не имея никакого представления о синтаксисе языка Transact-SQL.

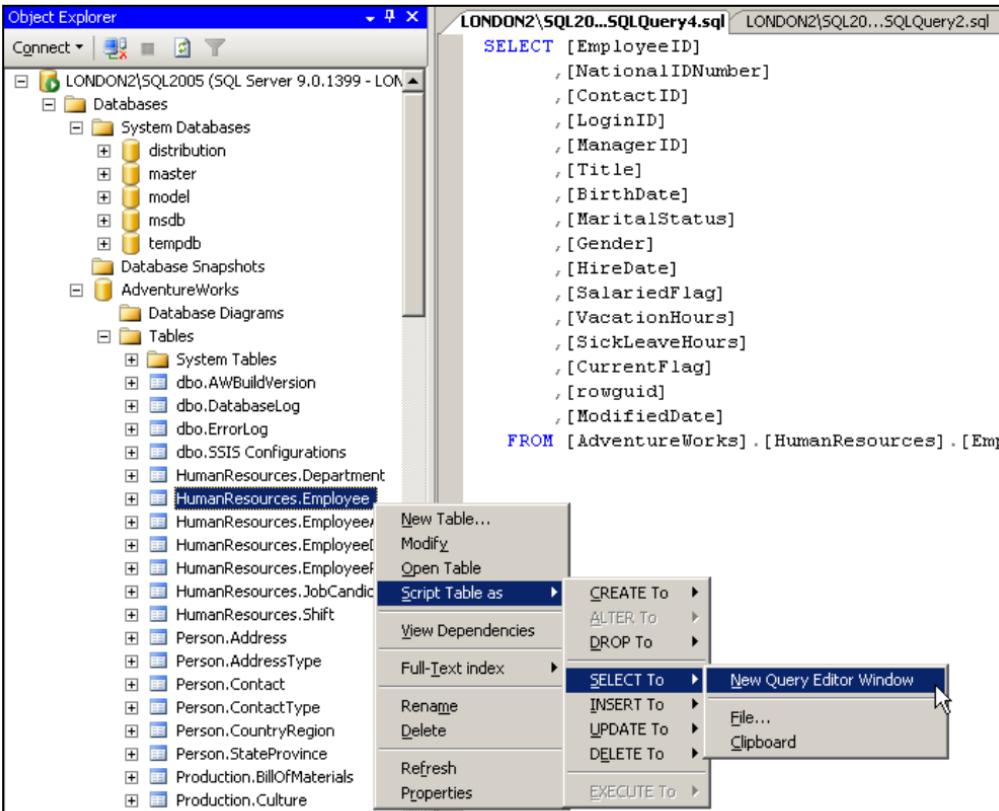


Рис. 3.6. Автоматическая генерация скриптов средствами SQL Server Management Studio

Проще всего использовать построитель запроса так: нужно создать в окне редактора скриптов пустой скрипт и щелкнуть правой кнопкой мыши по пустому пространству в этом окне. Затем в контекстном меню нужно выбрать пункт **Design Query in Editor** (Спроектировать запрос в редакторе) (рис. 3.7).

Откроется окно **Query Designer**, в котором можно будет выбрать нужные таблицы, столбцы в них, назначить столбцам псевдонимы, выбрать порядок сортировки, фильтры и т. п. (рис. 3.8).

Кроме того, если вы хотите создать скрипт для какой-либо административной операции (создание логина, предоставление разрешений и т. п.), то в вашем распоряжении — кнопка **Script** (Скрипт) в верхней части окна SQL Server Management Studio (рис. 3.9). При помощи этой кнопки можно автоматически

создать скрипт, в который будут подставлены введенные вами на графическом экране значения.

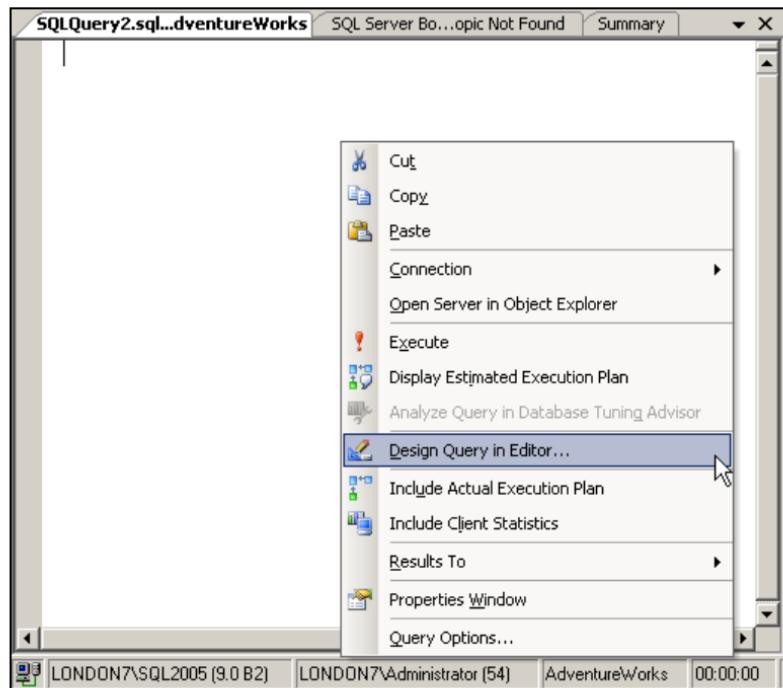


Рис. 3.7. Запуск построителя запросов

Одна из самых разрекламированных возможностей, которой не было в Query Analyzer и которая появилась в SQL Server Management Studio, — **Intellisense**, т. е. подсказка при вводе кода (она тоже пришла из Visual Studio). К сожалению, подсказка в Management Studio не работает в редакторе кода SQL, что резко снижает ее ценность. Она применяется только во вспомогательных редакторах, например, в редакторе кода XML (XML Editor). Настроить работу с подсказками можно при помощи меню **Tools | Options**, далее в дереве элементов нужно развернуть узел **Text Editor | All Languages | General** (Редактор текста | Все языки | Общие). Группа элементов **Statement Completion** (Завершение команд) как раз и отвечает за подсказки.

Отметим еще одну важную возможность. Если вам нужно просмотреть информацию какой-то таблицы или внести в нее изменения, то это можно сделать из SQL Server Management Studio, не обращаясь к коду Transact-SQL. Все это можно выполнить на интуитивно понятном графическом интерфейсе, аналогично тому, как это сделано в Access. Для того чтобы открыть встроенный редактор данных Management Studio, необходимо в окне **Object Explorer**

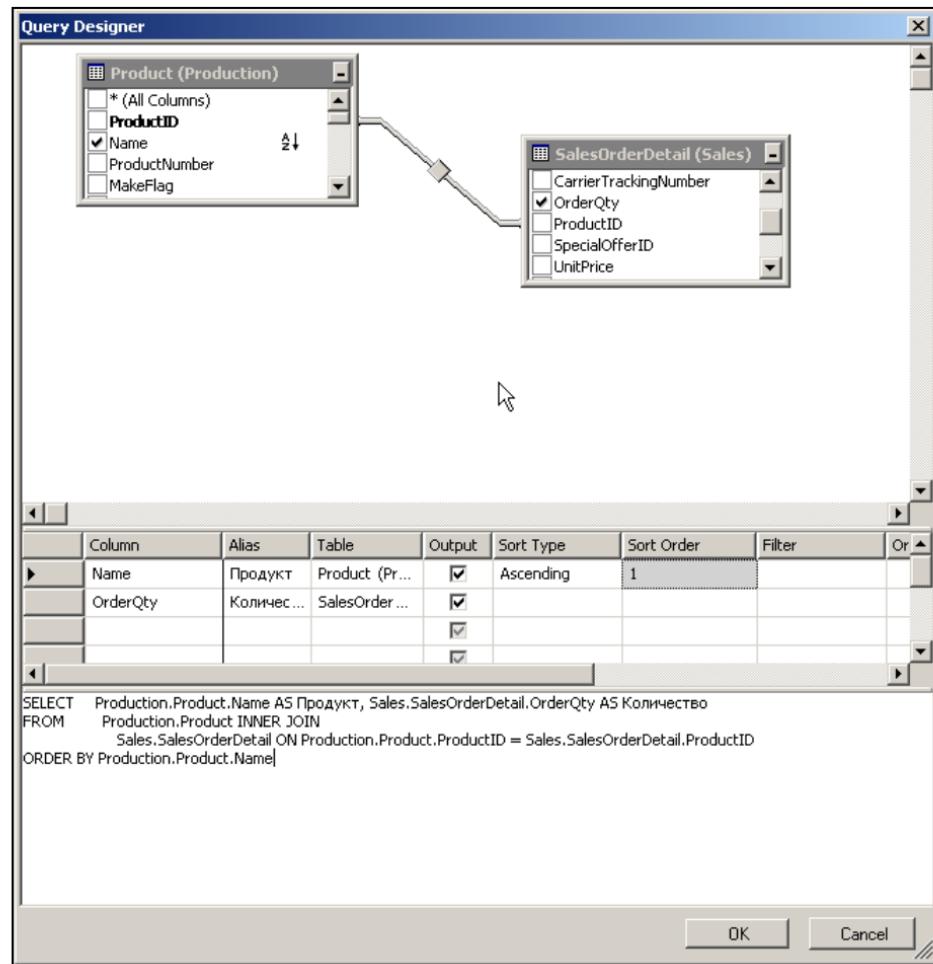


Рис. 3.8. Возможности построителя запросов

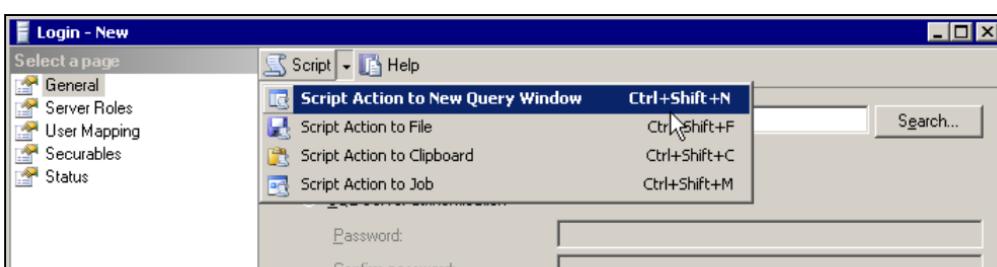


Рис. 3.9. Автоматическая генерация скриптов для действий, выполняемых средствами Management Studio

The screenshot shows the integrated data editor in SQL Server Management Studio. The title bar says "Table - Person.Contact". The table has columns: ContactID, NameStyle, Title, FirstName, MiddleName, and LastName. The data shows 18 rows of contact information. At the bottom, there are navigation buttons for the table, including arrows for row selection and a search bar with the number "1" and the text "of 19972".

	ContactID	NameStyle	Title	FirstName	MiddleName	LastName
▶	1	False	Mr.	Gustavo	NULL	Kossner
	2	False	Ms.	Catherine	R.	Torvall
	3	False	Ms.	Kim	NULL	De Luca
	4	False	Sr.	Humberto	NULL	Montgomery
	5	False	Sra.	Pilar	NULL	Alonso
	6	False	Ms.	Frances	B.	Chen
	7	False	Ms.	Margaret	J.	King
	8	False	Ms.	Carla	J.	Ernst
	9	False	Mr.	Jay	NULL	Spangler
	10	False	Mr.	Ronald	L.	Colmenares
	11	False	Mr.	Samuel	N.	Fuller
	12	False	Mr.	James	T.	Freeman
	13	False	Mr.	Robert	E.	King
	14	False	Mr.	François	NULL	LeBlanc
	15	False	Ms.	Kim	NULL	Lu
	16	False	Ms.	Lili	J.	Montgomery
	17	False	Ms.	Amy	E.	Wong
	18	False	Ms.	Janet	NULL	Fuller

Рис. 3.10. Встроенный редактор данных SQL Server Management Studio

щелкнуть правой кнопкой мыши по объекту таблицы и в контекстном меню выбрать пункт **Open Table** (Открыть таблицу). Откроется окно, аналогичное представленному на рис. 3.10, в котором вы можете просматривать и изменять данные таблицы.

3.2. Business Intelligence Development Studio

Business Intelligence Development Studio — это второе важнейшее графическое средство для работы с SQL Server 2005. Business Intelligence дословно переводится как "бизнес-разведка", и, вообще говоря, этот термин традиционно относится к технологии Data Mining — добычи данных. Поддержка этой технологии была реализована в Analysis Services, которые поставлялись с SQL Server 7.0 и 2000. Видимо, название Business Intelligence Development Studio отражает стремление Microsoft привлечь дополнительное внимание к этой возможности SQL Server 2005.

Business Intelligence Development Studio, как и SQL Server Management Studio, объединяет в себе возможности сразу нескольких программных средств, которые в предыдущих версиях SQL Server существовали по отдельности.

Главное слово здесь — *Development (разработка)*: это средство предназначено для работы с программными проектами. При помощи Business Intelligence Development Studio вы можете работать с проектами следующих типов:

- **проекты Analysis Services**, которые представляют собой базы данных OLAP с необходимыми компонентами: кубами, общими измерениями, моделями добычи данных и т. п.;
- **проекты Integration Services**, которые призваны заменить функциональность пакетов DTS предыдущих версий SQL Server. Вообще Integration Services — это компонент SQL Server, который в новой версии изменился, наверное, больше всех остальных. Работе с ним посвящена гл. 10;
- **проекты типа Report Project** — это отчеты к базам данных, которые раньше нужно было создавать средствами Report Designer. Теперь Reporting Services поставляются вместе с SQL Server 2005, а Report Designer интегрирован в Business Intelligence Development Studio;
- **проекты типа Report Model** — специальный тип отчета, предназначенный для того, чтобы наглядно представить структуру источника данных. Главные компоненты такого проекта — это Data Source Views (фактически это диаграммы баз данных, которые раньше создавались в Enterprise Manager, а теперь создаются из SQL Server Management Studio) и Report Models (описание сущностей, атрибутов и связей между ними в базе данных).

Интерфейс Business Intelligence Development Studio вряд ли заслуживает отдельного описания, поскольку при запуске этого приложения вам просто открывается среда разработки Visual Studio 2005. В ней вы должны создать или открыть проект нужного вам типа и работать с ним стандартными средствами Visual Studio.

3.3. SQL Server Configuration Manager

3.3.1. Что такое SQL Server Configuration Manager

SQL Server Configuration Manager — это еще одно новое средство администрирования SQL Server 2005. Запускается оно из системного меню **Пуск | Программы | Microsoft SQL Server 2005 | Configuration Tools | SQL Server Configuration Manager**. Оно объединяет в себе возможности Service Manager, Server Network Utility и Client Network Utility из SQL Server 2000. Каждой из этих бывших утилит соответствует свой контейнер Configuration Manager (рис. 3.11).

Первый контейнер **SQL Server 2005 Services** (Службы SQL Server 2005) ответственен за службы SQL Server 2005, второй контейнер **SQL Server 2005**

Network Configuration (Сетевая конфигурация SQL Server 2005) — за серверные сетевые библиотеки SQL Server, а третий контейнер **SQL Native Client Configuration** (Конфигурация SQL Native Client) — за параметры работы SQL Native Client. О каждом из них (и о компонентах SQL Server 2005, которыми управляет Configuration Manager) рассказывается в следующих разделах.

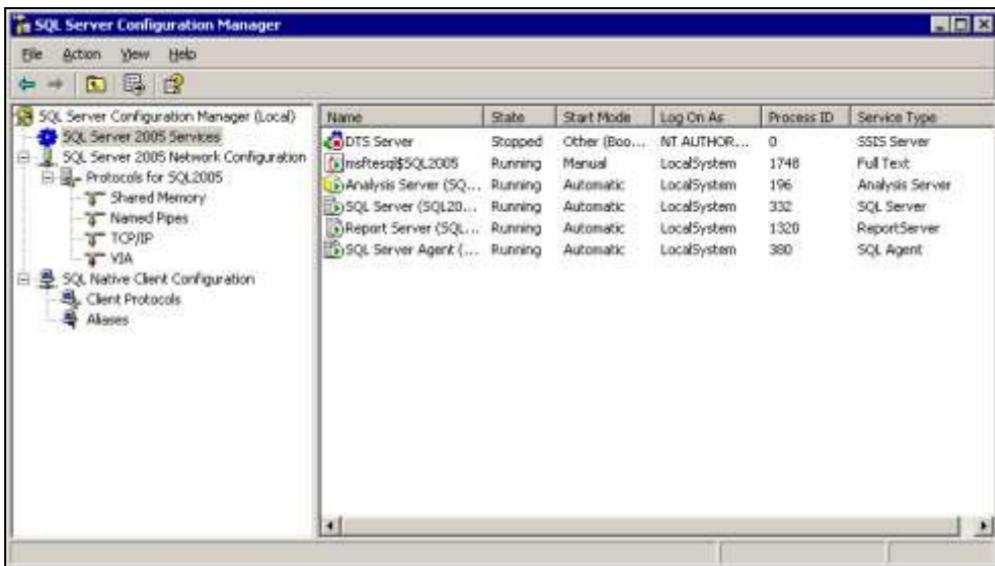


Рис. 3.11. Консоль SQL Server Configuration Manager

3.3.2. Службы SQL Server 2005

SQL Server 2005, как и все серверные продукты Microsoft, реализован в виде набора служб. Службы можно определить как специальные программы, которые работают от имени своей собственной учетной записи. Службы запускаются независимо от того, вошел ли пользователь в систему. Для каждой службы создаются специальные записи в разделе реестра `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services`.

На самом деле, разница между службами и обычными приложениями не так уж велика. Любое приложение Windows можно сделать службой. Для этого достаточно создать в реестре необходимые записи вручную или воспользоваться утилитами из набора Windows Server 2000 Resource Kit: мастером Service Installation Wizard (`svrinstw.exe`) с графическим интерфейсом или консольной утилитой `Srvany.exe`. И наоборот, многие службы можно запустить в режиме обычного приложения. Например, в режиме обычного прило-

жения можно запустить службы SQL Server 2000 и SQL Server 2005. Чтобы запустить SQL Server 2005 из командной строки, необходимо открыть командную строку в Windows, перейти в каталог установки SQL Server 2005 (по умолчанию это C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn) и выполнить команду `sqlservr`. При этом учетная запись пользователя, под которой вы работаете, должна обладать всеми необходимыми правами на вашем компьютере (входить в систему как служба, работать как часть операционной системы и т. п. — полный список был приведен в *разд. 2.2.4*), которыми по умолчанию не обладают обычные записи (даже с правами администратора). При запуске сервера из командной строки вы увидите пошаговый отчет о ходе запуска. Обычно SQL Server в таком режиме запускают либо для диагностики в случае сбоев при запуске, либо для проверки его поведения при работе под определенной учетной записью (например, чтобы выяснить, достаточно ли прав для выполнения определенных операций).

Какие службы входят в состав SQL Server 2005? Приведем их перечень с краткими комментариями.

- **SQL Server** — это сам SQL Server, ядро базы данных. Оно ответственно за работу с файлами базы данных, прием пользовательских подключений, выполнение запросов и т. п.
- **SQL Server Agent** — специальная служба, которая ответственна за автоматизацию работы с SQL Server. Она отвечает за выполнение заданий по расписанию, за предупреждения и другие служебные операции. Для хранения информации этой службы на SQL Server создается специальная служебная база данных MSDB. Обратите внимание: если вы принимали значения по умолчанию на экранах мастера установки SQL Server 2005, то эта служба автоматически запускаться не будет.
- **Report Server** — эта служба представляет серверный компонент Reporting Services. Она ответственна за генерацию отчетов, предоставление их пользователям, выполнение различных служебных операций с отчетами.
- **Analysis Server** — ядро сервера баз данных OLAP. Эта служба полностью ответственна за работу с базами данных OLAP и их компонентами, например, с кубами.
- **DTS Server** — это служба, ответственная за работу с новой средой DTS (т. е. за операции загрузки, выгрузки и преобразования данных, которые проводятся при помощи пакетов DTS).
- **msftesql** — эта служба раньше называлась Microsoft Search. Ее главная задача — работа с полнотекстовыми индексами (еще раз напомним, что SQL Server 2005 теперь поддерживает и русскоязычный полнотекстовый поиск).

В состав SQL Server 2005 входят еще две службы, но в Configuration Manager они почему-то не отображаются.

- **SQL Browser** — эта служба ответственна за формирование списка серверов SQL Server в сети.
- **SQL Writer** — работает с теневыми копиями (*shadow copies*) баз данных SQL Server 2005 и используется для проведения резервного копирования в оперативном режиме, без отключения пользователей.

Как вы видите, по сравнению с SQL Server 2000 добавилось несколько новых служб. В списке отсутствует служба Distributed Transaction Coordinator, которая входила в состав SQL Server 2000, но она никуда не делась: теперь она считается компонентом операционной системы.

Что нужно сказать про службы SQL Server 2005 с точки зрения администратора баз данных?

Во-первых, напомним, что во многих ситуациях предпочтительнее, чтобы служба SQL Server и служба SQL Server Agent работали от имени доменной учетной записи (*подробнее об этом см. разд. 2.2.4*).

Во-вторых, лишние службы можно отключить. На многих предприятиях вполне достаточно только службы SQL Server. Если вы не работаете с OLAP или Reporting Services или вам не нужны средства автоматизации SQL Server Agent, то для экономии системных ресурсов эти службы можно перевести в режим запуска вручную (*manual*). Запуск, отключение и изменение режима запуска служб SQL Server 2005 можно произвести как из консоли **Службы** в системном меню **Пуск | Программы | Администрирование** операционной системы, так и из Configuration Manager.

В-третьих, на вкладке **Advanced** (Дополнительно) свойств службы в Configuration Manager вы можете просмотреть или изменить многие важные параметры работы службы SQL Server 2005, например, информацию об установленных пакетах обновления, параметрах запуска и т. п.

3.3.3. Настройка серверных сетевых библиотек средствами SQL Server Configuration Manager

Еще одна важная функция SQL Server Configuration Manager — возможность настройки серверных и клиентских сетевых библиотек (то, что в SQL Server 2000 выполнялось при помощи утилит Server Network Utility и Client Network Utility). Сетевые библиотеки — это программные модули, при помощи которых происходит подключение клиентского программного обеспечения к серверу SQL Server. Про выбор серверных сетевых библиотек подробно рассказывалось в *разд. 2.5.2*, а здесь отметим только, что включить сетевую библиотеку можно из ее контекстного меню в Configuration Manager. Дополнительные возможности можно настроить, открыв свойства данной

сетевой библиотеки. Для наиболее часто используемой сетевой библиотеки **TCP/IP** можно настроить отдельные параметры для каждого IP-адреса, используемого для компьютера, на котором установлен SQL Server 2005. Эта операция производится в окне свойств объектов **IP1**, **IP2** и т. п., которые появляются в правой части консоли Configuration Manager, когда в левой части выбран протокол **TCP/IP**. Для всех IP-адресов, для которых не определены свои параметры, настройка производится при помощи объекта **IPAll**. По умолчанию используется только этот объект, все остальные объекты для конкретных IP-адресов отключены.

Самое важное, что можно определить для объекта конкретного IP-адреса или всех IP-адресов, — это номера портов, по которым будет принимать пользовательские соединения SQL Server. Экземпляр по умолчанию (*default instance*) SQL Server 2005, если вы не изменили исходные параметры, автоматически занимает порт 1433. Остальные (именованные) экземпляры SQL Server 2005 используют динамические порты, т. е. любые свободные порты на этом компьютере. При подключении пользователя к именованному экземпляру служба SQL Server Browser (работающая на порту 1434) сообщает приложению этого пользователя, какой порт каким экземпляром используется в настоящее время. Включить динамические порты для любого экземпляра можно очень просто: достаточно под контейнером **TCP/IP** для этого экземпляра раскрыть объект **IP** с нужным номером и напротив значения свойства **TCP Dynamic Ports** (Динамические порты) настроить значение 0. Однако чаще требуется выполнить обратную операцию — перевести SQL Server на работу со статическим портом (обычно этого требует конфигурация брандмауэра). Для настройки статического порта нужно просто ввести номер порта в свойствах объекта **IP** для параметра **TCP Port**. Некоторые клиенты после этого требуют внесения изменений в их конфигурацию: не забудьте проверить возможность их подключения к SQL Server 2005!

SQL Server 2005 можно сделать невидимкой в вашей сети, чтобы он не передавал информацию о себе другим компьютерам (и не появлялся в раскрывающихся списках серверов SQL Server). Настроить этот параметр (**HideInstance**) можно из свойств контейнера **Protocols for имя_экземпляра**. Отсюда же можно включить шифрование передаваемых по сети SQL Server данных средствами SSL и выбрать используемый для этого сертификат. Подробнее про защиту трафика SQL Server 2005 будет рассказано в разд. 5.6.2.

3.3.4. Настройка клиентских сетевых библиотек средствами SQL Server Configuration Manager. SQL Native Client

Еще одна возможность Configuration Manager — управление клиентскими сетевыми библиотеками, т. е. драйверами со стороны клиента, при помощи

которых производится подключение к серверным сетевым библиотекам (т. е. к SQL Server 2005). Сразу уточним следующий момент: все настройки, которые производятся из Configuration Manager для клиентских библиотек, относятся к SQL Native Client. Что это такое?

SQL Native Client — это набор программных объектов, которые поставляются с SQL Server 2005 и позволяют использовать новые возможности SQL Server 2005 (такие как MARS, который позволяет отправлять на SQL Server новые запросы, не дожидаясь возврата результатов выполнения предыдущих, работа с типом данных XML и т. п.). SQL Native Client использует стандартный набор драйверов и программных объектов MDAC (Microsoft Data Access Component — компоненты доступа к данным Microsoft). SQL Native Client можно считать надстройкой над стандартными интерфейсами доступа к данным OLE DB и ODBC. Этот набор программных объектов совместим с ADO, и его возможности можно использовать из ADO напрямую.

Устанавливать SQL Native Client на клиентские компьютеры или нет, полностью зависит от того, как написано приложение, которое будет обращаться к клиентскому компьютеру. Подавляющее большинство клиентских приложений ничего не знает о SQL Native Client и использует для подключения стандартные интерфейсы OLE DB, ODBC или BDE. На момент написания этой книги SQL Native Client был нужен только программам, поставляемым с самим SQL Server 2005, например, SQL Server Management Studio. Но даже если вашим программам SQL Native Client не нужен, некоторые параметры, настроенные для него (например, псевдонимы), понимают и приложения, работающие через OLE DB и ODBC. Поэтому, несмотря на то, что в большинстве ситуаций производить установку SQL Native Client (и последующую его настройку средствами SQL Server Configuration Manager) на клиентские компьютеры не нужно, вариант с установкой SQL Native Client всегда следует держать в голове. Установку SQL Native Client можно произвести либо при помощи программы установки SQL Server 2005, либо воспользовавшись файлом SQLNCLI.msi в каталоге SETUP на компакт-диске с дистрибутивом SQL Server 2005.

Для SQL Native Client из SQL Server Configuration Manager можно настроить следующие параметры:

- включить принудительное шифрование всего трафика, которым клиент будет обмениваться с SQL Server, и определить, стоит ли всегда доверять сертификату сервера (без обычной проверки этого сертификата в центре сертификации (*Certification Authority*));
- включить или отключить определенные сетевые библиотеки и настроить их свойства (например, для библиотеки **TCP/IP** настроить порт по умолчанию для обращения на SQL Server);
- настроить псевдонимы (*aliases*).

Про псевдонимы нужно сказать подробнее. Это замечательное средство, про которое часто забывают. Обычно псевдоним нужен тогда, когда в клиентском приложении жестко прописано имя сервера, к которому это приложение должно обращаться, а база данных перенесена на сервер с другим именем. В этом случае проще всего создать псевдоним на клиенте, при помощи которого клиент, обращаясь по старому имени, будет перенаправляться на новый сервер. Другая ситуация, когда вам может потребоваться псевдоним, — когда вы обращаетесь на SQL Server по нестандартному порту.

Псевдонимы, которые настраиваются средствами SQL Server Configuration Manager, работают не только для SQL Native Client, но и для подключений по OLE DB и ODBC.

3.4. SQLCmd

3.4.1. Применение SQLCmd

SQLCmd — это еще одна новая (и очень важная) утилита, входящая в состав SQL Server 2005. Она предназначена для выполнения скриптов Transact-SQL из командной строки и призвана заменить использовавшиеся для этой цели в предыдущих версиях SQL Server утилиты osql и isql. При этом утилита isql (которая использовала для подключения устаревшую библиотеку DBLibrary) вообще удалена из поставки SQL Server 2005, а osql (которая работает по ODBC) пока сохранена, но рекомендуется по возможности всегда использовать только SQLCmd. Сама утилита SQLCmd работает по OLE DB без использования SQL Native Client.

SQLCmd используется всегда, когда нужно выполнить команду Transact-SQL, скрипт или набор скриптов из командной строки операционной системы. Ситуации, когда вам пригодится эта утилита, могут быть такими:

- на SQL Server 2005 нужно создать и настроить базу данных. Произвести создание самой базы данных, объектов в ней, а также выполнить первоначальную загрузку данных удобнее всего при помощи скриптов SQLCmd (если вы не используете для этой цели резервные копии базы данных);
- необходимо внести обновления в существующую базу данных, например, изменить ее структуру. Если вы — разработчик, и вам нужно обеспечить единообразное внесение изменений, например, во всех филиалах, то применение SQLCmd может оказаться самым простым и надежным решением;
- когда нужно, чтобы выполнение определенных команд Transact-SQL инициировала операционная система (например, если вы используете планировщик операционной системы для выполнения каких-то действий на SQL Server по расписанию).

SQLCmd может работать в двух режимах: интерактивном и пакетном. При работе в интерактивном режиме SQLCmd запускается, и затем в открывшемся приглашении SQLCmd вводятся команды. В пакетном режиме вы сразу передаете SQLCmd нужный запрос или файл скрипта.

Полный синтаксис команд SQLCmd приводиться здесь не будет: его всегда можно посмотреть в документации. Приведем только самый распространенный вариант применения SQLCmd. Предположим, что вам нужно выполнить скрипт из файла C:\SQLQuery.sql и записать результаты в файл C:\Results.txt. Скрипт нужно выполнить на сервере LONDON\SQL2005, подключившись от имени пользователя SA с паролем P@ssw0rd. Команда при этом может быть такой:

```
sqlcmd -S LONDON7\SQL2005 -Usa -PP@ssw0rd -i C:\SQLQuery.sql -o  
C:\Results.txt
```

Отметим некоторые моменты, связанные с применением SQLCmd:

- скрипты для SQLCmd можно готовить и отлаживать в привычной графической среде SQL Server Management Studio (с поддержкой всех специальных команд и возможностей SQLCmd). Для этого достаточно перевести редактор кода в специальный режим написания скриптов SQLCmd. Эта операция производится из меню **Query | SQLCmd Mode** (Запрос | Режим SQLCmd). Правда, необходимо учитывать, что в этом режиме SQL Server Management Studio будет использовать для подключения SQL Native Client (в отличие от OLE DB, используемого SQLCmd): могут быть мелкие отличия в результатах выполнения скриптов;
- SQLCmd умеет использовать переменные окружения операционной системы. Если вы заранее создали на компьютере нужные вам переменные окружения, можно не указывать в скрипте, например, логин, используемый для подключения, пароль, имя сервера, имя базы данных и т. п. — всю необходимую информацию SQLCmd "подхватит" автоматически. Можно даже определить специальную переменную окружения sqlcmdini. Скрипт, который вы укажете при помощи этой переменной окружения, будет выполнен автоматически:

```
set sqlcmdini=C:\SQLQuery.sql  
sqlcmd
```
- SQLCmd можно использовать для выгрузки данных с SQL Server в формате XML (для выполнения команд SELECT с параметрами FOR XML и записи результатов в файл). Для этого необходимо перевести SQLCmd в специальный XML-режим с помощью команды :XML ON. Отключить этот режим можно командой :XML OFF.

3.4.2. Специальный режим подключения Dedicated Administrator Connection

При работе с предыдущими версиями SQL Server иногда возникала ситуация, когда какой-то некорректный запрос или клиентское соединение забирали все ресурсы сервера. Сервер переставал отвечать на запросы (в том числе и на запросы подключения от администратора), и выйти из этой ситуации можно было только при помощи перезапуска сервера. При этом, конечно, терялись все сеансы других пользователей.

SQL Server 2005 позволяет решить такие проблемы. При запуске SQL Server 2005 сразу резервирует ресурсы на одно подключение пользователя. Даже если какой-то запрос забрал все ресурсы, администратор сможет подключиться к серверу за счет резерва. После этого уже можно, например, закрыть проблемный сеанс при помощи команды KILL.

Средство для подключения к SQL Server 2005 за счет специально зарезервированных для этого ресурсов называется DAC (Dedicated Administrator Connection — выделенное административное подключение). Для того чтобы подключиться к серверу в этом режиме, используется команда SQLcmd с параметром -A, однако в окончательную версию SQL Server 2005 была добавлена возможность использовать для этой цели и SQL Server Management Studio. Чтобы подключиться в режиме DAC из SQL Server Management Studio, нужно выполнить следующие действия:

1. Нажмите кнопку **New Query** на панели инструментов и в раскрывшемся списке выберите **Database Engine Query** (Запрос к ядру базы данных). Откроется окно **Connect to Database Engine** (Подключение к ядру базы данных).
2. В поле **Server Name** вместо обычного имени сервера, например, LONDON\SQL2005 введите ADMIN:имя_экземпляра, например, ADMIN:LONDON\SQL2005.
3. Выберите режим аутентификации и подключитесь к серверу.

Подключение в режиме DAC обладает некоторыми специфическими особенностями:

- по умолчанию соединение в режиме DAC можно выполнить только с локального компьютера (т. е. с того компьютера, на котором работает SQL Server 2005). Чтобы разрешить такие соединения с удаленного компьютера, необходимо настроить для параметра сервера remote admin connection значение 1:

```
sp_configure 'remote admin connections', 1;
```

- в этом режиме к серверу одновременно может быть установлено только одно соединение;

- подключение в режиме DAC может производиться только от имени учетной записи, обладающей правом CONTROL SERVER для экземпляра SQL Server. По умолчанию этим правом обладают только системные администраторы;
- подключение в этом режиме может быть установлено только с использованием сетевой библиотеки **TCP/IP**;
- подключение в режиме DAC является "неубиваемым": его нельзя закрыть командой `KILL`;
- при подключении в режиме DAC вы получаете возможность напрямую производить запросы и вносить изменения в системные таблицы сервера в базе данных `master` (и то и другое для обычных подключений в SQL Server 2005 запрещено);
- только в режиме DAC (и только тогда, когда сервер запущен в однопользовательском режиме) вы можете получить доступ к секретной базе данных `resource` (обратиться к ней можно по команде `USE mssqlsystemresource`). Эта база данных содержит копии всех системных объектов (например, системных таблиц в базах данных), которые поставляются с SQL Server 2005. Изменения в нее вносятся только при установке пакетов обновления и патчей.

Для подключения в режиме DAC используется свой собственный планировщик SQL Server, который обеспечивает такому подключению приоритет перед всеми остальными. Поэтому может возникнуть соблазн использовать этот режим для выполнения срочных запросов на загруженном сервере. Однако нужно учитывать, что подключению в режиме DAC выдается практически фиксированный набор ресурсов (и при этом не очень большой). На выполнение базовых административных операций этих ресурсов вполне хватает, но на выполнение сложных запросов их может быть и не достаточно. Поэтому рекомендуется использовать DAC только по своему прямому назначению: для исправления аварийных ситуаций на сервере.

3.5. SQL Server Surface Area Configuration

Это еще одно средство администрирования SQL Server 2005, которого не было в предыдущих версиях. Дословно SQL Server Surface Area Configuration переводится как "Настройка поверхности SQL Server" или "Настройка контактной зоны SQL Server". Под "настройкой поверхности" подразумевается возможность убрать с SQL Server 2005 все лишние компоненты, которые в конкретной задаче могут быть не нужны. Смысл этого действия — максимально снизить число возможных способов проникновения в SQL Server для хакеров за счет сокращения "поверхности, доступной для атаки".

При запуске программы SQL Server Surface Area Configuration через системное меню **Пуск | Программы | Microsoft SQL Server 2005 | Configuration Tools** откроется окно, в котором вы можете выбрать один из двух вариантов работы:

- Surface Area Configuration for Services and Connections** — включить/отключить службы и сетевые протоколы;
- Surface Area Configuration for Features** — настроить используемые/неиспользуемые возможности этих служб.

Если вы пойдете по первому пути, то возможностей для выбора у вас будет совсем немного: вы сможете только настроить параметры работы служб SQL Server (например, отключить службу или изменить ее режим запуска при загрузке компьютера) и выбрать используемые сетевые протоколы. Второй вариант намного интереснее. С его помощью вы можете указать, будут ли на вашем сервере включены разные возможности. Для самого сервера это:

- возможность использования функций `OPENROWSET` и `OPENDATASOURCE` (по умолчанию обе они отключены). Эти функции позволяют устанавливать соединения с внешним источником данных в момент выполнения запроса;
- CLR Integration** (т. е. возможность обращения к сборкам .NET из кода Transact-SQL) — это самая рекламированная возможность SQL Server 2005, но она также по умолчанию отключена;
- Database Mail** — программный модуль, который позволяет отправлять почтовые сообщения из кода Transact-SQL по протоколу SMTP. По умолчанию отключен;
- Dedicated Administrator Connection (DAC)** — здесь имеется в виду возможность подключения в этом режиме к серверу с других компьютеров. Как уже говорилось в *разд. 3.4.2* про `SQLCmd`, эта возможность изначально отключена;
- Native Web Services** — настройка параметров HTTP Endpoints (точек прямого подключения к SQL Server 2005 по протоколу HTTP);
- OLE Automation** — это любимые многими разработчиками хранимые процедуры автоматизации (`sp_OACreate`, `SP_OAMethod` и т. п.), которые также по умолчанию в SQL Server 2005 не работают;
- Service Broker Endpoints** — точки подключения Service Broker. При помощи этого пункта вы можете удалить ненужные точки подключения;
- SQL Mail** — оставленная для обратной совместимости с предыдущими версиями SQL Server система работы с электронной почтой, работающая по MAPI. Она отключена по умолчанию, видимо, заодно с Database Mail;

- **xp_cmdshell** — это расширенная хранимая процедура, позволяющая выполнять из кода Transact-SQL команды операционной системы. Также по умолчанию отключена;
- **Web Assistant** — хранимые процедуры, которые позволяют генерировать файлы HTML как результаты запросов к базам данных. Они тоже по умолчанию не работают в SQL Server 2005.

Многие возможности отключены по умолчанию и в Analysis Services. Их можно включить (или снова отключить) при помощи Surface Area Configuration. Это следующие возможности Analysis Services:

- функция OPENROWSET из языка запросов DMX (Data Mining Extensions — расширения SQL для запросов к моделям добычи данных). Как и однотипная функция из Transact-SQL, эта функция OPENROWSET позволит (если ее включить) устанавливать соединение с внешним источником данных в момент выполнения запроса;
- возможность устанавливать анонимные соединения с Analysis Services (базами данных OLAP);
- возможность использовать внешние функции, созданные в формате модулей DLL с использованием COM;
- возможность добавлять в свои кубы измерения и группы показателей с других серверов.

В Reporting Services, наоборот, возможности, управляемые Surface Area Configuration, по умолчанию включены, и вы можете их отключить. Таких возможностей всего две: создание и доставка отчетов по расписанию и возможность обращения к отчетам по протоколу HTTP (если вы отключите эту возможность, то при помощи средств разработки и администрирования к отчетам не смогут обращаться не только пользователи, но и вы сами).

Подводя итоги, можно сказать, что SQL Server Surface Area Configuration не относится к числу незаменимых приложений. Настраивать режим работы службы и включать/отключать сетевые библиотеки можно при помощи SQL Server Configuration Manager, а включать/отключать возможности компонентов SQL Server можно при помощи стандартных средств, например, SQL Server Management Studio или из кода Transact-SQL. Однако в Surface Area Configuration управление этими возможностями сведено воедино, что очень удобно.

3.6. SQL Server Profiler

SQL Server Profiler, который специалисты обычно называют профилировщиком, — одно из самых полезных программных средств, входящих в состав

SQL Server. Запустить это приложение можно из системного меню **Пуск | Программы | Microsoft SQL Server 2005 | Performance Tools** или из меню **Tools** двух других приложений — SQL Server Management Studio и Database Engine Tuning Advisor.

Главное назначение SQL Server Profiler — это просмотр (или запись в файл или в таблицу) всех событий SQL Server, включая выполняемые на нем команды Transact-SQL. Типичная ситуация, когда без профилировщика не обойтись, выглядит так: у вас есть приложение, написанное другими разработчиками, которое обращается к таблицам, представлениям, хранимым процедурам своей базы данных SQL Server. Как показывает опыт, разработчики редко балуют пользователей своего приложения (и администраторов), которые их обслуживают, подробной документацией, в которой описаны таблицы и другие объекты, используемые приложением. В то же время часто возникают ситуации, когда необходимо получить информацию о том, какие команды выполняет на сервере приложение, например:

- вам нужно самим создать отчет, который в приложении не предусмотрен, и вы не знаете, в каких таблицах находятся нужные вам данные;
- при выполнении приложением определенных операций на сервере возникает ошибка, и вы хотите понять, какая команда Transact-SQL к ней приводит;
- вы хотите понять, насколько оптимально с точки зрения производительности выглядят запросы, выполняемые приложением.

Во всех этих очень распространенных ситуациях вам поможет профилировщик.

Но у него есть и другие применения. Например, профилировщик можно использовать для записи активности пользователей в файл или в таблицу SQL Server, а затем использовать полученные данные для аудита. Такой же файл или таблицу можно использовать в качестве исходной информации для Database Engine Tuning Advisor (*см. разд. 11.5.5*).

Профилировщик поставлялся и с предыдущими версиями SQL Server, однако в SQL Server 2005 его возможности значительно расширены. Были добавлены новые возможности:

- профилировка Analysis Services — теперь вы можете просматривать команды и события не только для обычных баз данных, но и для баз данных OLAP;
- профилировка событий Integration Services — теперь вы можете при помощи профилировщика отслеживать ход выполнения новых пакетов DTS;
- возможность при записи информации выполнения команды записывать показания счетчиков из Performance Monitor;

- в профилировщик добавлено множество новых событий и источников информации, которые могут выбираться для записи в файл трассировки. Определение того, что нужно записывать в файл трассировки, теперь можно сохранить в формате XML;
- возможность сохранять в формате XML и результаты трассировки (возможность записи в форматах ANSI, OEM, UNICODE также сохранена);
- возможность сохранять в формате XML даже планы выполнения команд Transact-SQL, перехваченных профилировщиком. Затем сохраненные в таком формате планы можно открыть в SQL Server Management Studio для дальнейшего анализа;
- возможность группировать события прямо в окне профилировщика. С ее помощью, например, вы можете очень просто посчитать, сколько раз в течение дня на сервере выполнялась та или иная команда Transact-SQL.

Подробно про работу с профилировщиком будет рассказано в разд. 11.2.3.

3.7. Database Engine Tuning Advisor

Это программное средство в SQL Server 2005 заменило программу Index Tuning Wizard из предыдущих версий SQL Server. Оно включено в SQL Server 2005 в двух вариантах:

- графическом (исполняемый файл DTAShell.exe). Его можно запустить из меню **Пуск | Программы | Microsoft SQL Server 2005 | Performance Tools**, или из меню **Tools** в SQL Server Management Studio, или SQL Server Profiler. Кроме того, это средство автоматически запускается в режиме анализа команды или скрипта Transact-SQL, если эту команду или скрипт целиком выделить в окне редактора кода SQL Server Management Studio и в контекстном меню выбрать **Analyze Query in Database Tuning Advisor** (Анализировать запрос в Database Tuning Advisor);
- консольном. Запускается из командной строки по команде **DTA**.

Это программное средство предназначено для того, чтобы облегчить работу по оптимизации индексов и других структур в базе данных. Оно принимает в качестве исходной информации файл или таблицу трассировки, созданную при помощи профилировщика. Обычно в таком файле или таблице собирается информация о работе пользователей на сервере за продолжительный промежуток времени (например, за рабочий день). В качестве варианта можно передать Tuning Advisor команду или набор команд из окна редактора кода SQL Server Management Studio. Затем Tuning Advisor в соответствии с указанными вами параметрами рассчитывает возможные варианты внесения изменений в индексы и другие объекты базы данных (например, оценивает ва-

рианты с секционированием) и по результатам анализа генерирует отчет (в формате XML) и рекомендации. Самая важная информация этого отчета — на сколько увеличится производительность каких отчетов при реализации предложенных рекомендаций. Сами рекомендации (например, команды на создание, изменение, удаление индексов) можно сохранить в виде скрипта SQL для дальнейшего анализа или применить к базе данных.

Если база данных у вас большая, и при выполнении некоторых запросов могут возникнуть проблемы с производительностью, то без оптимизации системы индексов вам не обойтись. Tuning Advisor — не чудодейственное средство, автоматически решающее все проблемы, но оно может взять на себя значительную часть работы, которую в противном случае пришлось бы выполнять вручную. Это программное средство позволяет определить основные направления, на которых нужно сосредоточить усилия при оптимизации базы данных. В некоторых ситуациях его применение может сразу же дать достаточно большой выигрыш в производительности.

По сравнению с Index Tuning Wizard в Database Engine Tuning Advisor были внесены значительные изменения:

- можно производить анализ для нескольких баз данных одновременно. Для этого Tuning Advisor учитывает все команды USE в исходных данных;
- в анализ включаются команды, работающие с временными таблицами, пользовательские функции и команды, которые выполняются триггерами;
- можно задать максимум времени, которое Tuning Advisor затратит на анализ и выработку рекомендаций. Такая возможность может оказаться полезной, если в вашем распоряжении есть только небольшое временное окно для проведения анализа (например, в остальное время дополнительно загружать сервер нельзя). Однако рекомендуется не ограничивать Tuning Advisor во времени: чем больше времени будет отведено на анализ, тем качественнее получатся рекомендации. В качестве варианта можно рассмотреть возможность работы Tuning Advisor с копией рабочей базы данных на запасном сервере;
- появилась возможность настраивать работу Tuning Advisor в режиме "никакие новые индексы не создаем, определяем только возможность удаления существующих индексов";
- новая возможность Tuning Advisor — оценка сценариев, предлагаемых пользователем, например: "А что будет, если этот индекс добавить, а этот — удалить?";
- ведется журнал анализа. В этот журнал, например, заносится информация о всех записях из файла трассировки, которые не удалось использовать для анализа;

- появилась возможность сохранять параметры анализа в XML-файле (при помощи меню **File | Export Session Definition** (Файл | Экспортировать определение сеанса)) и использовать их повторно. Такие файлы можно передавать и консольному варианту Tuning Advisor. Отчеты о результатах анализа, как уже говорилось ранее, также формируются в формате XML (их можно просматривать как из специальных средств для работы с XML, так и при помощи графического интерфейса Tuning Advisor).

Подробно про работу с Database Tuning Advisor будет рассказано в разд. 11.5.5.

3.8. Другие графические утилиты SQL Server 2005

В этом разделе представлена краткая информация о других графических утилитах, которые поставляются вместе с SQL Server 2005. Для большинства из них нет ярлыков, доступных из меню **Пуск**. Для запуска этих утилит потребуется найти их исполняемый файл на диске (некоторые из них можно также вызвать из других программных средств, таких как SQL Server Management Studio). Далее они перечислены по алфавиту.

- **Analysis Services Deployment Wizard** (`Microsoft.AnalysisServices.Deployment.exe`) — мастер развертывания проектов Analysis Services на другом сервере. Обычно используется для переноса созданной вами на тестовом сервере базы данных OLAP на рабочий сервер. Этую утилиту можно запустить из меню **Пуск | Программы | Microsoft SQL Server 2005 | Analysis Services**.
- **Analysis Services Instance Rename** (`ASInstanceRename.exe`) — утилита, которая позволяет переименовать экземпляр Analysis Services. Используется обычно для перевода тестового сервера Analysis Services в рабочий режим в тех ситуациях, когда имя Analysis Services жестко прописано в клиентских приложениях. Можно запустить из того же меню, что и Deployment Wizard.
- **Analysis Services Migration Wizard** (`MigrationWizard.exe`) — средство, предназначенное для переноса баз данных OLAP, созданных в предыдущей версии Analysis Services (поставлявшейся с SQL Server 2000) на SQL Server 2005. Запускается из того же меню. У этой программы есть консольный вариант, который называется `MigrationWizardConsole.exe`.
- **Replication Conflict Viewer** (`ConflictViewer.exe`) — средство, которое позволяет просматривать и разрешать конфликты, возникающие в процессе репликации слиянием (*merge replication*). Обычно запускается из SQL Server Management Studio.

- **Configure Web Synchronization Wizard** (ConnWiz30.exe) — это программное средство используется для настройки синхронизации по протоколу HTTP между SQL Server Mobile Edition (редакция для наладонных компьютеров и смартфонов) и обычным SQL Server 2005. Фактически позволяет настроить репликацию между этими двумя редакциями SQL Server 2005. Обычно запускается из SQL Server Management Studio.
- **Copy Database Wizard** (CopyDatabaseWizard.exe) — этот мастер позволяет перенести базы данных (со всеми объектами, данными, разрешениями и т. п.) с SQL Server 2000 или SQL Server 2005 на другой экземпляр SQL Server 2005. Это один из самых простых способов произвести обновление базы данных SQL Server 2000 до SQL Server 2005. Обычно запускается из SQL Server Management Studio (в **Object Explorer** через контекстное меню базы данных **Tasks | Copy Database**).
- **Execute Package Utility** (DTEExecUI.exe) — наследница утилиты DTSERunUI в предыдущих версиях SQL Server. Позволяет запустить на выполнение пакеты SQL Server Integration Services (SSIS (в предыдущих версиях это DTS)), запланировать их выполнение по расписанию или создать командную строку для запуска пакета из консольного аналога DTSEExec.exe. Про работу с этой утилитой будет рассказываться в *разд. 10.28.*
- **Replication Monitor** (sqlmonitor.exe) — важное средство для мониторинга и диагностики репликации. Можно запустить как из командной строки, так и из SQL Server Management Studio. Про работу с ним рассказывается в *разд. 12.7.2.*
- **Reporting Services Configuration** (RSConfigTool.exe) — эта программа предназначена для управления настройками Reporting Services. Она доступна через меню **Пуск | Программы | Microsoft SQL Server 2005 | Configuration Tools**. Консольный вариант этой программы называется rsconfig.exe.
- **SQL Server Import and Export Wizard** (DTSWizard.exe) — утилита, заменившая в новой версии SQL Server старый мастер импорта и экспорта данных DTS Import/Export Wizard. Предназначена она для той же цели — быстрое создание простых пакетов для загрузки/выгрузки данных в SSIS (бывший DTS). Работа с этим мастером будет рассматриваться в *разд. 10.3.*
- **SQL Server Integration Services Migration Wizard** (DTSMigrationWizard.exe) — утилита, предназначенная для переноса и обновления пакетов DTS, созданных в SQL Server 7.0 и 2000, на SQL Server 2005 в новую среду SSIS.

3.9. Другие консольные утилиты SQL Server 2005

В этом разделе представлена информация о консольных утилитах, которые входят в состав SQL Server 2005. Поскольку все они запускаются из командной строки, то вначале приводится название исполняемого файла, а в скобках полное название утилиты (если оно предусмотрено).

- **bcp.exe** (Bulk Copy Program) — эта утилита-ветеран играет важную роль и в SQL Server 2005. Это самый быстрый способ загрузить данные на SQL Server 2005 и выгрузить их. Расплатой за скорость является невысокая функциональность: вы можете загружать данные только из текстовых файлов или выгружать их в текстовые файлы. Если нужно произвести обмен данными с другими источниками данных или параллельно с загрузкой/выгрузкой выполнить преобразования или проверки, то придется использовать пакеты SSIS (DTS). Новой в SQL Server 2005 является возможность применения для этой программы файла форматирования XML (поддержка старых файлов форматирования также сохранена).
- **cidump.exe** — недокументированная, но очень интересная утилита, которая позволяет производить проверку целостности индексов и отображать их статистику. Может использоваться также для выгрузки (*dump*) индексов: целиком или отдельных диапазонов.
- **dta.exe** — как уже говорилось, это консольный вариант Database Engine Tuning Advisor (см. разд. 3.7 и 11.5.5).
- **dtattach.exe** — эта недокументированная утилита позволяет подключить отладчик к пакетам SSIS, в которых используется Script Task.
- **DTEexec.exe** — эта утилита позволяет запускать на выполнение пакеты SSIS из командной строки. Команду на ее запуск с необходимыми параметрами удобнее всего создавать при помощи утилиты DTEexecUI.
- **DTSRun.exe** — утилита для запуска пакетов DTS, которая поставлялась с SQL Server 2000. Она входит и в SQL Server 2005, но теперь ее предлагают использовать только для единственной цели — для сбора информации журнала запуска отчетов с Reporting Services.
- **DTUtil.exe** — утилита для управления пакетами SSIS (копирование, удаление, переименование и т. п.) из командной строки.
- **Irtest.exe** (Language Resource test) — утилита, предназначенная для тестирования изменений, которые вносятся в языковые ресурсы полнотекстовых индексов (фильтров шумовых слов и т. п.).
- **MigrationWizardConsole.exe** — консольный вариант Analysis Services Migration Wizard — мастера переноса баз данных OLAP с Analysis Services в SQL Server 2000 на SQL Server 2005.

- **nscontrol.exe** (Notification Services Control Utility) — основное средство администрирования компонента SQL Server 2005, который называется Notification Services. Это средство позволяет создавать экземпляры Notification Services, настраивать их параметры, создавать правила и т. п.
- **nsservice.exe** — это сама служба Notification Services. Если запустить ее с параметром -a, то она будет работать как консольное приложение.
- **osql.exe** — утилита для выполнения команд Transact-SQL и скриптов из командной строки. Оставлена в SQL Server 2005 для обеспечения обратной совместимости. Microsoft рекомендует всегда использовать вместо этой утилиты команду `SQLCmd`.
- **PSSDiag.exe** — эта утилита призвана заменить утилиту `SQLDiag` из предыдущих версий SQL Server. Главное ее назначение — диагностика сервера. Она производит тестирование различных компонентов сервера и выдает информацию о результатах проверки.
- **rsactivate.exe** (Report Server Activation Tool) — утилита, которая позволяет инициализировать Report Server. Используется после выполнения операций с сертификатами Report Server или изменения важных параметров его работы.
- **rsconfig.exe** (Report Server Configuration Management) — эта утилита позволяет изменять настройки Reporting Services из командной строки.
- **RSKeyMgmt.exe** (Report Server Key Manager) — утилита позволяет выполнять различные операции с сертификатами Reporting Services.
- **SqlMailWizard.exe** — странный гибрид консольного и графического приложений. Эта программа предназначена для настройки подсистемы `SQLiMail` в базах данных. Принимает параметры только из командной строки, однако сообщения выдает в графическом режиме с кнопками, которые, например, позволяют отправить сообщение по электронной почте средствами `SQLiMail`.
- **sqlmailnt.exe** — к радости многих администраторов SQL Server 2000, эта утилита оставлена и в SQL Server 2005. Однако оставлена она ненадолго: уже в следующей версии ее обещают убрать. Эта утилита выполняет те же функции, что и в SQL Server 2000 — позволяет выполнять административные операции (резервное копирование, проверку целостности, обновление статистики, перестроение отчетов) из командной строки и создавать отчеты о их выполнении в текстовом формате или в формате HTML, а также отсылать их по электронной почте. Вместо этой утилиты рекомендуется использовать планы обслуживания баз данных.
- **tablediff.exe** (Replication Table Diff Tool) — очень интересная утилита, которая позволяет сравнивать информацию в двух таблицах и выводить про-

токол различий в файл или в таблицу SQL Server. Как понятно из названия, в основном эта утилита предназначена для диагностики репликации, однако ее вполне можно использовать и для других целей.

Задание для самостоятельной работы 3.1. Работа со скриптами в SQL Server Management Studio и SQLCmd

Ситуация:

Вам необходимо создать в базе данных AdventureWorksDW на вашем локальном компьютере таблицу dbo.BuyerCopy со структурой, аналогичной структуре таблицы dbo.ProspectiveBuyer в этой же базе данных.

Задание:

1. Создайте при помощи средств автоматической генерации скриптов скрипт на создание таблицы dbo.BuyerCopy в соответствии с поставленными условиями, сохраните этот скрипт как C:\Buyer_Creation.sql.
2. Создайте пакетный файл C:\Buyer.bat. В этом пакетном файле должны находиться команды на создание таблицы dbo.BuyerCopy средствами утилиты SQLCmd с использованием созданного вами файла C:\Buyer_Creation.sql. Все ошибки, возникающие при выполнении команд SQLCmd должны записываться в файл C:\Buyer_Creation_Log.txt.
3. Запустите этот пакетный файл на выполнение и убедитесь, что таблица dbo.BuyerCopy действительно создана.

Решение:

К пункту 1 задания — генерация скрипта на создание таблицы:

1. Запустите SQL Server Management Studio и подключитесь к своему локальному серверу при помощи аутентификации Windows.
2. В окне **Object Explorer** раскройте контейнер **имя_вашего_сервера | Databases | AdventureWorksDW | Tables**.
3. Щелкните правой кнопкой мыши по объекту таблицы dbo.ProspectiveBuyer и в контекстном меню выберите **Script Table as | Create to | New Query Editor Window** (Отскриптовать таблицу как | Создать в | Новое окно редактора запросов). Откроется новое окно редактора кода, в которое будет помещен сгенерированный скрипт на создание таблицы.

4. В этом скрипте замените строку:

```
CREATE TABLE [dbo].[ProspectiveBuyer];
```

на строку:

```
CREATE TABLE [dbo].[BuyerCopy];
```

Остальные строки оставьте без изменений.

5. Нажмите комбинацию клавиш <Ctrl>+<S>. Сохраните скрипт в файле C:\Buyer_Creation.sql.

К пункту 2 — создание пакетного файла:

Код для пакетного файла Buyer.bat может быть таким:

```
@echo off  
sqlcmd -S LONDON2\SQL2005 -i c:\Buyer_Creation.sql -o  
c:\Buyer_Creation_Log.txt
```

Задание для самостоятельной работы 3.2. Работа с серверными сетевыми библиотеками и псевдонимами

Задание:

1. Включите на сервере серверную сетевую библиотеку **TCP/IP**.
2. Определите, использует ли экземпляр *имя_вашего_компьютера\SQLServer2005* статический порт или работает с динамическими портами.
3. Настройте на вашем компьютере псевдоним MyServer. При обращении к этому псевдониму должно производиться подключение к серверу *имя_вашего_компьютера\SQL2005*.

Решение:

К пунктам 1 и 2 задания — включение сетевой библиотеки и просмотр информации об используемых портах:

1. Откройте SQL Server Configuration Manager (из системного меню **Пуск | Программы | Microsoft SQL Server 2005 | Configuration Tools**).
2. Раскройте контейнер **SQL Server 2005 Network Configuration | Protocols for SQL2005**, щелкните правой кнопкой мыши по объекту **TCP/IP** в правой части экрана и в контекстном меню выберите команду **Enable**.

3. В том же контекстном меню выберите команду **Properties**, а затем перейдите на вкладку **IP Addresses** (IP-адреса). Для всех IP-адресов параметр **TCP Port** должен быть пустым, а для параметра **TCP Dynamic Ports** должно быть установлено значение 0. Установите указатель ввода для значения параметра **TCP Dynamic Ports** и прочитайте описание для этого параметра в нижней части экрана.
4. Закройте окно свойств сетевой библиотеки **TCP/IP**, перейдите в контейнер **SQL Server 2005 Services** и перезапустите службу SQL Server (SQL2005).

К пункту 3 задания — настройка псевдонима для обращения к серверу:

1. В SQL Server Configuration Manager раскройте контейнер **SQL Native Client Configuration | Aliases**, щелкните по нему правой кнопкой мыши и в контекстном меню выберите **New Alias** (Новый псевдоним).
2. Для параметра **Alias Name** (Имя псевдонима) создаваемого псевдонима введите значение `MyServer`. Значение для параметра **Port No** (Номер порта) оставьте пустым. Для параметра **Protocol** оставьте предлагаемое по умолчанию значение `TCP/IP`, а для параметра **Server** введите значение `имя_вашего_компьютера\SQL2005` (например, `LONDON\SQL2005`).
3. Нажмите кнопку **OK** и закройте SQL Server Configuration Manager с сохранением внесенных изменений.
4. Для проверки работоспособности созданного псевдонима подключитесь к серверу `MyServer` из SQL Server Management Studio.



ГЛАВА 4

Создание баз данных и настройка параметров

Вы научились производить установку SQL Server 2005 и познакомились с программными средствами, которые можно использовать для работы с ним. Обычно следующая задача администратора SQL Server — создание базы данных и настройка параметров ее работы. Эти темы и будут рассмотрены в этой главе.

4.1. Служебные и учебные базы данных SQL Server 2005

Прежде чем мы приступим к обсуждению вопросов, связанных с созданием своих собственных рабочих баз данных, скажем несколько слов о служебных базах данных SQL Server, которые создаются автоматически в процессе его установки, и о учебных базах данных, которые можно установить в качестве дополнительного необязательного компонента.

Если мы откроем контейнер **Databases | System Databases** (Базы данных | Системные базы данных) в **Object Explorer** в SQL Server Management Studio, то увидим тот же набор служебных баз данных, что и в SQL Server 2000:

- master** — главная служебная база данных всего сервера. В ней хранится общая служебная информация сервера: настройки его работы, список баз данных на сервере с информацией о настройках каждой базы данных и ее файлах, информация об учетных записях пользователей для подключения к SQL Server (логинах), серверных ролях и т. п. Главным отличием базы данных **master** SQL Server 2005 от предыдущих версий является то, что практически все ее системные таблицы (а других в этой базе данных и нет) теперь недоступны не только для прямого внесения изменений, но и для просмотра. Точно так же защищены системные таблицы всех других баз

данных. Единственная возможность выполнять запросы и вносить изменения в таблицы базы данных `master` напрямую — это перевести сервер в однопользовательский режим и подключиться в режиме Dedicated Administrator Connection (см. разд. 3.4.2). Сами системные таблицы в базе данных `master` также изменились очень сильно;

- `msdb` — эта база данных в основном используется для хранения информации службы SQL Server Agent (пакетных заданий, служб, предупреждений и т. п.), но в нее записывается и другая служебная информация (например, история резервного копирования);
- `model` — эта база данных является шаблоном для создания новых баз данных в SQL Server. Если внести в нее изменения, например, создать набор таблиц, то эти таблицы будут присутствовать во всех создаваемых базах данных;
- `tempdb` — эта база данных предназначена для временных таблиц и хранимых процедур, создаваемых пользователями и самим SQL Server, а также для хранения копий изменяемых данных в режиме изоляции транзакций моментальных снимков (*snapshot isolation*) и промежуточных данных при перестроении индексов. Эта база данных создается заново при каждом запуске SQL Server;
- `distribution` — этой служебной базы данных изначально в SQL Server 2005 нет. Она появляется при настройке репликации (для нее можно выбрать и другое название).

Других служебных баз данных в **Object Explorer** не видно. Но если вы откроете на диске каталог Data для вашего экземпляра SQL Server 2005, то вы увидите файлы и для других служебных баз данных, которые спрятаны от глаз пользователей и администраторов:

- база данных `resource` (ей соответствует файл `mssqlsystemresource.mdf`). Подключиться к ней можно только в режиме Dedicated Administrator Connection (и только когда сервер запущен в однопользовательском режиме). Эта база данных содержит копии всех системных объектов (например, системных таблиц в базах данных), которые поставляются с SQL Server 2005. Изменения в нее вносятся только при установке пакетов обновления и патчей;
- база данных `distmodel` (ей соответствует файл `distmdl.mdf`). Это шаблон, используемый для создания базы данных `distribution` при настройке репликации.

Если вы установили программный компонент Reporting Services, то на SQL Server 2005 будут созданы дополнительные служебные базы данных (они отображаются в **Object Explorer** не в **System Databases**, а как пользователь-

ские базы данных). Их названия по умолчанию начинаются с префикса ReportServer\$.

Кроме служебных баз данных, после установки на SQL Server 2005 могут появиться еще две базы данных: AdventureWorks и AdventureWorksDW (буквы DW означают Data Warehouse, т. е. эта база помещена как пример хранилища данных). Это новые учебные базы данных, которые заменили привычные Pubs и Northwind. По сравнению с Pubs и Northwind это совсем другие базы данных и по структуре, и по объему (основная учебная база данных AdventureWorks занимает больше 180 Мбайт).

Конечно, на рабочем сервере учебных баз данных быть не должно, а вот на сервере, который используется для целей разработки, их вполне можно оставить, поскольку многие примеры в документации используют именно эти базы данных.

4.2. Создание пользовательских баз данных

Мы разобрались с тем, какие базы данных изначально могут быть на SQL Server 2005. Но, конечно, для хранения информации ваших приложений вам потребуются свои собственные пользовательские базы данных. Их придется создавать самим.

Создание баз данных на SQL Server 2005 может производиться множеством разных способов, которые будут рассмотрены далее в этом разделе.

4.2.1. Создание базы данных из SQL Server Management Studio

Самый простой способ создать базу данных — воспользоваться графическим интерфейсом SQL Server Management Studio. Сама процедура создания занимает секунды. Нужно щелкнуть правой кнопкой мыши по контейнеру **Database** в **Object Explorer** и в контекстном меню выбрать **New Database** (Новая база). Откроется диалоговое окно **New Database**, в котором в самом простом случае вам достаточно будет ввести только имя создаваемой базы данных (рис. 4.1). Для всех остальных параметров будут подставлены значения по умолчанию.

Однако чаще всего при создании базы данных вам потребуется указать свою собственную конфигурацию файлов и файловых групп, сразу же настроить параметры базы данных, а иногда и определить расширенные свойства. Обо всех этих моментах будет рассказано в следующих разделах этой главы.

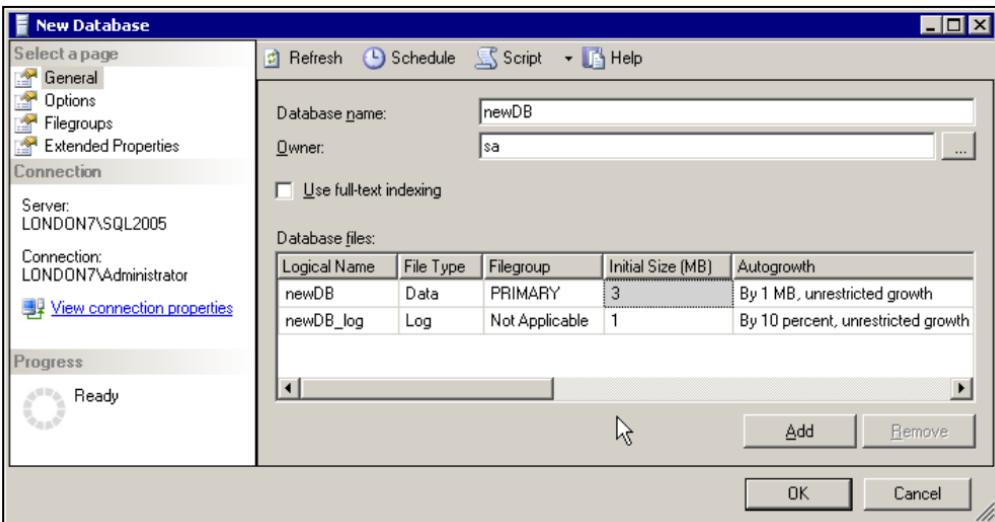


Рис. 4.1. Окно создания новой базы данных

4.2.2. Создание базы данных при помощи команды *CREATE DATABASE*

Очень часто рабочие базы данных создаются при помощи команды Transact-SQL `CREATE DATABASE`. Обычно эта команда помещается в скрипт, который, помимо создания самой базы данных, выполняет и другие операции, например, настройку параметров базы данных и создание в ней объектов. Полный синтаксис команды `CREATE DATABASE` здесь приводиться не будет: это заняло бы несколько страниц, и пришлось бы полностью дублировать документацию. Вместо этого мы расскажем о том, как такой скрипт можно создать в автоматизированном режиме.

Вместо того чтобы писать команду `CREATE DATABASE` вручную (это может быть достаточно трудоемким занятием, кроме того, всегда есть риск допустить ошибки), ее можно сгенерировать автоматически. Это можно сделать двумя способами: сгенерировать скрипт для существующей базы данных и воспользоваться шаблоном редактора кода.

Первый вариант (с генерацией скрипта) особенно удобен тогда, когда у вас уже есть оттестированная база данных на сервере, который использовался для разработки, и вы хотите разместить ее копию (возможно, включая объекты) на другом сервере или на множестве серверов (например, в филиалах). Создать скрипт можно двумя способами.

Более простой и менее функциональный вариант выглядит так:

1. Откройте SQL Server Management Studio и подключитесь к серверу, на котором расположена интересующая вас база данных.
2. Раскройте контейнер **Databases**.
3. Щелкните правой кнопкой мыши по объекту нужной базы данных и в контекстном меню выберите **Script Database As | Create to | New Query Editor Window** (Отскриптовать базу данных как | Создать | Новое окно редактора кода) (как вариант, созданный скрипт можно также сохранить в файле (пункт контекстного меню **File**) или поместить в буфер обмена (пункт **Clipboard**)).

Однако при этом вы создадите только скрипт на создание самой базы данных и настройку ее параметров. Если вам нужно также поместить в скрипт команды на создание объектов баз данных (таблиц, представлений и т. п.), придется использовать другой способ:

1. Точно так же откройте контекстное меню нужной базы данных.
2. В контекстном меню выберите **Tasks | Generate Scripts** (Задачи | Сгенерировать скрипты).

После этого откроется мастер генерации скриптов **Generate SQL Server Scripts Wizard**. На страницах этого мастера вы можете выбрать множество параметров генерации скрипта: для какой базы данных он создается, нужно ли помещать в скрипт команды на создание всех объектов в этой базе данных или вы хотите поместить в скрипт команды только на создание объектов определенного типа (таблиц, представлений, индексов, ограничений целостности, триггеров и т. п.), скриптовать ли пользователей базы данных и разрешения, которые им предоставлены на объекты, помещать в команды на создание проверку существования объектов с такими же именами, генерировать ли скрипты на удаление объектов и т. п.

Единственное, что, пожалуй, не хватает в этом мастере — возможности одновременно генерировать команды `INSERT` для загрузки данных. Но для создания базы данных с загруженными данными придется использовать другие способы, которые будут рассмотрены в следующем разделе.

Второй способ упростить создание скрипта `CREATE DATABASE` — воспользоваться шаблоном редактора кода. Выглядит эта операция так:

1. Откройте SQL Server Management Studio и в меню **File** выберите **New | File**.
2. В окне **New File** (Новый файл) в списке **Categories** (Категории) выберите категорию **SQL Server Query | Database** (Запрос SQL Server | База данных) и в списке справа выберите подходящий шаблон для создания базы

данных (самый простой из предлагаемых вариантов называется `create database`).

- Шаблон будет загружен в окно редактора кода SQL Server Management Studio. Его можно доработать вручную, а можно воспользоваться специальным окном для заполнения значений параметров. Это окно можно открыть при помощи меню **Query | Specify Values for Template Parameters**.

4.2.3. Альтернативные способы создания базы данных

Ранее были перечислены классические способы создания баз данных: при помощи графического интерфейса и при помощи команды `Create Database`. Это самые простые способы, но у них есть существенный недостаток — они не позволяют создавать базы данных с изначально загруженными данными. В то же время в реальной работе обычно требуется не только создать базу данных, но и сразу загрузить в нее некоторые данные, например, информацию справочников. Для этой цели в вашем распоряжении — альтернативные способы создания баз данных:

- восстановление с резервной копии;
- подключение файлов существующей базы данных;
- применение мастера копирования баз данных;
- использование средств SSIS (пакета DTSX).

Для всех этих способов необходимо, чтобы у вас уже существовала готовая база данных-образец с загруженными данными.

Чаще всего используется первый способ — с резервным копированием и с восстановлением баз данных. Про резервное копирование и восстановление будет рассказываться в гл. 6.

Второй способ, безусловно, самый простой и надежный, однако многие администраторы и разработчики про него забывают. Поэтому расскажем о нем подробнее.

Подключение файлов существующей базы данных можно использовать в разных ситуациях: для быстрого восстановления базы данных SQL Server в случае сбоя сервера, когда сами файлы базы данных сохранились, при изменении дисковой конфигурации сервера, или, как в нашем случае, при необходимости скопировать базу данных на другой сервер. Предположим, что вам нужно скопировать вашу базу данных с тестового сервера на рабочий. Последовательность действий может выглядеть так:

- Первое, что вам нужно сделать, — скопировать файлы базы данных с тестового сервера. Однако просто взять и скопировать их вы не можете: они

открыты сервером и при попытке копирования будет выдано сообщение об ошибке. Чтобы на время закрыть эти файлы, можно использовать два способа:

- более быстрый способ: отсоединить базу данных. Для этого в контекстном меню этой базы данных в **Object Explorer** выберите команду **Tasks | Detach** (Задачи | Отсоединить);
- более надежный способ: выполнить команду `ALTER DATABASE`. Эта команда предназначена для изменения базы данных, в том числе и для изменения ее состояния. Например, чтобы перевести базу данных `DB1` в автономный режим (*offline*), команда может выглядеть так:

```
ALTER DATABASE DB1 SET OFFLINE;
```

Другая возможность перевести базу данных в автономный режим — воспользоваться командой меню **Tasks | Take offline** (Задачи | Перевести в автономный режим) в SQL Server Management Studio.

База данных, которая находится в автономном режиме, будет помечена в окне **Object Explorer** специальной красной меткой.

2. Скопируйте файлы базы данных в нужное вам место обычными средствами Windows. Лучше скопировать не только файлы базы данных, но и журнал транзакций. В принципе, если журнал транзакций утрачен, то его можно сгенерировать заново, но это усложнит процесс присоединения (вам придется использовать команду `CREATE DATABASE ... FOR ATTACH_REBUILD_LOG`).
3. Верните исходную базу данных в рабочее состояние. Если она была отсоединенна при помощи **Object Explorer**, подключите ее снова. Для этого нужно воспользоваться командой **Attach** (Присоединить) контекстного меню контейнера **Databases**. Если же база данных была переведена в автономный режим (*offline*), то верните ее обратно в оперативный режим (*online*):

```
ALTER DATABASE DB1 SET ONLINE;
```

4. Присоедините базу данных к новому серверу. Для этого в контекстном меню контейнера **Databases** в **Object Explorer** выберите команду **Attach** и в открывшемся окне **Attach Database** (Присоединение базы данных) при помощи кнопки **Add** выберите файл подсоединяемой базы данных (если журнал транзакций лежит в том же каталоге, он будет подключен автоматически), а затем нажмите **OK**. Если вам нужно подсоединить базу данных под другим именем, то вы можете ввести новое имя в столбце **Attach As** (Присоединить как). При помощи этого способа можно подключить базу данных к тому же серверу, на котором она находилась раньше, но уже в виде копии существующей базы данных и под другим именем.

На практике на выполнение этой операции требуется намного меньше времени, чем на чтение ее описания.

Таким же способом можно произвести обновление базы данных SQL Server 7.0 или 2000 до SQL Server 2005. Необходимо отключить базу данных от SQL Server старой версии, скопировать ее файлы и присоединить к SQL Server 2005. Обновление произойдет автоматически при присоединении.

В предыдущих версиях SQL Server для присоединения баз данных можно было использовать хранимые процедуры `sp_attach_db` или `sp_attach_single_file_db`. Они оставлены в SQL Server 2005 для обратной совместимости, но Microsoft рекомендует использовать вместо них команду `CREATE DATABASE` с параметром `FOR ATTACH`. Хранимой процедуре, которая использовалась для отсоединения базы данных (`sp_detach_db`), пока ничего не угрожает.

Третий способ создания базы данных — это копирование базы данных при помощи утилиты Copy Database Wizard. Этот способ тоже очень прост. С его помощью вы можете копировать базы данных не только между серверами SQL Server 2005, но и с сервера SQL Server 2000 на SQL Server 2000 или 2005: в последнем случае обновление базы данных произойдет автоматически. Недостатком этого способа является то, что оба сервера — и исходный, и сервер-получатель — должны быть доступны по сети, поэтому скопировать базу данных в филиалы таким способом получается не всегда. Copy Database Wizard можно запустить или из контекстного меню контейнера **Databases** в **Object Explorer** (команда меню **Copy Database**), или при помощи командной строки операционной системы (команда `CopyDatabaseWizard`).

Четвертый способ — применение SSIS для создания базы данных и заполнения ее данными. Он будет рассмотрен в гл. 10.

4.3. Файлы баз данных и журналов транзакций

Одно из самых важных решений, которое необходимо принять при создании базы данных, — это решение о структуре и размещении файлов данных и журналов транзакций. Неверное решение может существенно снизить производительность и надежность работы вашего приложения.

Вначале приведем немного общих сведений.

Для любой базы данных создаются файлы самой базы данных и файлы журналов транзакций. В файлах базы данных хранится вся информация о самой базе данных. В файлы журналов транзакций производится последовательная запись всех изменений, которые вносятся в базу данных. Минимальный набор файлов для любой базы данных (он же используется по умолчанию) со-

держит один файл для самой базы данных и один файл для журнала транзакций.

В каждой базе данных обязательно есть один главный (*primary*) файл. По умолчанию для него используется расширение mdf (хотя использовать именно такое расширение не обязательно — для любых файлов баз данных и журналов транзакций расширения могут быть любыми, а могут и отсутствовать). Удалять этот файл нельзя. Для базы данных можно создать и дополнительные файлы (*secondary*), для которых по умолчанию используется расширение ndf. Точно так же есть главный и дополнительные файлы у журналов транзакций, для них по умолчанию используется расширение ldf.

В принципе, база данных может вообще обходиться без файлов. Вся необходимая информация при этом будет храниться на неформатированном диске. Такой вариант называется **использованием неформатированных разделов** (*raw partitions*). Однако, в отличие от связки Unix/Oracle, связка Windows/SQL Server ничего не выигрывает от применения неразмеченных разделов, и поэтому этот вариант используется редко.

А теперь подробнее остановимся на тех решениях, которые вам придется принимать при создании базы данных.

Первое решение — это где будут размещены файлы баз данных и журналов транзакций. По умолчанию и файлы баз данных, и файлы журналов помещаются в каталог C:\Program Files\Microsoft SQL Server\MSSQL.X\MSSQL\Data (где X — номер экземпляра SQL Server 2005). Такое размещение, конечно, не оптимально.

Идеальный вариант, с точки зрения размещения файлов баз данных, — поместить их на отдельный внешний аппаратный RAID-массив. Причем на этом RAID-массиве не должно быть ничего, кроме файлов базы данных, и, кроме того, на нем должно быть как минимум 50% пустого пространства. Такой вариант обеспечивает ряд преимуществ:

- RAID-массив (в зависимости от выбранного уровня) обеспечивает высокую производительность и отказоустойчивость;
- внешний RAID-массив делает процесс восстановления больших баз данных предельно простым и быстрым. Если на сервере возникли какие-то проблемы, а с файлами базы данных все в порядке, достаточно просто подключить RAID-массив к другому серверу и присоединить базу данных (см. разд. 4.2.3);
- если вы выберете внешний RAID-массив, который входит в список совместимого оборудования (Hardware Compatibility List) для кластеров Windows Server, то при необходимости вы сможете еще больше повысить отказоустойчивость за счет создания кластера;

- если на диске будет не менее половины пространства свободно, то этим вы обеспечите себе отсутствие проблем при выполнении различных служебных операций, например, при перестроении кластеризованных индексов для больших таблиц.

Конечно, внешний RAID-массив — это идеальный вариант. Но на многих предприятиях денег на него может просто не быть. В этом случае рекомендуется, по крайней мере, использовать для файлов баз данных отдельный быстрый жесткий диск. Категорически не рекомендуется помещать на тот же диск, где находятся файлы баз данных, программные файлы операционной системы и SQL Server. Помните также, что на контроллерах доменов для разделов, на которые помещается база данных Active Directory (по умолчанию она находится в каталоге C:\Windows\NTDS), отключается кэширование на запись: падение производительности может быть просто устрашающим.

Конечно же, никогда нельзя сжимать рабочие файлы баз данных средствами NTFS или помещать их на сжатые диски.

Теперь — о размещении файлов журналов транзакций.

Требования по производительности к ним намного меньше, чем к файлам баз данных (одна из причин связана с тем, что в файлы журналов транзакций записываются только изменения, которые вносятся в базу данных — команды SELECT на них, за исключением специальных случаев, влияния не оказывают). Как размещать эти файлы, зависит от требований к базе данных и бюджета вашего предприятия. Далее представлены разные варианты, начиная от наиболее желательного и заканчивая наименее удачным:

- второй RAID-массив;
- отдельный набор дисков на том же RAID-массиве, что и файлы баз данных;
- два обычных диска, которые зазеркалированы по отношению друг к другу;
- просто обычный отдельный диск;
- размещение на том же диске, на котором размещены файлы баз данных (этот вариант наименее желателен, но именно он по умолчанию выбирается SQL Server 2005 в расчете на однодисковые системы).

Главное, что вам нужно постараться обеспечить, — это размещение журналов транзакций на другом физическом диске по отношению к файлам баз данных. Если вы выполните это условие, то в случае отказа жесткого диска сможете провести восстановление на момент сбоя (конечно, при условии, что резервное копирование все-таки производилось). Если же файлы баз данных и журналов транзакций находились на одном диске, и этот диск отказал, восстановить базу данных удастся только на момент создания последней резерв-

ной копии. В этом случае пользователям придется заново вносить данные, например, за последний день или за последнюю неделю — в зависимости от того, когда последний раз производилось резервное копирование.

Второе решение, которое вы должны принять при создании базы данных, — выбор размера файлов баз данных и журналов транзакций.

Конечно, размер файлов баз данных полностью зависит от задачи, для которой используется эта база данных. Однако у вас есть выбор — сразу создать большие файлы баз данных или настроить для них режим автоматического приращения, когда файлы при необходимости будут автоматически увеличиваться. Если такая возможность имеется, всегда нужно с самого начала создавать файлы максимального размера (или, по крайней мере, настраивать автоприращение сразу большими частями, например, в несколько гигабайт), даже несмотря на то, что в течение продолжительного времени значительная часть этих файлов использоваться не будет. Аргументация здесь проста — таким образом вы снижаете фрагментацию файлов баз данных, повышая производительность. По умолчанию для файлов баз данных настраивается худший вариант — автоприращение маленькими "порциями": 1 Мбайт для файлов базы данных и 10% от существующего размера для файлов журналов транзакций.

Однако нужно помнить, что при уменьшении размера файла (например, при удалении информации из базы данных) сделать его меньше исходного размера не получится. Поэтому создавать файлы большого размера нужно осторожно, хорошо продумав все варианты.

Настройку режима автоприращения при помощи графического интерфейса можно выполнить в окне **New Database** при создании новой базы данных или на вкладке **General** свойств базы данных в SQL Server Management Studio. Режим автоприращения устанавливается в соответствующей строке для каждого файла базы данных нажатием на кнопку в столбце **Autogrowth** (Автоматический рост) свойств данного файла (см. рис. 4.1).

Если вы уже создавали файлы баз данных большого размера (гигабайты и десятки гигабайт) в SQL Server предыдущих версий, то могли заметить, что их создание требует довольно длительного времени. Связано это было с тем, что SQL Server предыдущих версий "форматировал" пространство внутри создаваемых файлов данных, заполняя его двоичными нулями. В SQL Server 2005 появилась новая возможность, которая называется немедленной инициализацией файлов (*instant file initialization*). Она позволяет не заполнять файлы данных нулями, что резко сокращает время, требуемое для создания файлов баз данных или их увеличения. Однако эта возможность используется только при двух условиях:

- SQL Server работает под управлением операционной системы Windows Server 2003 или Windows XP;

- учетная запись, от имени которой работает SQL Server, обладает специальной привилегией операционной системы `SE_MANAGE_VOLUME_NAME` (по умолчанию такая привилегия есть у встроенной группы **Administrators**).

Эти же самые принципы относятся и к настройке автоприращения файлов журналов транзакций (заметим только, что немедленная инициализация файлов при создании файлов журналов не используется). Однако для файлов журналов транзакций возникает еще один вопрос: а какой размер журнала транзакций нужен для конкретной базы данных? Ответ на этот вопрос зависит от множества факторов:

- первый фактор — в каком режиме используется база данных. Если это база данных OLTP (т. е. данные в ней изменяются постоянно, чаще всего пользователями при помощи клиентских приложений), к которой относится абсолютное большинство используемых на предприятиях баз данных, то Microsoft рекомендует устанавливать для журналов транзакций размер от 10 до 25% от общего размера файлов баз данных. Для баз данных Data Warehouse (архивные хранилища, которые в обычном режиме используются только на чтение и пополняются, как правило, средствами массовой загрузки данных или пакетами SSIS/DTS) достаточно будет и нескольких процентов от объема файлов баз данных;
- второй фактор — какой режим восстановления настроен для базы данных. Режим восстановления (*recovery model*) настраивается на вкладке **Options** свойств базы данных, подробнее о нем будет рассказано в разд. 4.5. Режим восстановления **Simple** предъявляет минимальные требования к размеру файлов журнала (поскольку старые записи в журнале сразу же перезаписываются), а режим восстановления **Full** требует файлов журнала намного большего размера;
- если база данных работает в режиме восстановления **Full**, то записи в журнале транзакций будут копиться до бесконечности, пока не будет произведено резервное копирование журнала транзакций или журнал транзакций не будет очищен вручную. Поэтому при определении необходимого размера файлов журналов транзакций нужно учитывать и частоту резервного копирования.

4.4. Применение файловых групп

При создании файла базы данных вы можете указать, к какой файловой группе он будет относиться. Файловая группа (*Filegroup*) — это, как понятно из названия, способ организации файлов базы данных. По умолчанию для любой базы данных создается файловая группа `PRIMARY`, и все создаваемые файлы базы данных по умолчанию будут относиться именно к ней. В каких

ситуациях вам может потребоваться создание дополнительных файловых групп?

Первая ситуация — оптимизация резервного копирования. Предположим, что вы работаете с базой данных, все таблицы которой можно условно разделить на две части:

- пользовательские таблицы, которые постоянно изменяются пользователями;
- таблицы справочника, которые меняются очень редко (например, когда приходят обновления от разработчиков).

Предположим, что у пользовательских таблиц объем небольшой, а таблицы справочника, наоборот, занимают много места. В то же время, с точки зрения резервного копирования, намного важнее пользовательские таблицы. Оптимизировать резервное копирование в этой ситуации можно так:

- при создании базы данных создаем дополнительную файловую группу `USERS`. Создаем новый файл данных, например `users.mdf`, и определяем, что он будет относиться к этой группе;
- при создании пользовательских таблиц и индексов к ним определяем, что они будут принадлежать файловой группе `USERS` (для этой цели в командах `CREATE TABLE` и `CREATE INDEX` используется ключевое слово `ON` с указанием имени файловой группы). Обратите внимание, что назначать таблицам и индексам конкретные файлы нельзя, а файловые группы — можно;
- таблицы справочников оставляем в файловой группе `PRIMARY` или создаем для них свою файловую группу.

Далее будем производить резервное копирование отдельных файловых групп с разным расписанием. Например, резервное копирование файловой группы `USERS` можно производить каждый день, а файловой группы `PRIMARY` — раз в месяц.

Единственный момент, который при этом нужно учесть, связан с тем, что для обеспечения целостности данных между файловыми группами SQL Server не будет очищать журнал транзакций до тех пор, пока не будет произведено резервное копирование всех файловых групп (используется система так называемых поколений резервных копий). Таким образом, очистка журнала транзакций в этой ситуации будет производиться раз в месяц.

Вторая ситуация, когда нам может потребоваться использовать дополнительные файловые группы, — ручное распределение нагрузки в дисковой подсистеме. Предположим, например, что у вас на сервере есть два жестких диска: быстрый и относительно медленный. Файлы из файловой группы `USERS` можно разместить на быстром диске, а файлы для редкоиспользуемых справоч-

ников — на медленном, и, таким образом, можно повысить скорость работы системы для пользователей.

Третья ситуация — ручное распараллеливание запросов в дисковой подсистеме. Разместив, например, в разных файловых группах таблицу и индексы к ней, можно добиться, чтобы в обслуживании запроса к этой таблице принимали участие оба диска. Выигрыш, конечно, получится небольшим (RAID-массив, с точки зрения производительности, дает намного большие преимущества), тем не менее такая возможность существует.

И, наконец, четвертый, самый экзотический вариант использования файловых групп — оптимизация работы в многопроцессорных системах. Дело в том, что при обращении к каждой файловой группе открывается новый поток операционной системы. Конечно, существенный выигрыш в производительности при этом получить вряд ли удастся, но попробовать можно.

По опыту преподавания автором курсов по SQL Server и знакомства с задачами на разных предприятиях можно сказать, что файловые группы на предприятиях применяются очень редко (а многие администраторы и не подозревают об их существовании). Тем не менее в некоторых ситуациях они вполне могут пригодиться.

Создать новые файловые группы можно или при помощи графического интерфейса SQL Server Management Studio (на вкладке **Filegroups** (Файловые группы) свойств базы данных), или при помощи команд `CREATE DATABASE`, `ALTER DATABASE`. Создаваемую файловую группу можно сделать файловой группой по умолчанию. В этом случае все новые таблицы и индексы в базе данных при создании будут по умолчанию помещаться в эту файловую группу.

4.5. Режим восстановления базы данных

Одно из важных решений, которое нужно принять при создании базы данных — в каком режиме восстановления будет работать база. Этот параметр выбирается на вкладке **Options** свойств базы данных в строке **Recovery Model** (Режим восстановления) (над списком остальных параметров). Изменить режим восстановления базы данных можно также при помощи команды `ALTER DATABASE`.

Всего предусмотрено три режима восстановления базы данных.

- **Full** (режим полного протоколирования) — в этом режиме максимальное количество операций записывается в журнал транзакций. Журнал транзакций автоматически не обрезается. Этот режим обеспечивает максимальные возможности восстановления (за счет снижения производительности). Только в этом режиме вы можете использовать зеркальное отображение

баз данных и автоматическую доставку журналов (*log shipping*). Именно этот режим выбирается по умолчанию для пользовательских баз данных, поскольку он настроен для базы данных `model`. Если изменить режим восстановления для базы данных `model`, то для создаваемых баз данных по умолчанию будет выбираться новый режим.

□ **Bulk-logged** (режим неполного протоколирования) — это компромисс между требованиями производительности и возможностями восстановления. При использовании этого режима запись в журнал практически отключается (в терминологии Microsoft — проводится минимальное протоколирование) для операций следующих типов:

- массовой вставки (команды `BULK INSERT`, `SELECT INTO`, загрузка средствами `bcp` и т. п.);
- вставка/изменение больших двоичных данных (`text`, `ntext`, `image`);
- операции по созданию, перестроению и удалению индексов.

Автоматическая перезапись журналов транзакций при этом не производится, работа с транзакциями, не включающими в себя перечисленные операции, производится как обычно.

При работе в этом режиме вы лишаетесь возможности использовать журнал транзакций для восстановления (при утрате файлов данных, на момент времени или на метку транзакции), если в нем была хотя бы одна запись о перечисленных ранее операциях. Microsoft рекомендует не использовать этот режим восстановления на постоянной основе, а переключаться в него из режима **Full** на время выполнения больших операций массовой вставки, а потом возвращаться обратно.

□ **Simple** (простая модель восстановления) — максимальный выигрыш в производительности и удобстве работы за счет возможностей восстановления. Минимально протоколируются те же операции, что и в режиме восстановления **Bulk-logged**, а кроме этого, журнал транзакций автоматически очищается (блоками, размер которых изначально равен 256 Кбайт, но при необходимости он может быть автоматически увеличен). В результате вы получаете максимальную производительность и возможность не думать о потенциальной нехватке места в журнале транзакций. Но в этом режиме использовать журнал транзакций для восстановления уже удастся. Вы не сможете даже выполнить резервное копирование журнала транзакций: команда `BACKUP LOG` в этом режиме сразу вернет ошибку.

Какой же режим восстановления выбрать? Microsoft (в своих учебных курсах) рекомендует для рабочих баз данных выбирать только режим **Full**. Однако из опыта проведения автором этих самых учебных курсов и общения со слушателями можно сказать, что очень многие опытные администраторы

сознательно настраивают для своих баз данных режим восстановления **Simple**. Значительное повышение производительности при операциях массовой вставки и при работе с большими двоичными данными вполне оправдывает некоторое снижение возможностей резервного копирования и восстановления. Что важнее для вашей задачи — дополнительные возможности восстановления или максимальная производительность, решать вам.

4.6. Режимы работы базы данных

Для баз данных SQL Server 2005 предусмотрено несколько режимов работы (которые называются также состояниями базы данных, *database state*). Настраиваются они при помощи вкладки **Options** окна свойств базы данных или при помощи команды ALTER DATABASE (за исключением режима ONLINE/OFFLINE, который изменяется не на вкладке **Options**, а из контекстного меню базы данных в окне **Object Explorer**). Можно выбрать следующие режимы работы базы данных:

- режимы ONLINE/OFFLINE/EMERGENCY.** Режим ONLINE (оперативный режим) — это нормальный рабочий режим. Если перевести базу данных в режим OFFLINE (автономный режим), то:
 - база данных станет недоступной для пользователей;
 - на нее больше не будет расходоваться оперативная память сервера;
 - файлы базы данных и журнала транзакций освободятся, и их можно будет, например, скопировать средствами операционной системы.

Режим EMERGENCY (аварийный) — это новый режим, который появился в SQL Server 2005. Если перевести базу данных в режим EMERGENCY, то:

- она станет доступной только на чтение;
- будет отключено протоколирование (т. е. запись в журналы транзакций);
- к базе данных смогут обращаться только системные администраторы (т. е. члены серверной роли sysadmin).

Этот режим рекомендуется использовать для целей диагностики базы данных, если вы подозреваете, что в ней возникли проблемы;

- режимы READ-ONLY/READ-WRITE.** По умолчанию все базы данных находятся, конечно, в режиме READ-WRITE (чтение и запись). Перевод базы данных в режим READ-ONLY (только чтение) лишает пользователей возможности вносить изменения в данные, но серьезно ускоряет считывание данных за счет того, что никакие блокировки не накладываются. Чтобы перевести базу данных в режим READ-ONLY, нужно вначале отключить всех пользователей.

В этом режиме часто работают архивные хранилища данных Data Warehouses (на время пакетной загрузки данных их режим меняется на READ-WRITE);

- режимы MULTI_USER/RESTRICTED_USER/SINGLE_USER. Режим MULTI_USER (многопользовательский) — это обычный режим, в нем по умолчанию работают все базы данных. В режиме RESTRICTED_USER (ограничения доступа пользователей) в базу данных допускаются только пользователи, которые принадлежат к роли базы данных db_owner или к одной из серверных ролей sysadmin или dbcreator. Этот режим используется в ситуации, когда работа пользователей с базой данных нежелательна (массовая загрузка данных, обновление структуры, перестройка индексов), но нужно иметь возможность открывать несколько соединений с базой данных. В режиме SINGLE_USER (однопользовательский) разрешается только одно подключение к базе данных. Этот режим может использоваться, например, в разных аварийных ситуациях, когда производится восстановление базы данных.

Изменение режима работы базы данных требует отключения пользователей, которые в настоящее время работают с базой данных. Если в базе данных пользователи не работают, то изменение режима пройдет без каких-либо проблем. Если же в настоящее время к этой базе данных открыты соединения, то вам придется либо ответить на вопросы на графическом экране SQL Server Management Studio, либо предусмотреть соответствующий параметр в команде ALTER DATABASE. Вариантов изменения режима работы базы данных у вас четыре:

- вообще ничего не указывать. В этом случае команда ALTER DATABASE будет ждать бесконечное время, пока пользователи не закончат в базе данных свою работу. После этого команда переведет базу данных в нужный режим (если вы не хотите ждать, выполнение команды можно прервать);
- попытаться перевести базу данных в нужный режим без ожидания. Для этого используется параметр WITH NO_WAIT. Так же, как и в первом случае, переход в нужный режим произойдет, если нет пользовательских подключений, которые этому мешают. Если команда ALTER DATABASE не может изменить режим работы немедленно, она не будет ждать, а сразу вернет ошибку;
- указать, сколько секунд будет дано пользователям для завершения работы и до разрыва их соединений. Это делается при помощи параметра WITH ROLLBACK AFTER количество_секунд. Например, если вы хотите дать всем пользователям без административных прав 10 минут на завершение, то команда может выглядеть так:

```
ALTER DATABASE testdb SET RESTRICTED_USER WITH ROLLBACK AFTER 600;
```

На работу пользователей, которые подключились к базе данных с административными правами (т. е. с правами специальной роли `db_owner`) эта команда не повлияет;

- отключать пользователей немедленно, откатывая их незавершенные транзакции. Для этой цели используется параметр `WITH ROLLBACK IMMEDIATE`.

Можно рассмотреть еще один вариант, когда вы просто отслеживаете соединения всех пользователей через окно Activity Monitor (оно доступно из контейнера **Management** (Управление) в окне SQL Server Management Studio), а просматриваете, что они делают, а затем отключаете каждого по отдельности (возможно, предварительно предупредив их).

4.7. Другие параметры базы данных

Кроме режима восстановления и режимов работы, у баз данных SQL Server 2005 есть множество других важных свойств. Эти свойства можно настроить при помощи графического интерфейса на вкладке **Options** свойств базы данных в SQL Server Management Studio (при этом будут доступны не все параметры) или при помощи команды `ALTER DATABASE`. При этом названия параметров и их значения на графическом интерфейсе и в синтаксисе команды `ALTER DATABASE` выглядят по-разному. Например, на вкладке **Options** для параметра **ANSI NULLS Enabled** предусмотрены значения `TRUE` и `FALSE`. При использовании команды `ALTER DATABASE` тот же параметр называется `ANSI_NULLS`, и для него используются значения `ON` и `OFF`.

Далее приведена информация обо всех параметрах базы данных, которые не были описаны в двух предыдущих разделах. Названия параметров баз данных и их значения приводятся "в формате" команды `ALTER DATABASE`.

- **ALLOW_SNAPSHOT_ISOLATION** — определяет, можно ли при работе с этой базой данных использовать новый режим изоляции транзакций моментальных снимков. По умолчанию значение для этого параметра устанавливается в `OFF`.
- **ANSI_NULL_DEFAULT** — определяет, будут ли по умолчанию столбцы в создаваемых таблицах этой базы данных допускать значения типа `NULL`. По умолчанию значение этого параметра — `OFF` (т. е. не допускать), что нарушает стандарт ANSI. Этот параметр можно изменять на уровне отдельного сеанса. Все подключения по OLE DB и ODBC по умолчанию представляют его значение на уровне сеанса в `ON`.
- **ANSI_NULLS** — если включить этот параметр, то любые сравнения значений, по крайней мере, одно из которых является `NULL`, будут возвращать `NULL`. Такое поведение предписано стандартом ANSI. Если же этот параметр ус-

тановлен в `OFF` (по умолчанию), то SQL Server при сравнении двух значений типа `NULL` будет возвращать `TRUE`. Этот параметр предписывается обязательно устанавливать в `ON` при создании или изменении индексированных представлений или индексов для вычисляемых столбцов. Все подключения по OLE DB и ODBC также по умолчанию переставляют его значение на уровне сеанса в `ON`.

- **`ANSI_PADDING`** — определяет, будут ли сохраняться символы пустого пространства (например, пробелы) при вставке значений типа `varchar` и `nvarchar` в столбцы (значение `ON` предписано стандартом ANSI). По умолчанию в SQL Server установлено значение `OFF`. Точно так же, как и для предыдущего параметра, этот параметр предписывается обязательно устанавливать в `ON` при создании или изменении индексированных представлений или индексов для вычисляемых столбцов, а также он устанавливается в `ON` по умолчанию всеми подключениями OLE DB и ODBC на уровне сеанса. В документации Microsoft вообще рекомендуется, чтобы этот параметр всегда был установлен в `ON` (однако значение по умолчанию, как уже было сказано, — `OFF`).
- **`ANSI_WARNINGS`** — этот параметр определяет, будут ли генерироваться предупреждающие сообщения, если агрегатной функции предложили произвести деление на ноль или поработать со значением типа `NULL`. В отношении значений справедливо все то же самое, что и для предыдущего параметра: значение `ON` рекомендуется стандартом ANSI, по умолчанию устанавливается на уровне сеанса подключениями по OLE DB и ODBC, необходимо для работы с индексированными представлениями и индексами для вычисляемых столбцов. Но для баз данных SQL Server 2005 по умолчанию установлено значение `OFF`.
- **`ARITHABORT`** — если этот параметр установить в `ON`, то арифметическая ошибка (переполнение переменной или деление на ноль) приведет к остановке пакета или откату той транзакции, в которой она возникла. Если значение этого параметра установлено в `OFF` (по умолчанию в SQL Server), то все ограничится предупреждающим сообщением, а выполнение пакета/транзакции будет продолжено. Для работы с индексированными представлениями и индексами для вычисляемых столбцов значение этого параметра должно быть установлено в `ON`.
- **`AUTO_CLOSE`** — если этот параметр установлен в `ON`, то после отключения из базы данных последнего пользователя она будет автоматически закрыта SQL Server. Фактически при этом база данных перейдет в автономный режим: ее файлы можно будет спокойно копировать. При первом обращении к этой базе данных со стороны пользователей она будет автоматически открыта. По умолчанию этот параметр отключен для всех баз данных.

Включать его рекомендуется только для редкоиспользуемых баз данных для экономии ресурсов. Если его включить для обычной рабочей базы данных, к которой часто подключаются пользователи, то это может серьезно замедлить работу. В редакции SQL Server 2005 Desktop Edition, которая рассчитана на использование в качестве настольного приложения, этот параметр, наоборот, включен по умолчанию для всех баз данных.

- **AUTO_CREATE_STATISTICS** и **AUTO_UPDATE_STATISTICS** — если эти параметры включены (по умолчанию), то SQL Server 2005 автоматически создает и обновляет статистику для столбцов таблиц этой базы данных. Статистика — это специальная служебная информация о распределении данных в столбцах таблиц, которая используется оптимизатором запросов. Представьте, например, что у вас выполняется запрос, который просит вернуть всех Ивановых, проживающих в городе Санкт-Петербурге. При этом предположим, что у 90% записей в этой таблице одно и то же значение в столбце "Город" — Санкт-Петербург. Конечно, с точки зрения выполнения запроса, вначале выгоднее выбрать в таблице всех Ивановых (их явно будет не 90%), а затем уже проверять значение столбца "Город" для каждой отобранный записи. Однако для того, чтобы узнать, как распределяются значения в столбце, нужно вначале выполнить запрос. Поэтому SQL Server самостоятельно инициирует выполнение таких запросов, а потом сохраняет информацию о распределении данных (такая информация и называется статистикой) в служебных таблицах базы данных.

Статистика нужна обязательно, иначе оптимизатор не сможет создавать правильные планы выполнения запросов. Параметры **AUTO_CREATE_STATISTICS** и **AUTO_UPDATE_STATISTICS** имеет смысл отключать только тогда, когда база данных у вас очень большая, и вы боитесь, что активность по созданию статистики может замедлить работу пользователей. В этом случае можно производить обновление статистики вручную, например, во внебиржевое время.

- **AUTO_SHRINK** — при включении этого параметра файлы баз данных и журналов транзакций автоматически будут сжиматься, возвращая неиспользованное пространство операционной системе. По умолчанию этот параметр отключен. Для рабочих баз данных включение этого параметра может оказаться очень вредным из-за возрастания фрагментации.
- **CONCAT_NULL_YIELDS_NULL** — этот параметр иногда создает большие проблемы разработчикам и администраторам. Если его значение установлено в **ON**, то слияние обычного строкового значения со значением типа **NULL** даст в итоге **NULL**. Например, если мысливим имя и фамилию с незаполненным отцеством, то на выходе получится значение типа **NULL**. Если же значение этого параметра установлено в **OFF** (по умолчанию в SQL Server),

то вернется, как обычно, имя и фамилия. Проблема заключается в том, что по умолчанию подключения по OLE DB и ODBC устанавливают на уровне сеанса для этого параметра значение **ON**. Если приложение вы исправлять не можете, то один из возможных выходов — создать для столбца, в котором могут появиться пустые значения, безопасное значение по умолчанию (например, пустое строковое значение). Для работы с индексированными представлениями и индексированными столбцами значение этого параметра должно быть установлено в **ON**.

- **CURSOR_CLOSE_ON_COMMIT** — если включить этот параметр (по умолчанию он отключен), то курсор будет удаляться из памяти сразу после завершения транзакции. В противном случае он будет находиться в памяти до закрытия соединения или до явного удаления курсора пользователем. При включении этого параметра мы проигрываем в скорости выполнения пользователем некоторых операций, зато экономим ресурсы сервера. Этот параметр можно настроить на уровне отдельного сеанса при установке соединения.
- **CURSOR_DEFAULT** — для этого параметра можно использовать два значения: **LOCAL** и **GLOBAL** (по умолчанию установлено значение **GLOBAL**). Он определяет, курсоры какого типа будут создаваться по умолчанию: локальные (видимые только на уровне конкретного пакета, триггера, хранимой процедуры) или глобальные (видимые на уровне всего подключения). Какое значение выбирать в каждом конкретном случае — полностью зависит от разработчиков. Самим же разработчикам рекомендуется не полагаться на значение этого параметра, а явно определять в своем коде курсоры нужного типа.
- **DB_CHAINING** — этот параметр определяет, будет ли в этой базе данных разрешено участие в цепочках владения (*chaining ownership*) за пределами баз данных. По умолчанию установлено значение **OFF** — такое участие запрещено. Цепочки владений — это ситуация, которая возникает, когда один и тот же пользователь владеет и главным объектом (например, таблицей), и производным (например, представлением, которое обращается к этой таблице). За счет цепочек владения такой пользователь, предоставляя другому пользователю права на представление, тем самым неявно предоставляет тому права на работу с данными таблицы через это представление.
- **NUMERIC_ROUNDABORT** — если при вычислении в рамках запроса происходит потеря точности, и значение этого параметра установлено в **ON**, то возникает ошибка. По умолчанию в SQL Server для этого параметра используется значение **OFF**. Для возможности работы с индексированными представлениями и индексами для вычисляемых столбцов значение этого параметра должно быть установлено в **OFF** (а не **ON**, как для предыдущих параметров!).

□ **PAGE_VERIFY** — этот параметр призван заменить параметр **TORN_PAGE_DETECTION** в предыдущих версиях SQL Server (параметр **TORN_PAGE_DETECTION** оставлен для обратной совместимости, но пользоваться им уже не рекомендуется). **PAGE_VERIFY** определяет режим обнаружения ошибок ввода/вывода при работе со страницами базы данных. Обычно такие ошибки возникают по двум причинам:

- проблемы с диском (контроллером, драйвером контроллера и т. п.).
- в процессе записи в базу данных отключилось питание. SQL Server работает со страницами размером 8 Кбайт, а операционная система производит ввод/вывод обычно блоками по 512 байт. При отключении питания вполне может сложиться ситуация, когда страница в базе данных записана только частично. При обнаружении такой страницы подключение, которое к ней обращалось, автоматически закрывается, а в журнал событий записывается ошибка с кодом 824.

Для параметра **PAGE_VERIFY** предусмотрено три значения.

- **CHECKSUM** — это значение установлено для всех пользовательских баз данных по умолчанию. При использовании этого значения при каждом изменении для страницы будет рассчитываться контрольная сумма и записываться в заголовок страницы. Этот режим наилучшим образом выявляет большинство проблем со страницами баз данных (особенно дисковые проблемы), однако у него есть два недостатка: повышенный расход ресурсов по сравнению с двумя другими вариантами и менее качественное, чем при значении **TORN_PAGE_DETECTION**, обнаружение проблем, возникающих при неполной записи страниц во время отключения питания.
- **TORN_PAGE_DETECTION** — вместо контрольной суммы используется контрольный бит для каждого из блоков (размером 512 байт) страницы данных. Такой режим использует меньшее количество ресурсов и позволяет лучше, чем режим **CHECKSUM**, обнаруживать ошибки неполной записи страниц. Однако при использовании этого режима все остальные ошибки обнаруживаются хуже. Этот режим по умолчанию установлен только для базы данных **master** (и изменить его для этой базы данных нельзя).
- **NONE** — в этом режиме не будут использоваться ни контрольные суммы, ни контрольные биты. Обнаружение проблемных страниц будет затруднено, зато вы выиграете в производительности. Этот режим рекомендуется использовать только тогда, когда базу данных при необходимости можно будет легко воссоздать (например, информация в эту базу данных поступает при помощи репликации).

- **QUOTED_IDENTIFIER** — этот параметр определяет, можно ли использовать двойные кавычки для имен объектов (таблиц, столбцов и т. п.). Обычно такая необходимость возникает, если используется нестандартное имя объекта (например, состоящее из двух слов с пробелом). Однако в стандарте ANSI SQL для таких случаев предусмотрены квадратные скобки, которыми и рекомендуется пользоваться.

По умолчанию для этого параметра в SQL Server 2005 установлено значение **ON** — использовать двойные кавычки можно. Подключения по OLE DB и ODBC автоматически устанавливают значение для этого параметра на уровне сеанса в **ON**. Значение **ON** также необходимо для работы с индексированными представлениями или индексами для вычисляемых столбцов.

- **RECURSIVE_TRIGGERS** — этот параметр определяет, могут ли триггеры, настроенные для таблицы, своими действиями вызывать повторное срабатывание самих себя. По умолчанию для этого параметра установлено значение **OFF**, т. е. не могут. Однако для вашей таблицы может быть настроена целая система триггеров, которые могут вызывать срабатывание друг друга не напрямую. Чтобы запретить и такой вариант, нужно установить значение серверного параметра **NESTED TRIGGERS**.
- **TRUSTWORTHY** — этот параметр определяет, разрешается ли объектам данной базы (представлениям, пользовательским функциям, хранимым процедурам) обращаться к объектам за пределами данной базы данных (например, к таблицам в другой базе данных) в режиме имперсонации (другое название этого режима — делегирование). В режиме имперсонации при обращении к внешнему объекту для него передается информация об учетной записи того пользователя, который подключился к вашей базе данных. Получается, что права пользователя как бы "проходят" через вашу базу данных. В целях безопасности такой режим работы в SQL Server 2005 по умолчанию отключен на уровне базы данных (параметр **TRUSTWORTHY** установлен в **OFF**).

4.8. Расширенные свойства баз данных

Кроме возможности использовать существующие свойства баз данных, в вашем распоряжении также есть возможность создавать свойства самому. Свойства, которые вы можете создать сами, называются расширенными (*extended properties*). Их можно создать и настроить либо из свойств базы данных (в окне **Properties** на вкладке **Extended Properties** (Расширенные свойства)), либо из кода Transact-SQL при помощи хранимых процедур

`sp_addextendedproperty`, `sp_updateextendedproperty`, `sp_dropextendedproperty`. Просмотреть значения этих свойств можно на той же вкладки **Extended Properties** либо при помощи функции `fn_listextendedproperty`.

Как показывает опыт, многие администраторы и разработчики не знают о возможностях, которые предоставляют расширенные свойства, поэтому расскажем о них чуть подробнее.

Расширенные свойства предусмотрены не только для баз данных, но и для многих других объектов SQL Server 2005, например, для таблиц, столбцов таблиц, представлений, хранимых процедур, триггеров, функций и т. п. Расширенных свойств у объектов SQL Server 2005 может быть неограниченное количество. Для каждого из свойств предусмотрен единственный тип данных `sql_variant`, который вмещает в себя до 7500 байт данных.

Расширенные свойства используются для хранения любой информации, которую разработчик или администратор хочет сохранить вместе с данным объектом. Наиболее часто расширенные свойства для баз данных используются в качестве флага, который о чем-то сигнализирует. Например, в расширенные свойства можно поместить информацию о времени последней массовой загрузки данных в базу, о версии структуры базы данных, если разработчики периодически ее изменяют, о том, обрабатывалась ли она каким-то скриптом или приложением (например, если такое приложение на регулярной основе перестраивает все индексы) и т. п. На уровне столбца таблицы в расширенные свойства, например, можно поместить дополнительную информацию, которая может быть использована клиентским приложением для проверки вводимых значений.

Как правило, решение об использовании расширенных свойств принимается разработчиком, а изменение и чтение расширенных свойств производится внешним приложением при помощи команд Transact-SQL или при помощи объектов объектной модели SMO (SQL-DMO в предыдущих версиях SQL Server).

4.9. Выполнение некоторых служебных операций с базами данных

После того как база данных создана и настроена, для нее приходится с определенной периодичностью выполнять операции по обслуживанию. Некоторым из них посвящены отдельные главы книги (например, резервному копированию и восстановлению, автоматизации административных операций), а в этом разделе речь пойдет о тех операциях, которые не рассматриваются в других главах.

4.9.1. Увеличение размера базы данных

Эта операция выполняется очень просто. На графическом экране SQL Server Management Studio для этого достаточно открыть свойства базы данных, перейти на вкладку **Files** и ввести новый размер для файла базы данных в столбце **Initial Size** (Исходный размер) (или добавить в список новый файл). Можно также использовать команду Transact-SQL `ALTER DATABASE`. Например, чтобы увеличить размер файла `testdb1` для базы данных `testdb` до 10 Гбайт, можно использовать следующие команды:

```
USE master;
GO
ALTER DATABASE testdb MODIFY FILE (NAME = testdb1, SIZE = 10GB);
GO
```

Здесь `testdb1` — это логическое имя файла, которое можно увидеть в столбце **Logical Name** (Логическое имя) на вкладке **Files** свойств базы данных.

На всякий случай напомним, что из-за возможной фрагментации не рекомендуется включать для файлов базы данных режим автоматического увеличения размера.

4.9.2. Уменьшение размера базы данных

Иногда нужно не увеличить размер базы данных, а уменьшить ее размер, освободив дисковое пространство за счет неиспользуемого места в файлах базы данных. Обычная ситуация, когда этим приходится заниматься, — старые данные перенесены в архив и удалены из текущей базы данных.

Уменьшение размера файлов данных также можно произвести двумя способами:

1. На графическом экране SQL Server Management Studio в контекстном меню базы данных выбрать команду **Tasks | Shrink** (Задачи | Сжать), а затем в открывшемся окне выбрать, что вы хотите уменьшить: все файлы данных в базе данных или только выбранный файл.
2. Из скрипта Transact-SQL это можно сделать при помощи команд `DBCC SHRINKDATABASE` (для всех файлов базы данных) или `DBCC SHRINKFILE` (для отдельного файла).

При уменьшении размера файлов баз данных нужно учитывать следующие моменты:

- при уменьшении размера всей базы данных вы не сможете уменьшить ее менее исходного размера, который был определен при создании. Но добиться такого результата можно, если сжимать файлы базы данных по отдельности;

само уменьшение может производиться в четырех режимах:

- по умолчанию — все используемые страницы переносятся в начало файла, и пустое пространство освобождается для использования операционной системой;
- режим NOTRUNCATE — все используемые страницы переносятся в начало файла, но пустое пространство не возвращается операционной системе;
- режим TRUNCATEONLY — страницы внутри файла не переносятся, файл уменьшается только за счет пустого пространства в конце;
- режим EMPTYFILE — файл не уменьшается, но SQL Server 2005 пытается перенести все используемые страницы в нем в другие файлы той же файловой группы. Если это удается, то пустой файл можно будет удалить при помощи команды ALTER DATABASE;

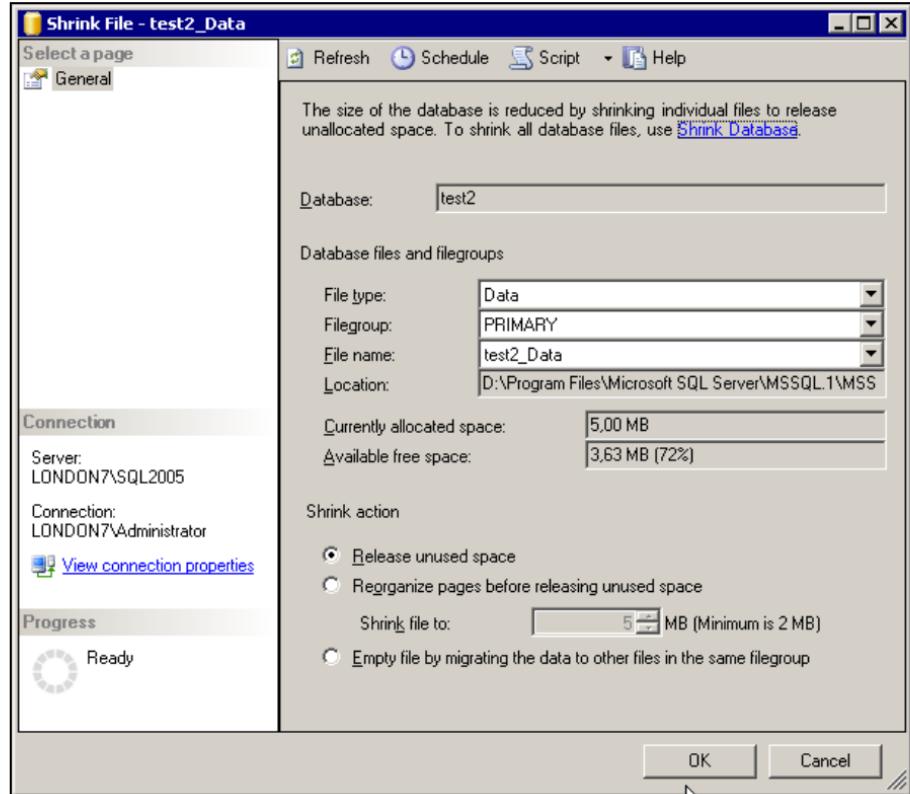


Рис. 4.2. Диалоговое окно Shrink File

- перед уменьшением может быть полезно получить информацию о том, сколько места можно высвободить в файле данных. Удобнее всего просмотреть эти данные в окне **Shrink File** (Сжать файл) в SQL Server Management Studio (рис. 4.2), которое открывается при помощи контекстного меню **Tasks | Shrink | Files** базы данных. Средствами Transact-SQL эту информацию можно получить при помощи команды DBCC SHOWFILESTATS (для файлов журналов транзакций — DBCC SQLPERF(LOGSPACE));
- если база данных, размер которой вы хотите уменьшить, будет в ближайшем будущем опять расти в размере, то лучше не прибегать к уменьшению: это вредно скажется как на внешней фрагментации (в файловой системе), так и на внутренней (с точки зрения организации страниц) фрагментации файлов данных.

И напомним, что в подавляющем большинстве случаев настройка автоматического уменьшения базы данных при помощи параметра AUTO_SHRINK — это очень вредная вещь.

4.9.3. Перенос файлов базы данных

Новая возможность SQL Server 2005 — возможность стандартными средствами, не прибегая к разным хитрым способам (например, резервному копированию и восстановлению, отсоединению и присоединению баз данных), переместить файлы базы данных в другое место. Такая необходимость на практике возникает достаточно часто, например, при изменении дисковой конфигурации сервера.

Перенос файлов базы данных производится очень просто:

- переведите базу данных в автономный режим, например:

```
ALTER DATABASE testdb SET OFFLINE;
```

- затем средствами операционной системы перенесите файлы баз данных в другое место;
- после этого укажите SQL Server, что файлы базы данных теперь находятся в другом месте:

```
ALTER DATABASE testdb MODIFY FILE (NAME = testdb, FILENAME = 'D:\testdb1.mdf');
```

- верните базу данных в обычный режим:

```
ALTER DATABASE testdb SET ONLINE;
```

Отметим, что таким способом можно перемещать не только пользовательские, но и системные базы данных (за исключением базы данных resource).

4.9.4. Переименование базы данных

Еще одна очень полезная возможность SQL Server 2005 — возможность переименовать базу данных. Это можно сделать тремя способами:

- на графическом экране из контекстного меню базы данных в SQL Server Management Studio при помощи команды **Rename** (Переименовать);
- при помощи команды ALTER DATABASE, например:

```
ALTER DATABASE currentdb MODIFY NAME = archivedb;
```

- при помощи хранимой процедуры sp_renamedb, например:

```
sp_renamedb 'currentdb', 'archivedb';
```

Хранимая процедура sp_renamedb оставлена только для обеспечения обратной совместимости с предыдущими версиями SQL Server, и ее использование не рекомендуется.

В любом случае перед переименованием базы данных вначале потребуется отключить от нее всех пользователей. В документации говорится также, что перед переименованием базу данных нужно перевести в однопользовательский режим, но, как показывает проверка, это не обязательно.

4.9.5. Изменение владельца базы данных

В SQL Server 2005 есть возможность менять владельца базы данных. Производится эта операция при помощи такой же хранимой процедуры sp_changedbowner, что в SQL Server 2000. Пример вызова этой хранимой процедуры может выглядеть так (меняется владелец текущей базы данных):

```
sp_changedbowner 'user1';
```

На графическом экране SQL Server Management Studio поменять владельца базы данных нельзя.

Произвести смену владельца для системных базы данных master, model и tempdb невозможно.

4.9.6. Удаление базы данных

Удалить базу данных в SQL Server 2005 можно двумя способами:

- из контекстного меню базы данных в SQL Server Management Studio по команде **Delete** (Удалить);
- при помощи команды DROP DATABASE, например:

```
DROP DATABASE testdb;
```

В любом случае удаляемая база данных не должна быть открыта ни пользователями, ни служебными процессами, такими как репликация.

При удалении базы данных SQL Server 2005 одновременно удаляются файлы этой базы данных на диске (если она не находилась в автономном режиме).

4.9.7. Проверка целостности базы данных

Иногда возникает необходимость убедиться в работоспособности базы данных, например, если начались проблемы с каким-то приложением или жалобы со стороны пользователей. Первое, что нужно сделать в этой ситуации, — обратиться к журналам событий операционной системы и SQL Server 2005. Если в них ничего подозрительного не замечено, возможно, имеет смысл провести проверку целостности базы данных. Для этого в SQL Server предусмотрен набор команд DBCC (DataBase Console Commands):

- **DBCC CHECKDB** — главная команда, которая используется для проверки целостности базы данных. Эта команда умеет проверять логическую и структурную целостность базы данных, находить ошибки в организации данных (например, страниц в файле). С ее помощью можно также попытаться исправить обнаруженные ошибки в разных режимах, но если у вас есть работоспособная резервная копия, предпочтительнее будет воспользоваться именно ею;
- **DBCC CHECKALLOC** — эта команда работает с набором операций, которые выполняет DBCC CHECKDB, и вместо нее правильнее будет использовать DBCC CHECKDB. Команда DBCC CHECKALLOC производит проверки только на наличие ошибок, связанных с организацией данных на диске. При помощи нее можно также попытаться исправить обнаруженные ошибки;
- **DBCC CHECKCATALOG** — еще одна команда, которая выполняет часть действий, выполняемых DBCC CHECKDB. Команда DBCC CHECKCATALOG проверяет структуру системных таблиц указанной базы данных. Исправлять ошибки эта команда не умеет.

Задание для самостоятельной работы 4.1. Перенос баз данных с SQL Server 2000 на SQL Server 2005

Ситуация:

Вам необходимо скопировать базы данных Northwind и Pubs с SQL Server 2000, который установлен на вашем локальном компьютере, на SQL Server 2005 (экземпляр *имя_вашего_компьютера\SQL2005*, который вы установи-

ли согласно заданию 2.1). При этом исходные базы данных должны остаться на сервере SQL Server 2000.

Задание:

1. Перенести на сервер SQL Server 2005 базу данных Northwind. Используйте для этой цели присоединение базы данных.
2. Перенесите на сервере SQL Server 2005 базу данных Pubs. Используйте для этого Copy Database Wizard.

Решение:

К пункту 1 задания — присоединение базы данных SQL Server 2000:

1. Откройте SQL Server Management Studio. В меню **File** выберите **Connect Server Explorer** (Подключить Server Explorer). В окне **Connect to Server** (Подключить к серверу) в поле **Server Type** (Тип сервера) выберите **Database Engine** (Ядро базы данных), в поле **Server Name** выберите **имя_вашего_компьютера** (например, LONDON2), в поле **Authentification** (Аутентификация) оставьте **Windows Authentification** и нажмите кнопку **Connect**. SQL Server Management Studio подключится к серверу SQL Server 2000.
2. Раскройте контейнер **Databases** для SQL Server 2000, щелкните правой кнопкой мыши по базе данных Northwind и в контекстном меню выберите **Tasks | Take Offline**. После окончания **операции** база данных Northwind будет помечена специальной красной меткой.
3. Найдите файлы базы данных Northwind (по умолчанию они находятся в каталоге C:\Program Files\Microsoft SQL Server\MSSQL\Data) и скопируйте файлы northwind.mdf и northwind.ldf в корневой каталог диска С:.
4. Еще раз в SQL Server Management Studio щелкните правой кнопкой мыши на базе данных Northwind на SQL Server 2000 и в контекстном меню выберите **Tasks | Bring Online** (Задачи | Перевести в оперативный режим).
5. В окне SQL Server Management Studio раскройте контейнер **Databases** уже для SQL Server 2005, щелкните по нему правой кнопкой мыши и в контекстном меню выберите **Attach**.
6. В окне **Attach Database** нажмите кнопку **Add** и выберите файл Northwind.mdf в корневом каталоге диска С:. Затем нажмите кнопку **OK**. После окончания присоединения новая база данных Northwind появится на SQL Server 2005.

К пункту 2 — применение Copy Database Wizard:

1. В окне SQL Server Management Studio раскройте контейнер **Databases** для SQL Server 2000 и щелкните правой кнопкой мыши по базе данных `Pubs`. В контекстном меню этой базы выберите **Tasks | Copy Database**. Откроется первый экран Copy Database Wizard. Нажмите на нем кнопку **Next**.
2. На экране мастера **Select a source server** (Выберите сервер-источник) оставьте значение по умолчанию (`имя_вашего_компьютера`) и нажмите кнопку **Next**.
3. На экране **Select a destination server** (Выберите сервер назначения) выберите нужный сервер SQL Server 2005 (`имя_вашего_компьютера\SQL2005`, например, `LONDON2\SQL2005`) и нажмите кнопку **Next**. Если появится предупреждающее сообщение о том, что SQL Server Agent на сервере назначения не запущен, нажмите в этом окне кнопку **No** и запустите SQL Server Agent. Для этого нужно в окне SQL Server Management Studio выбрать узел **SQL Server Agent** и воспользоваться командой **Start** (Старт) контекстного меню для него.
4. На экране **Select the Transfer Method** (Выберите метод переноса) оставьте переключатель в положении **Use the detach and attach method** (Использовать метод отсоединения и присоединения) и нажмите кнопку **Next**.
5. На экране **Databases** убедитесь, что установлен флажок только в столбце **Copy** напротив базы данных `Pubs`.
6. На экране **Configure Destination Database** (Настройте базу данных назначения) оставьте параметры, предлагаемые по умолчанию.
7. На экране **Select Database Objects** (Выберите объекты базы данных) нажмите кнопку **<<**, чтобы отменить копирование связанных с базой данных логинов.
8. На экранах **Configure the Package** (Настройте пакет) и **Schedule the Package** (Запланируйте пакет для выполнения) оставьте значения, предлагаемые по умолчанию, а затем на экране **Complete the Wizard** (Завершение работы мастера) нажмите кнопку **Finish**. После окончания копирования база данных `Pubs` появится на SQL Server 2005.



ГЛАВА 5

Безопасность SQL Server 2005

После того как SQL Server 2005 установлен и созданы рабочие базы данных, одна из следующих обязанностей администратора — позаботиться о разграничении доступа и защите данных SQL Server 2005. Конечно, об этом также должен думать и разработчик при создании приложения. Действительно, защищенные приложения, работающие с SQL Server 2005, — это результат совместной работы разработчиков и администраторов.

На практике очень часто разработчики не уделяют вопросам безопасности должного внимания, и все вопросы по обеспечению безопасности ложатся на администратора. Например, существует множество распространенных приложений, которые требуют, чтобы все пользователи подключались к приложению с правами системного администратора. Другие приложения используют без шифрования сетевого трафика роли приложений, а некоторые применяют Web-интерфейс, при подключении к которому пользователи передают свои пароли и данные открытым текстом. Поэтому администраторы, которые обычно и ответственны за защиту данных, должны не только обеспечивать защищенную работу приложения, но и при приеме приложения в эксплуатацию требовать от разработчиков (если есть такая возможность) использования средств защиты данных.

Система безопасности SQL Server 2005 по сравнению с предыдущими версиями изменилась очень сильно. Поменялись терминология, функциональные возможности, настроенные разрешения по умолчанию. Появилось ожидаемое в течение многих лет встроенное шифрование информации в базах данных. Фактически безопасность SQL Server стала намного мощнее и сложнее. О том, как выглядит система безопасности SQL Server 2005, и о практическом применении ее возможностей пойдет речь в этой главе.

5.1. Терминология и основы системы безопасности SQL Server 2005

В этом разделе представлены основные термины и концепции системы безопасности, которые помогут нам ориентироваться в следующих разделах этой главы.

Принципалы (principals) — это те объекты, которым в SQL Server 2005 можно предоставлять разрешения. Они могут быть как индивидуальными (например, учетная запись), так и групповыми (например, роль). Далее приведен полный список принципалов в системе безопасности SQL Server 2005, которые существуют на трех уровнях:

- на уровне операционной системы Windows:
 - логин для доменной учетной записи Windows (*Windows Domain login*) — учетная запись, созданная на уровне SQL Server 2005 для подключения от имени учетной записи Windows. Чтобы отличать доменные учетные записи от учетных записей Windows, в этой книге первые будут называться логинами (как обычно они и называются на предприятиях);
 - логин для локальной учетной записи Windows (*Windows Local login*);
- на уровне сервера SQL Server 2005:
 - логин SQL Server 2005 (*SQL Server login*);
- на уровне базы данных:
 - пользователь базы данных (*database user*);
 - роль базы данных (*database role*);
 - роль приложения (*application role*).

Securables (дословно "защищаемые") — еще одна важнейшая концепция системы безопасности SQL Server 2005. Это все, на что в SQL Server 2005 можно назначить разрешения. Они также относятся к трем уровням SQL Server 2005, но уровни другие:

- на уровне сервера SQL Server:
 - логин (теперь можно одному пользователю SQL Server 2005 предоставить разрешения на объект другого пользователя);
 - база данных;
 - точка подключения (*endpoint*), т. е. можно предоставить разрешения на точки подключения по HTTP;

на уровне базы данных:

- пользователь;
- роль;
- роль приложения;
- сборка;
- тип сообщения;
- маршрут;
- служба;
- привязка удаленной службы;
- полнотекстовый каталог;
- сертификат;
- асимметричный ключ;
- симметричный ключ;
- контракт;
- схема;

 на уровне схемы:

- таблица;
- представление;
- функция;
- процедура;
- ограничение целостности;
- очередь;
- статистика;
- пользовательский тип данных;
- синоним;
- агрегат;
- коллекция схем XML.

Как вы видите, в SQL Server 2005 появилось множество новых объектов, на которые можно предоставлять разрешения.

В большинстве случаев процесс предоставления разрешений выглядит очень просто:

1. Создать логин — учетную запись для подключения к SQL Server.
2. Затем создать пользователя базы данных, которому соответствует этот логин.
3. Предоставить пользователю необходимые разрешения.

Однако в этой простой схеме кроется множество тонкостей и дополнительных возможностей, некоторые из них появились только в SQL Server 2005. Их рассмотрению посвящены следующие разделы этой главы.

5.2. Логины SQL Server 2005

5.2.1. Выбор типа логина

Первое, что нужно сделать при предоставлении разрешений пользователю на информацию в SQL Server 2005, — это предоставить ему логин, т. е. учетную

запись, которая будет использоваться для подключения к серверу SQL Server. Однако прежде, чем создавать логин, необходимо понять, какой тип вы будете использовать для этого логина.

В SQL Server 2005 придется выбирать из тех же двух главных типов, которые встречались в предыдущих версиях:

- логин Windows** (разновидности: логин для локальной учетной записи Windows, логин для доменной учетной записи Windows, логин для группы Windows);
- логин SQL Server.**

Отличия между двумя этими типами логинов такие же, как и в предыдущих версиях SQL Server. При использовании логинов Windows в системные таблицы базы данных `master` записывается информация об идентификаторе учетной записи или группы Windows (но не пароль). Аутентификация (т. е. проверка имени пользователя и пароля) производится обычными средствами Windows при входе пользователя на свой компьютер.

При использовании логина SQL Server пароль для этого логина (точнее, его хэшированное значение) хранится вместе с идентификатором логина в базе данных `master`. При подключении пользователя к серверу ему придется указать имя логина и пароль. Фактически в этом случае пользователю придется помнить два набора логин/пароль: один — для входа на компьютер, а второй — для подключения к SQL Server.

Какой из двух главных типов логинов следует выбирать? Обычно перед администратором такой вопрос не стоит: механизм подключения клиентского приложения и тип учетных записей выбирается разработчиком приложения. Если вы сами разработчик и имеете возможность принимать решения самостоятельно, то идеальный вариант логина для пользователя — это логин Windows, при этом не для учетной записи, а для группы (лучше всего для локальной доменной группы). Преимуществ у такого решения множество:

- пользователю достаточно помнить один пароль — для входа на свой компьютер. Администраторы знают, как сложно заставить пользователя запомнить хотя бы один длинный пароль. Необходимость помнить дополнительные пароли (например, пароль логина SQL Server) — это лишняя нагрузка на пользователей и администраторов;
- повышается уровень защищенности SQL Server. Это происходит, по крайней мере, за счет того, что пароль не будет передаваться по сети открытым текстом, как это происходит по умолчанию при использовании команд `CREATE LOGIN` и `ALTER LOGIN`. Кроме того, хэши Windows (в версии NTLMv2, которая используется с Windows 2000) намного более защищены, чем хэши логинов SQL Server, которые можно вскрыть достаточно

быстро (пример свободно доступной в Интернете программы, которая перехватывает хэши SQL Server и подбирает пароли, — Cain&Abel);

- проверка при входе пользователя производится быстрее.

Эти преимущества справедливы для любых логинов Windows: как для обычных учетных записей, так и для групп. Но идеальный вариант все-таки — использование учетной записи группы. Причины просты:

- снижается размер системных таблиц базы данных master, в результате чего аутентификация производится быстрее. На одном сервере SQL Server вполне может быть несколько тысяч логинов для пользователей Windows или, что значительно удобнее, всего пара десятков логинов для групп;
- резко упрощается предоставление разрешений для новых учетных записей. Предположим, что у вас появился новый пользователь. Если на вашем предприятии принято предоставлять разрешения каждому пользователю отдельно, то для этого пользователя придется создавать логины на каждом сервере SQL Server (а таких серверов на крупном предприятии вполне может быть несколько). Если же вы изначально используете логины для групп Windows, то тогда при появлении нового пользователя достаточно будет на уровне Windows скопировать для него учетную запись другого пользователя с похожими рабочими обязанностями. При копировании учетной записи Windows членство в группах (в отличие от явно назначенных разрешений) наследуется от исходной записи: в результате пользователь получит требуемые права на SQL Server через членство в группе.

Однако мы живем не в идеальном мире. Поэтому, по наблюдениям автора, примерно 80% реальных задач на предприятиях используют логины SQL Server, жертвуя при этом и удобством пользователей, и безопасностью, и производительностью. Аргумент в пользу этого решения очень простой и весомый: очень часто на предприятиях администрированием SQL Server и домена Windows занимаются разные люди, которым сложно согласовывать свои действия. Логины SQL Server позволяют администратору SQL Server быть независимым от администратора домена. Кроме того, у логинов SQL Server есть и другие преимущества, которые принимаются во внимание разработчиками:

- на предприятии вполне может не оказаться домена Windows (если, например, сеть построена на основе NetWare или UNIX);
- пользователи SQL Server могут не входить в домен (например, если они подключаются к SQL Server из филиалов или через Web-интерфейс с домашнего компьютера).

Так что решение остается за вами. Логины Windows — это удобство и защищенность, логины SQL Server — это большая гибкость и независимость от администратора сети.

5.2.2. Создание логина и настройка его параметров

Логины любого типа создаются одинаково:

- при помощи графического интерфейса — из окна **Login — New** (Новый логин). Это окно открывается с помощью команды **New Login** контекстного меню контейнера **Security | Logins** (Безопасность | Логины) в **Object Explorer** в SQL Server Management Studio;
- из скрипта — при помощи команды `CREATE LOGIN`. Хранимая процедура `sp_addlogin`, которая использовалась для этой цели в предыдущих версиях SQL Server, оставлена только для обеспечения обратной совместимости и к использованию не рекомендуется.

Например, команда на создание логина SQL Server с именем `User1` и паролем `P@ssw0rd` (для всех остальных параметров будут приняты значения по умолчанию) может выглядеть так:

```
CREATE LOGIN User1 WITH PASSWORD = 'P@ssw0rd';
```

Если вы создаете логин Windows, вам потребуется выбрать соответствующую учетную запись Windows (доменную или локальную). Не забудьте о возможности использовать группы Windows.

Если вы создаете логин SQL Server, вам придется ввести его имя и пароль. Пароль всегда чувствителен к регистру, а логин — только тогда, когда чувствительность к регистру была определена при установке SQL Server 2005.

Конечно, кроме имени и пароля для логинов можно определить множество других параметров. Некоторые из них появились только в SQL Server 2005. Далее эти параметры перечислены с комментариями.

Enforce password policy (Использовать парольную политику) — эта новая возможность (ее можно использовать, только если SQL Server работает под управлением Windows 2003 Server) позволяет определить требования к паролям для логинов SQL Server. Однако настроить эти требования на уровне SQL Server вы не можете: если флагок **Enforce password policy** будет установлен, то на пароль для данного логина просто будут распространены те требования, которые применяются к локальным учетным записям на данном компьютере. Проще всего просмотреть параметры парольной политики из редактора групповой политики, подключив его к локальному компьютеру. Последовательность действий при этом может выглядеть так:

1. Установить необходимую консоль на компьютер. Для этого в командной строке Windows 2000 Server или Windows 2003 достаточно выполнить команду `Adminpak.msi`. В Windows XP потребуется предварительно скопи-

ровать этот файл с компьютера, на котором установлен Windows Server 2003 (или с дистрибутива Windows 2003 Server).

2. Открыть эту консоль. Проще всего это сделать так:

- в командной строке выполнить команду MMC;
- в меню **Консоль** выбрать команду **Добавить или удалить оснастку**;
- в открывшемся окне нажать кнопку **Добавить**;
- в списке оснасток выбрать **Редактор объектов групповой политики**;
- на экране **Выбор объекта групповой политики** выбрать **Локальный компьютер**.

3. Раскрыть узел **Политика "Локальный компьютер" | Конфигурация компьютера | Конфигурация Windows | Параметры безопасности | Политики учетных записей | Политика паролей** и посмотреть, что ожидает пользователей SQL Server.

Если компьютер, на котором установлен SQL Server 2005, не входит в домен, то к логинам SQL Server 2005 по умолчанию будут применяться требования, представленные на рис. 5.1.

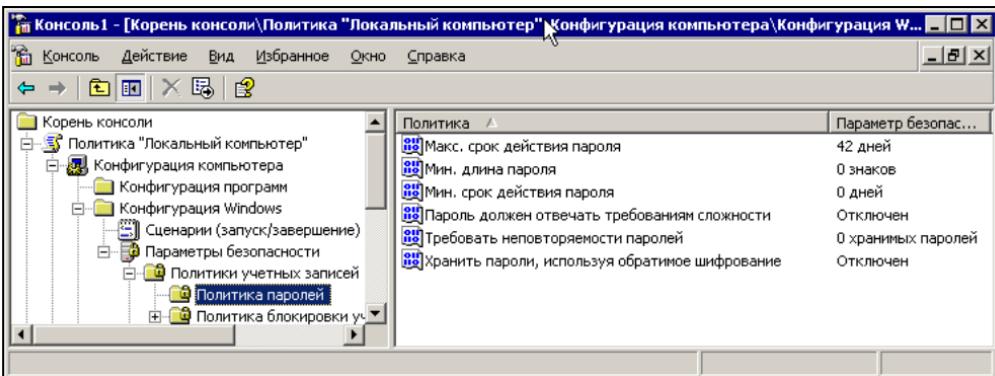


Рис. 5.1. Требования к паролям, применяемые по умолчанию для логинов SQL Server 2005

Если компьютер находится в домене, то все зависит от администратора домена.

Обратите внимание, что при установленном флагке **Enforce password history** (Использовать парольную политику) на логин распространяются не только требования к паролям, но и политики блокировки учетных записей, принятые на этом компьютере. Они настраиваются из той же консоли при помощи контейнера **Политика блокировки учетной записи**. Те параметры, которые можно настроить, и их значения по умолчанию представлены на рис. 5.2.

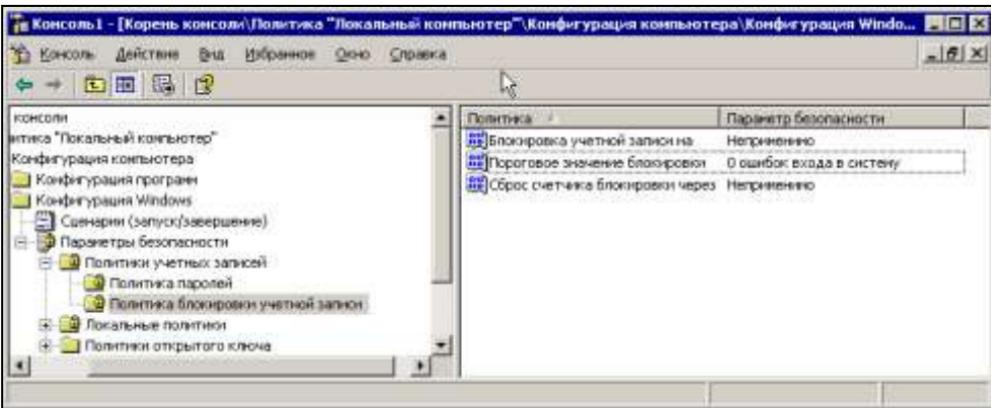


Рис. 5.2. Параметры блокировки паролей для логинов SQL Server 2005 по умолчанию

Enforce password expiration (Включить устаревание пароля) — этот параметр определяет, будут ли на логин SQL Server распространяться те же требования по смене пароля через определенный промежуток времени, что и для учетных записей Windows. Этот флагок можно установить только при установленном флагке **Enforce password history**.

С точки зрения автора, флагок **Enforce password expiration** лучше не устанавливать, поскольку тогда по умолчанию через 42 дня при входе пользователя в сеть потребуется смена пароля. В большинстве случаев клиентское приложение не обеспечивает необходимый интерфейс, и пользователь просто перестанет входить в SQL Server.

User must change password at next logon (Пользователь должен поменять пароль при следующем входе) — с этим флагком нужно быть исключительно осторожным. Окно для смены пароля появляется только в SQL Server Management Studio. Большинство других приложений при попытке подключиться от имени такого пользователя просто вернут ошибку. Работать этот флагок будет, только если возможность смены пароля предусмотрел разработчик клиентского приложения, что бывает далеко не всегда.

Отметим еще один момент, связанный с паролями. Как показывает опыт, даже если клиентское приложение предоставляет пользователю возможность поменять свой пароль для логина SQL Server самостоятельно, не стоит надеяться на то, что пользователь сможет сам придумать действительно защищенный пароль, а не простую комбинацию типа "123". По глубокому убеждению автора, пароли для пользователей должен придумывать тот, кто отвечает за безопасность системы, т. е. администратор SQL Server 2005.

Mapped to certificate (Привязан к сертификату) и **Mapped to asymmetric key** (Привязан к асимметричному ключу) — эти параметры позволяют просмот-

реть сертификат или асимметричный ключ, который будет применяться для данного пользователя. Назначить сертификат можно только при помощи команды Transact-SQL `CREATE LOGIN`. Подробнее про использование сертификатов и асимметричных ключей для аутентификации и шифрования данных будет рассказано в разд. 5.6.

База данных **Default database** (База данных по умолчанию), к которой по умолчанию будет подключаться пользователь при входе на SQL Server. По умолчанию используется база данных `master`. Как правило, менять этот параметр не следует: код для перехода к нужной базе данных при подключении обеспечивает клиентское приложение.

Default language (Язык по умолчанию) — язык, который будет использоваться по умолчанию данным пользователем во время сеансов. В основном он влияет на формат даты и времени, которые возвращает SQL Server. В большинстве случаев для этого параметра оставляется значение по умолчанию (т. е. язык, настроенный на уровне всего сервера), если о другом значении специально не просит разработчик.

На вкладке **Status** (Состояние) свойств логина можно настроить для этого логина дополнительные параметры:

- **Permissions to connect to database engine** (Разрешение на подключение к ядру баз данных) — по умолчанию для всех логинов устанавливается значение `Grant`, т. е. подключаться к SQL Server разрешено. Значение `Deny`, как правило, используется только в одном случае — когда вы предоставляете доступ на SQL Server 2005 логину для группы Windows, а одному или нескольким членам этой группы доступ нужно запретить. Поскольку явный запрет всегда имеет приоритет перед разрешением, то достаточно будет создать свои собственные логины Windows для этих пользователей и установить для них значение `Deny`.
- **Login enabled/disabled** (Логин включен/отключен) — конечно, все логины по умолчанию включены. Обычно отключать их приходится только в ситуации, когда какой-то пользователь увольняется или переходит на другую работу. Чтобы сэкономить время, достаточно просто отключить данный логин, а при появлении пользователя со схожими рабочими обязанностями переименовать этот логин, поменять пароль и включить. Заниматься предоставлением разрешений заново в этом случае не придется.

- **Login is locked out** (Логин заблокирован) — установить этот флагок вы не можете (только снять его). Учетная запись пользователя блокируется автоматически после нескольких попыток неверного ввода пароля для логина SQL Server, если такая блокировка настроена на уровне операционной системы, а для логина установлен флагок **Enforce password policy**.

Для логина предусмотрено также несколько "секретных" свойств, которые доступны только при использовании команд CREATE LOGIN, ALTER LOGIN или специальных функций Transact-SQL. Графического интерфейса для их изменения не предусмотрено. Все эти свойства являются новыми по отношению к предыдущим версиям SQL Server, поэтому далее приведен их перечень с комментариями.

- **HASHED** — указывает, что вы передаете не пароль, а его хэшированное значение. Это свойство стоит использовать только в том случае, если вы создаете свою систему безопасности для приложения.
- **CREDENTIAL** — назначает логину объект *Credential* (который представляет собой набор "имя учетной записи Windows/пароль"). Объект Credential обычно используется в ситуациях, когда пользователь из кода Transact-SQL должен выполнить какие-то действия в операционной системе или на другом сервере SQL Server. Эти действия он сможет выполнить от имени учетной записи, которая определена при помощи объекта Credential. Сам объект можно создать из контейнера **Credentials** в SQL Server Management Studio или при помощи команды CREATE CREDENTIAL.
- **SID** — позволяет явно назначить логину глобально-уникальный идентификатор безопасности (если его не указать, то он будет сгенерирован автоматически). Выглядеть он может, например, так: 0xD3B670F1A11E6C41B8F965EA3C2E189E. Просмотреть его для существующего пользователя можно при помощи функции `SUSER_SID`. Настраивать его стоит только в том случае, если он будет использоваться в таблицах ваших баз данных или для дополнительных проверок.
- **ID** — идентификатор логина в системных таблицах SQL Server. Если **SID** — это глобально-уникальный идентификатор, то **ID** — это просто номер, который может выглядеть, например, так: 266. Настроить его нельзя, а просмотреть можно при помощи функции `SUSER_ID`.

Параметры, которые есть на других вкладках свойств логина (**Server Roles**, **User Mapping**, **Securables**), будут рассмотрены в следующих разделах данной главы.

5.2.3. Режимы аутентификации. Аудит попыток входа

В предыдущем разделе рассматривалось создание логинов — учетных записей для подключения к SQL Server 2005. Но если вы создали логины SQL Server 2005, то вполне можете столкнуться с ситуацией, когда они работать не будут. Причина может заключаться в том, что не настроен требуемый режим аутентификации SQL Server 2005.

Режим аутентификации устанавливается еще при установке SQL Server 2005 (см. разд. 2.2.5). Поменять этот режим после установки можно на вкладке **Security** свойств SQL Server 2005 (рис. 5.3) в SQL Server Management Studio.

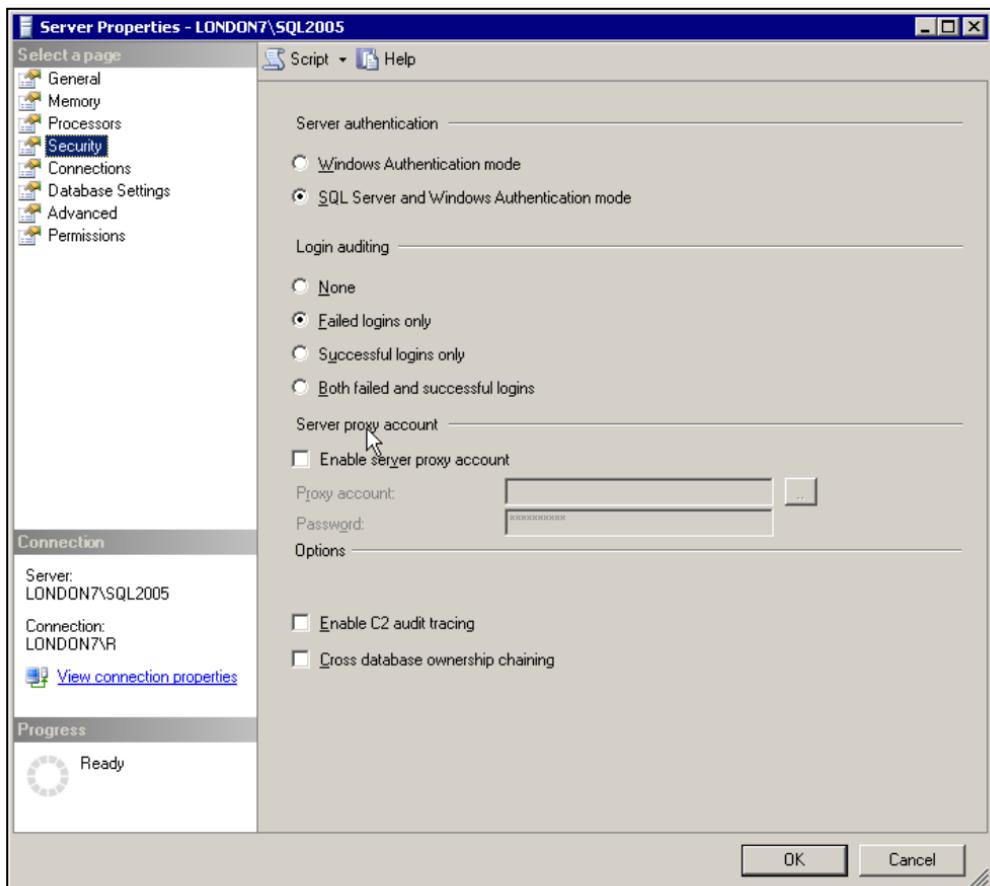


Рис. 5.3. Вкладка **Security** свойств SQL Server 2005

Как и в SQL Server 2000, в вашем распоряжении два режима аутентификации:

- Windows Authentication mode** (Режим аутентификации Windows) — этот режим, который выбирается по умолчанию при установке SQL Server 2005, разрешает использовать для подключения к серверу только логины Windows;
- SQL Server and Windows Authentication mode** (Режим аутентификации SQL Server и Windows) — в этом режиме можно использовать оба типа логинов — и логины SQL Server, и логины Windows. Другое название этого режима — Mixed mode (Смешанный режим).

Третьего варианта, в котором использование логинов Windows было бы запрещено, не предусмотрено: логины этого типа доступны всегда.

При изменении режима аутентификации сервер нужно перезапустить.

Отметим еще один момент: если вы переходите из режима аутентификации Windows в смешанный режим, то учетная запись sa по умолчанию останется отключенной. Ее придется включить при помощи команды ALTER LOGIN или из Management Studio (параметр **Login Enabled/Disabled** на вкладке **Status** свойств логина).

Если к вашему серверу применяются серьезные требования по безопасности, то вы можете включить аудит входов на SQL Server 2005. Выполняется эта операция также на вкладке **Security** свойств сервера. При помощи переключателя **Login Auditing** (Аудит входов) вы можете выбрать, аудит каких попыток входа будет производиться:

- None** — никаких;
- Failed logins only** — только неудачных (этот режим используется по умолчанию);
- Successful logins only** — только удачных;
- Both failed and successful logins** — любых.

Другой вариант включения аудита — установить флагок **Enable C2 audit tracing** (Включить трассировку аудита C2) на той же вкладке **Security** свойств SQL Server. В этом случае подробная информация о любых действиях пользователей (включая вход на сервер) будет записываться в текстовые файлы в каталог Data для данного экземпляра сервера. С этим параметром нужно быть очень осторожным, потому что информации будет записываться очень много и место на диске может закончиться.

5.2.4. Логины, создаваемые по умолчанию

Сразу же после установки SQL Server 2005 в контейнере **Logins** появляется набор логинов, которые создаются автоматически. Скорее всего, для подключения пользователей вы не будете их использовать. Тем не менее возникают ситуации, в которых знание встроенных логинов может пригодиться (например, если будет нечаянно заблокирован ваш административный логин).

- BUILTIN\Администраторы** (или BUILTIN\Administrators, в зависимости от языка операционной системы) — логину для этой группы Windows автоматически предоставляются права системного администратора SQL Server. Обратите внимание, что, если компьютер входит в домен, в эту группу автоматически попадает группа **Domain Admins** (Администраторы домена), и, таким образом, администраторы домена по умолчанию обла-

дают полными правами на SQL Server. Если такая ситуация нежелательна, то этот логин можно удалить. Но и в этом случае администраторам домена получить доступ к данным SQL Server будет несложно.

- `Имя_сервера2005MSFTEUser$Имя_сервера$Имя_экземпляра,`
`Имя_сервера2005MSSQLUser$Имя_сервера$Имя_экземпляра,`
`Имя_сервера2005SQLAgentUser$Имя_сервера$Имя_экземпляра` —

эти три логина для групп Windows используются для подключения соответствующих служб к SQL Server 2005. На уровне SQL Server 2005 с ними нет необходимости производить какие-то операции, поскольку все необходимые права уже предоставлены. В редких ситуациях вам может потребоваться добавить в эти группы на уровне Windows учетные записи, от имени которых работают службы SQL Server.

- `NT AUTHORITY\NETWORK SERVICE` — от имени этой учетной записи в Windows Server 2003 работают приложения ASP.NET, в том числе и службы Reporting Services (в Windows 2000 для этой цели используется учетная запись `ASPNET`). Этот логин Windows используется для подключения к SQL Server Reporting Services. Ему автоматически предоставляются необходимые права на базы данных `master`, `msdb` и на базы данных, используемые Reporting Services.
- `NT AUTHORITY\SYSTEM` — это локальная системная учетная запись операционной системы. Такой логин появляется в тех ситуациях, когда вы при установке настроили работу службы SQL Server от имени локальной системной учетной записи. Можно сказать, что при помощи этого логина SQL Server обращается к самому себе. Конечно же, этот логин обладает правами системного администратора SQL Server.
- `sa` (от *System Administrator*) — это единственный логин типа SQL Server, который создается по умолчанию. Он обладает правами системного администратора SQL Server, и отобрать эти права у него нельзя. Удалить этот логин также не получится. Зато его можно переименовать или отключить. Если для SQL Server 2005 будет настроена аутентификация только средствами Windows, то использовать этот логин для подключения к серверу не удастся.

5.2.5. Серверные роли. Разрешения на уровне сервера

Вы обеспечили пользователям возможность входа на SQL Server 2005, создав для них логины. Но сам по себе вход на сервер ничего не дает: пользователю нужны также права на выполнение определенных действий. Обычно для этой цели создаются пользователи или роли баз данных и им предоставляются

разрешения (как это сделать, будет рассмотрено в разд. 5.3). Однако есть и другой способ. Если вам нужно предоставить пользователю права на уровне всего сервера, а не отдельной базы данных, можно воспользоваться серверными ролями.

На графическом экране работа с ролями сервера производится или из свойств логина (вкладка **Server Roles** (Серверные роли)), или из свойств самой серверной роли (контейнер **Server Roles** в Management Studio). Из кода Transact-SQL для назначения логину серверной роли можно использовать хранимую процедуру `sp_addsrvrolemember`. Например, чтобы предоставить пользователю `User4` права роли `SYSADMIN`, соответствующий код может быть таким:

```
EXEC sp_addsrvrolemember @loginame = 'user4', @rolename = 'sysadmin';
```

Сразу отметим несколько моментов, связанных с серверными ролями:

- набор серверных ролей является фиксированным. Вы не можете создавать свои серверные роли (в отличие от ролей базы данных);
- для предоставления прав на уровне всего сервера необязательно использовать серверные роли. Вы вполне можете предоставить эти права напрямую логину (при помощи вкладки **Permissions** (Разрешения) свойств SQL Server). По умолчанию каждый логин обладает на уровне всего сервера двумя правами: `CONNECT SQL` (т. е. подключаться к серверу, это право предоставляется логину напрямую) и `VIEW ANY DATABASE` (т. е. просматривать список баз данных, это право пользователь получает через серверную роль `PUBLIC`);
- серверные роли используются только в специальных случаях. Для большинства пользователей настраивать их не нужно.

Серверных ролей не так много, поэтому приведем их полный список с комментариями:

- **PUBLIC** — эту роль вы не найдете в списке серверных ролей. Тем не менее она существует и активно используется. Права этой роли автоматически получают все, кто подключился к SQL Server, и лишить пользователя членства в этой роли нельзя. Обычно эта роль используется для предоставления разрешений всем пользователям данного сервера;
- **SYSADMIN** — логин, которому назначена эта роль, получает полные права на весь SQL Server (и возможность передавать эти права другим пользователям). С точки зрения серверных разрешений, это соответствует праву `CONTROL SERVER` с параметром `WITH GRANT` (т. е. с возможностью передачи);
- **SERVERADMIN** — эта роль для оператора, который обслуживает сервер. Можно изменять любые настройки работы сервера и отключать сервер, но получать доступ к данным и изменять разрешения нельзя;

- SECURITYADMIN** — эта роль для того, кто выдает разрешения пользователям, не вмешиваясь в работу сервера. Он может создавать логины для обычных пользователей, изменять их пароли, предоставлять разрешения в базах данных. Однако предоставить кому-либо административные права или изменять учетную запись администратора эта роль не позволяет;
- BULKADMIN** — роль для сотрудников, которые выполняют массовые загрузки данных в таблицы SQL Server;
- DBCREATOR** — эта роль позволяет создавать базы данных на SQL Server (а тот, кто создал базу данных, автоматически становится ее владельцем). Обычно эта роль предоставляется учетным записям, используемым приложениями, которые хранят свою информацию на SQL Server;
- DISKADMIN** — эта роль позволяет выполнять любые операции с файлами баз данных на диске;
- PROCESSADMIN** — эта роль предназначена для выполнения единственной обязанности: закрытия пользовательских подключений к серверу (например, зависших);
- SETUPADMIN** — права этой роли позволяют подключать внешние серверы SQL Server (*linked servers*).

Дополнительные возможности работы с правами на уровне сервера доступны из свойств сервера на вкладке **Permissions** (рис. 5.4).

На этой вкладке вы можете:

- более точно настроить права для каждого логина (если набор серверных ролей вас не устраивает);
- предоставить права всем пользователям сразу (при помощи специальной серверной роли **PUBLIC**);
- посмотреть, кто предоставил данные права пользователю (значение в столбце **Grantor**);
- посмотреть итоговые права для данного пользователя на уровне сервера. Для этой цели предназначена кнопка **Effective Permissions** (Действующие разрешения).

Еще одна возможность предоставить пользователю разрешения на объекты сервера — воспользоваться вкладкой **Securables** свойств логина. При помощи этой вкладки вы можете предоставить пользователю разрешения на:

- объект сервера (то же самое, что и из свойств сервера);
- объекты подключений по HTTP (*HTTP Endpoints*);
- объекты других логинов.

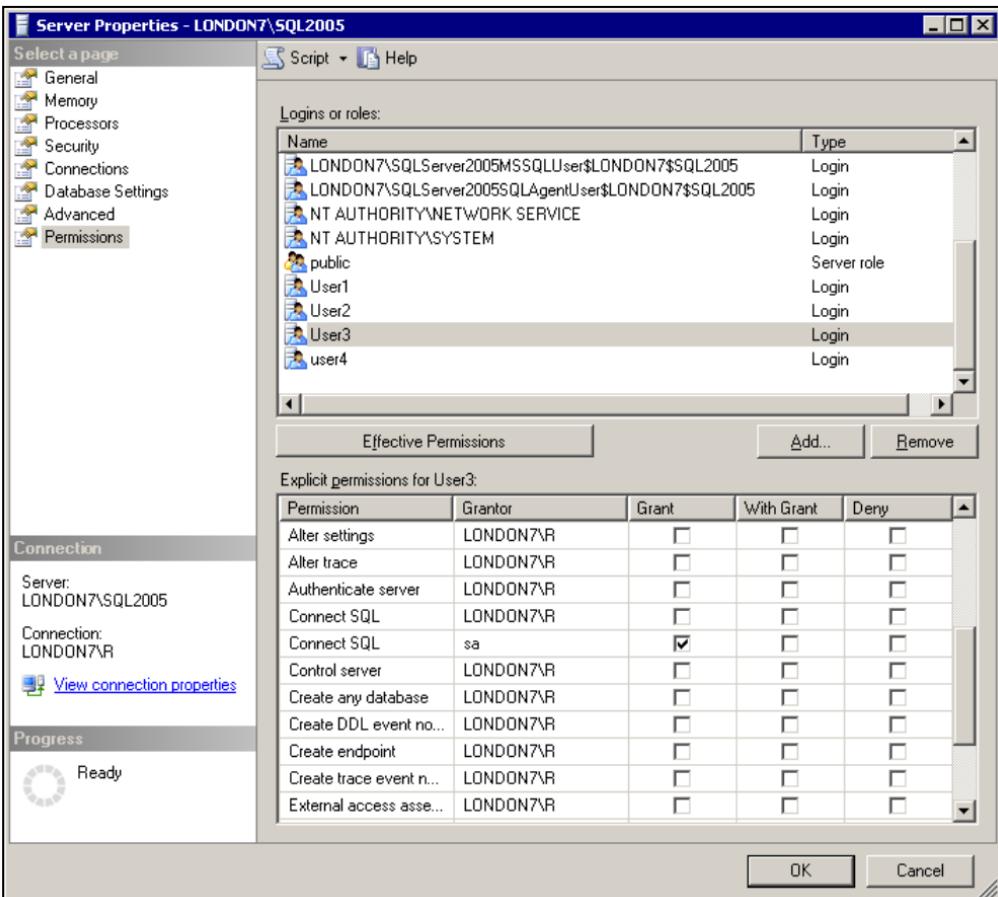


Рис. 5.4. Вкладка **Permissions** свойств SQL Server

Например, вы можете назначить пользователя, который сможет менять пароли для логинов SQL Server сотрудников своего отдела.

5.3. Пользователи баз данных

5.3.1. Что такое пользователь базы данных

После создания логинов следующая задача администратора — спуститься на уровень базы данных и создать объекты пользователей базы данных. Пользователи баз данных — это специальные объекты, которые создаются на уровне базы данных и используются для предоставления разрешений в базе данных (на таблицы, представления, хранимые процедуры). Для пользователей используется термин *database users* (или просто *users*), в отличие от логинов

(*logins*) — учетных записей для подключения к SQL Server. Логины и пользователи баз данных — это совершенно разные объекты.

У слушателей на курсах по SQL Server (особенно у тех, кто раньше работал с Oracle) часто возникает вопрос: а зачем нужен еще один набор учетных записей — пользователи баз данных? Ведь фактически он уже есть: это логины SQL Server. И часто каждому логину SQL Server соответствует только один пользователь базы данных. Неужели нельзя предоставлять права в базе данных непосредственно логинам?

Теоретически такое решение, видимо, вполне возможно. Но на практике разделение логинов и пользователей баз данных обеспечивает большую гибкость. Например, пользователь, который входит от имени одного и того же логина, сможет работать в разных базах данных от имени разных пользователей.

5.3.2. Пользователь и схема

В SQL Server 2000 и в более старых версиях объект пользователя базы данных нес на себе двойную нагрузку: во-первых, он использовался для предоставления разрешений в базе данных, а во-вторых, имя пользователя базы данных использовалось для идентификации объектов. Например, обращение к таблице по полному ее имени могло выглядеть так:

```
SELECT * FROM MyServer.MyDatabase.User1.Table1;
```

Если разработчик использовал в коде приложения просто имя объекта (например, `SELECT * FROM Table1`), то при подключении к серверу из этого приложения от имени другого пользователя могли возникнуть проблемы, поскольку вместо `User1` подставлялось текущее имя пользователя (а если объект с таким полным именем не был обнаружен, то имя специального пользователя `dbo`).

В SQL Server 2005 эти две функции разделены. Для предоставления разрешений в базе данных, как и раньше, используется объект пользователя, а для именования объектов в базе данных используется специальный объект схемы. Запрос с использованием полного формата имени в SQL Server 2005 теперь должен выглядеть так:

```
SELECT * FROM MyServer.MyDatabase.Schema1.Table1;
```

Схема формально определяется как набор объектов в базе данных, объединенных общим пространством имен. Проще всего представить себе схему как некий логический контейнер в базе данных, которому могут принадлежать таблицы, представления, хранимые процедуры, пользовательские функции, ограничения целостности, пользовательские типы данных и другие объекты.

базы данных. Этот контейнер удобно использовать как для именования объектов и их логической группировки, так и для предоставления разрешений. Например, если в базе данных есть набор таблиц с финансовой информацией, удобно поместить их в одну схему и предоставлять пользователям разрешения на эту схему (т. е. на этот набор таблиц).

Пользователю можно назначить схему по умолчанию. В эту схему SQL Server будет по умолчанию помещать объекты, которые создает этот пользователь. Кроме того, искать объекты, к которым обращается пользователь (например, в случае запроса вида `SELECT * FROM Table1`), SQL Server также будет в первую очередь в его схеме по умолчанию.

Применение схемы дает ряд дополнительных преимуществ по сравнению со старым подходом:

- некоторым пользователям можно назначить одну и ту же схему по умолчанию, что может быть удобно при разработке приложений;
- несколько пользователей (через группы Windows или роли баз данных) могут владеть одной и той же схемой. При этом один пользователь может являться владельцем сразу нескольких схем;
- при удалении пользователя из базы данных не придется переименовывать его объекты;
- как уже говорилось, упрощается предоставление разрешений для наборов объектов в базе данных.

Создание схемы производится из контейнера **Имя_базы_данных | Security | Schemas** в Management Studio или при помощи команды `CREATE SCHEMA`.

5.3.3. Создание, изменение и удаление пользователей базы данных

Создать пользователя базы данных можно:

- на графическом экране из контейнера **Имя_базы_данных | Security | Users** в Management Studio;
- при помощи команды `CREATE USER` (хранимая процедура `sp_adduser`, которая использовалась для этой цели в предыдущих версиях SQL Server, оставлена только для обеспечения обратной совместимости). Например, команда на создание пользователя `User1`, которому будет соответствовать логин SQL Server `Login1` со схемой по умолчанию `dbo`, может выглядеть так:

```
CREATE USER User1 FOR LOGIN Login1 WITH DEFAULT_SCHEMA = dbo;
```

При создании пользователя вам нужно будет указать:

- имя пользователя** (User name), к которому применяются те же правила, что и для других объектов SQL Server;
- логин** (SQL Server или Windows), которой будет назначен пользователю этой базы данных. После создания пользователя назначенный ему логин изменять будет нельзя. Можно создать пользователя, которому не будет назначен никакой логин (при помощи переключателя **Without login**). Такому пользователю уже не получится назначить логин. Пользователи этого типа — без логинов — используются только для дополнительной настройки безопасности в Service Broker. Отметим также, что если какой-то логин уже был назначен пользователю, то другому пользователю одновременно назначить его нельзя;
- сертификат** (Certificate name) или **асимметричный ключ** (Key name);
- схему по умолчанию** (Default schema);
- для каких **схем** этот пользователь будет являться **владельцем** (Owned schemas);
- какие **роли базы данных** (Database roles) будут ему назначены.

Обязательных параметров всего два — имя пользователя и логин.

На вкладке **Securables** пользователю можно сразу же предоставить разрешения на объекты базы данных. Речь о предоставлении разрешений пойдет в следующих разделах. Вкладка **Extended Properties** позволяет определить дополнительные пользовательские свойства для данного объекта. Применяются они для тех же целей, что и расширенные свойства баз данных (см. разд. 4.8).

Изменение свойств пользователя и его удаление производится из того же контейнера в Management Studio, что и создание пользователя, а также при помощи команд ALTER USER/DROP USER. Удалить пользователя, владеющего какими-либо объектами в базе данных, нельзя.

5.3.4. Встроенные пользователи базы данных

При создании любой базы данных в ней автоматически создаются четыре специальных пользователя:

- dbo** (от *database owner*) — пользователь-владелец базы данных. Он автоматически создается для того логина, от имени которого была создана эта база данных. Конечно же, как владелец, он получает полные права на свою базу данных (при помощи встроенной роли базы данных `db_owner`). В SQL Server 2005 (в отличие от предыдущих версий) `dbo` может удалить свою базу данных;

- **guest** (гость) — этот специальный пользователь предназначен для предоставления разрешений всем логинам, которым не соответствует ни один пользователь в базе данных. По умолчанию у этого пользователя нет права `login` для базы данных, и, следовательно, работать он не будет. Этот пользователь используется чаще всего для предоставления разрешений логинам на какие-то учебные/тестовые базы данных или на базы данных-справочники, доступные только на чтение;
- **INFORMATION_SCHEMA** — этому пользователю не может соответствовать ни один логин. Его единственное значение — быть владельцем схемы `INFORMATION_SCHEMA`, в которой хранятся представления системной информации для базы данных;
- **sys** — этому специальному пользователю, как и `INFORMATION_SCHEMA`, не могут соответствовать логины. Он является владельцем схемы `sys`, к которой принадлежат системные объекты базы данных.

5.3.5. Роли баз данных

Обычно после создания логина и пользователя базы данных следующее, что нужно сделать, — предоставить пользователю разрешения в базе данных. Один из способов сделать это — воспользоваться ролями базы данных.

Роли базы данных — это специальные объекты, которые используются для упрощения предоставления разрешений в базах данных. В отличие от серверных ролей, которые могут быть только встроенными, роли баз данных могут быть как встроенными, так и пользовательскими. Встроенные роли баз данных обладают предопределенным набором разрешений, а пользовательские роли можно использовать для группировки пользователей при предоставлении разрешений. Обычно пользовательские роли используются только для логинов SQL Server (хотя им можно назначать любых пользователей баз данных, в том числе созданных на основе логинов Windows). Для группировки логинов Windows обычно удобнее и проще использовать группы Windows.

Еще одна специальная разновидность ролей баз данных — роли приложений (*application roles*). Речь о них пойдет в разд. 5.4.

Вначале перечислим встроенные роли баз данных, которые "готовы к использованию" для предоставления разрешений:

- **public** — эта специальная роль предназначена для предоставления разрешений сразу всем пользователям базы данных. Обратите внимание на то, что она выполняет другие функции по сравнению со специальным пользователем `guest`. Пользователь `guest` используется для предоставления разрешений логинам, для которых не создано пользователей в базе данных, а `public` — только существующим пользователям.

Специально сделать пользователя членом этой роли или лишить его членства невозможно. Все пользователи базы данных получают права этой роли автоматически. Интересно, что при добавлении пользователя этой роли никакой ошибки не возникнет, но при следующем открытии ее свойств пользователь исчезнет из списка;

- **db_owner** — этой роли автоматически предоставляются полные права на базу данных. Изначально права этой роли предоставляются только специальному пользователю dbo, а через него — логину, который создал эту базу данных;
- **dbo_accessadmin** — роль для сотрудника, ответственного за пользователей базы данных. Этот сотрудник получит возможность создавать, изменять и удалять объекты пользователей баз данных, а также создавать схемы. Других прав в базе данных у него нет;
- **dbo_securityadmin** — эта роль дополняет роль dbo_accessadmin. Сотрудник с правами этой роли получает возможность назначать разрешения на объекты базы данных и изменять членство во встроенных и пользовательских ролях. Прав на создание и изменение объектов пользователей у этой роли нет;
- **db_backupoperator** — эта роль дает право, как понятно из ее названия, выполнять резервное копирование базы данных;
- **db_ddladmin** — эта роль применяется в редких ситуациях, когда пользователю необходимо дать право создавать, изменять и удалять любые объекты в базе данных, не предоставляя прав на информацию, которая содержится в существующих объектах;
- **db_datareader** и **db_datawriter** — эти встроенные роли дают право просматривать и изменять соответственно (в том числе добавлять и удалять) любую информацию в базе данных. Очень часто пользователю необходимо дать права на чтение и запись информации во всех таблицах базы данных, не предоставляя ему лишних административных разрешений (на создание и удаление объектов, изменение прав и т. п.). Самый простой вариант в этой ситуации (о котором забывают некоторые администраторы) — воспользоваться этими двумя ролями.
- **db_denydatareader** и **db_denydatawriter** — эти роли, как понятно из названий, противоположны ролям db_datareader и db_datawriter. Роль db_denydatareader явно запрещает просматривать какие-либо данные, а db_denydatawriter запрещает внесение изменений. Явный запрет всегда имеет приоритет перед явно предоставленными разрешениями. Обычно эти роли используются в ситуации, когда "разрешаем всем, а потом некоторым запрещаем".

Как уже говорилось ранее, в отличие от серверных ролей, роли баз данных вы можете создавать самостоятельно. Это можно сделать из контекстного меню для контейнера **Имя_базы_данных | Security | Roles | Database Roles** или при помощи команды CREATE ROLE, например:

```
CREATE ROLE DBRole1;
```

Кроме имени роли, при создании можно также указать ее владельца (если это не указано, владельцем роли автоматически станет создавший ее пользователь базы данных). Если вы создаете роль при помощи графического интерфейса, вы можете сразу назначить роль владельцем схемы в базе данных и назначить пользователей, которые будут обладать правами этой роли.

Встроенным ролям назначены предопределенные права, изменить которые невозможно. Предоставление прав пользовательским ролям производится точно так же, как и обычным пользователям базы данных.

5.3.6. Предоставление прав на объекты в базе данных

Если вы решили не применять встроенные роли баз данных, а воспользоваться обычными объектами пользователей (или пользовательскими ролями), то следующее действие — предоставление разрешений на объекты базы данных.

Объектов в базе данных, на которые можно предоставлять разрешения, в SQL Server 2005 стало намного больше по сравнению с предыдущими версиями SQL Server. В добавление к привычным таблицам, представлениям, хранимым процедурам и функциям теперь можно предоставлять разрешения на такие объекты, как роли, пользователи баз данных, сертификаты и асимметричные ключи, наборы схем XML, сборки .NET и т. п. Полный список объектов, на которые можно предоставлять разрешения, был приведен в разд. 5.1.

Однако главное принципиальное изменение, которое произошло в SQL Server 2005, — это появление возможности предоставлять разрешения на уровне схемы. К схеме в SQL Server 2005 могут относиться таблицы, представления, хранимые процедуры, пользовательские функции, ограничения целостности, пользовательские типы данных и другие объекты, на которые приходится предоставлять разрешения чаще всего. Если вы назначите пользователю разрешения на схему, то он получит разрешения на все объекты этой схемы.

Работа с разрешениями производится одинаково для всех объектов базы данных:

- на вкладке **Permissions** свойств этого объекта (эта вкладка, правда, предусмотрена не для всех объектов, для которых можно предоставить разрешения);

- при помощи команд GRANT (предоставить разрешение), DENY (явно запретить что-то делать) и REVOKE (отменить явно предоставленное разрешение или запрет).

Далее перечислены разрешения, которые можно предоставить на уровне схемы. Мы приведем только разрешения для этого объекта, поскольку, во-первых, разрешения в большинстве случаев придется предоставлять именно на уровне схемы, а во-вторых, разрешения схемы включают в себя разрешения, которые можно предоставить другим объектам, например, разрешения SELECT для таблиц и представлений и EXECUTE для хранимых процедур.

- **ALTER** — возможность вносить любые изменения в свойства объекта (за исключением смены владельца). На уровне базы данных можно настроить права ALTER ANY ... — возможность вносить изменения в любые объекты данного типа в базе данных, например, ALTER ANY TABLE.
- **CONTROL** — аналогично разрешению FULL CONTROL в операционной системе. Тот, кому предоставлено такое разрешение, получает полные права как на сам объект, так и на информацию в нем.
- **DELETE** — возможность удалять существующую информацию в таблицах. Применяется к таблицам, представлениям и столбцам.
- **EXECUTE** — право запускать на выполнение. Применяется к хранимым процедурам и функциям.
- **INSERT** — право на вставку новых данных в таблицы. Применяется к таблицам, представлениям и столбцам.
- **REFERENCES** — разрешение, которое можно предоставить для проверки ограничений целостности. Например, пользователь может добавлять данные в таблицу с внешним ключом, а на таблицу с первичным ключом ему нельзя предоставлять права на просмотр. В этом случае на таблицу с первичным ключом ему можно предоставить право REFERENCES — и он сможет производить вставку данных в таблицу с внешним ключом, не получая лишних прав на главную таблицу. Это разрешение можно предоставлять на таблицы, представления, столбцы и функции.
- **SELECT** — право на чтение информации. Предоставляется для таблиц, представлений, столбцов и некоторых типов функций (которые возвращают табличные наборы записей).
- **TAKE OWNERSHIP** — право на принятие на себя владения данным объектом. Владелец автоматически обладает полными правами на свой объект. Такое право можно назначить для любых объектов.
- **UPDATE** — возможность вносить изменения в существующие записи в таблице. Предоставляется на таблицы, представления и столбцы.

VIEW DEFINITION — право на просмотр определения для данного объекта.

Предусмотрено для таблиц, представлений, процедур и функций.

Для каждого разрешения мы можем установить три значения:

GRANT — разрешение предоставлено явно;

WITH GRANT — разрешение не только предоставлено данному пользователю, но он также получил право предоставлять это разрешение другим пользователям. Можно предоставлять, конечно, только вместе с **GRANT**;

DENY — явный запрет на выполнение действия, определенного данным разрешением. Как в любых системах безопасности, явный запрет имеет приоритет перед явно предоставленными разрешениями.

Из кода Transact-SQL разрешения можно предоставлять командами **GRANT**, **DENY** и **REVOKE**. Например, чтобы предоставить пользователю *User1* возможность просматривать данные в таблице *Table1* в схеме *dbo*, можно воспользоваться командой:

```
GRANT SELECT ON dbo.Table1 TO User1;
```

Лишить его ранее предоставленного права можно при помощи команды:

```
REVOKE SELECT ON dbo.Table1 TO User1;
```

Отметим еще несколько моментов, связанных с предоставлением разрешений:

в большинстве реальных задач используются десятки и даже сотни таблиц и других объектов базы данных. Предоставлять каждому пользователю разрешения на каждый из этих объектов очень неудобно. Если есть возможность, постарайтесь использовать разрешения на уровне схемы или всей базы данных. Часто упростить предоставление разрешений могут и встроенные роли баз данных (*db_datareader* и *db_datawriter*);

существует общий принцип: не стоит обращаться из клиентского приложения к таблицам базы данных напрямую. Для изменения данных лучше использовать хранимые процедуры, а для запросов на чтение — хранимые процедуры или представления (при этом хранимые процедуры предпочтительнее). Причина проста: вы избежите внесения изменений в клиентское приложение, если потребовалось поменять структуру вашей базы данных (например, какая-то таблица поделилась на текущую и архивную или добавился новый столбец). Это следует помнить и при предоставлении разрешений. Отметим также, что при помощи хранимых процедур можно очень просто реализовать дополнительные проверки в добавление к обычным разрешениям;

SQL Server позволяет настраивать разрешения на уровне отдельных столбцов. На практике лучше не пользоваться такими разрешениями из-за

падения производительности и усложнения системы разрешений. Если пользователю можно видеть не все столбцы в таблице (например, ему не нужны домашние телефоны сотрудников), то правильнее будет создать представление или хранимую процедуру, которые будут отфильтровывать ненужные столбцы;

- в SQL Server 2005 появилась замечательная кнопка **Effective Permissions** (она расположена на вкладке **Permissions**). Эта кнопка позволяет посмотреть итоговые разрешения для пользователя или роли базы данных (поскольку разрешения от разных ролей базы данных, назначенных этому пользователю, суммируются).

5.4. Роли приложений

Альтернативный метод предоставления разрешений в базе данных SQL Server 2005 — воспользоваться ролью приложения (*application role*). Отличием этого метода является то, что разрешения предоставляются не пользователю, а приложению, из которого пользователь подключается к базе данных.

Применение этого способа выглядит так:

- пользователь от имени любого логина (к которому должен соответствовать какой-нибудь объект пользователя в базе данных с правом CONNECT) подключается к серверу и делает нужную базу данных текущей;
- затем пользователь выполняет хранимую процедуру `sp_setapprole`, чтобы активизировать указанную им роль приложения. При этом в базе данных он получает права этой роли приложения (и теряет свои текущие права);
- после того, как необходимость в правах роли приложения закончилась, пользователь может опять переключиться на свою исходную учетную запись (и получить соответствующие ей права). Эта новая возможность SQL Server 2005 обеспечивается при помощи хранимой процедуры `sp_unsetapprole`.

Чаще всего такие роли используются в тиражируемых приложениях, когда задача вполне может обслуживаться не очень квалифицированным администратором. Преимущества такого подхода очевидны:

- гарантируется, что приложение будет обладать необходимыми правами в базе данных;
- администратору достаточно создать любой логин и любую учетную запись в базе данных, без необходимости настраивать требуемые разрешения.

Однако у ролей приложений имеются также и недостатки. В первую очередь, они связаны с тем, что пароль роли приложения передается по сети открытым

текстом (предусмотрена возможность использования функции ODBC Encrypt, но реальной защиты она не обеспечивает). Как правило, этот пароль можно извлечь и из двоичного кода клиентского приложения.

Стоит ли использовать роли приложений? Все зависит от ситуации. Если приложение будет работать только на одном предприятии и обслуживаться квалифицированным администратором, то проще использовать обычные средства предоставления разрешений. Если же приложение будет устанавливаться на десятках и сотнях предприятий и за квалификацию администратора вы поручиться не можете, то применение роли приложения может быть очень удачным решением. Часто в качестве альтернативы разработчики требуют предоставлять всем пользователям права владельца базы данных, а иногда и системного администратора сервера, что, конечно, является намного большим злом.

Создание роли приложения производится точно так же, как и создание обычных ролей — из контейнера **Имя_базы_данных | Security | Roles | Application Roles**. Главное отличие заключается в том, что назначить пользователей этой роли невозможно, зато ей нужно будет указать пароль, который будет использоваться при активизации. Команда Transact-SQL на создание роли приложения может выглядеть так:

```
CREATE APPLICATION ROLE AppRole1 WITH PASSWORD = 'password';
```

Предоставление прав этой роли ничем не отличается от предоставления прав обычным пользователям или ролям базы данных.

При подключении к базе данных клиентская программа, которая использует роль приложения, должна выполнить код, аналогичный следующему:

```
-- Объявляем переменную appCookie.  
-- Она потребуется для возвращения к исходным правам  
DECLARE @appCookie varbinary(8000);  
-- Активизируем роль приложения  
EXEC sp_setapprole 'AppRole1', 'password', 'none', TRUE,  
    @appCookie OUTPUT;  
-- Проверяем  
SELECT USER_NAME();  
-- Возвращаемся к исходным правам  
EXEC sp_unsetapprole @appCookie;  
-- Проверяем еще раз  
SELECT USER_NAME();
```

При желании, конечно, можно и не возвращаться к исходным правам. Но, например, если потребуется во время работы имени роли приложения обратиться к другой базе данных, вы не сможете получить в ней других прав, кроме прав, предоставленных для специального пользователя guest.

5.5. Изменение контекста выполнения. Выражение EXECUTE AS

Новая возможность SQL Server 2005 — изменение контекста выполнения команд Transact-SQL. Например, вы подключились от имени пользователя, у которого нет прав на определенную таблицу. Но в то же время вы обладаете правом выступать от имени другого пользователя, у которого права на эту таблицу есть (право действовать от имени другого пользователя называется IMPERSONATE). Вместо того чтобы разрывать соединение с SQL Server и входить заново от имени второго пользователя, вы можете просто "превратиться" во второго пользователя при помощи выражения EXECUTE AS и выполнить необходимые запросы к таблице от имени этого пользователя и с его правами, а потом, при желании, вернуться обратно.

В принципе, возможность менять контекст выполнения была и в предыдущих версиях SQL Server. Она представлялась командой SETUSER (для обратной совместимости эта команда сохранена и в SQL Server 2005). Однако возможностей у EXECUTE AS намного больше:

- команда SETUSER позволяла менять контекст выполнения только пользователям с правами системного администратора (на уровне сервера) или dbo (на уровне базы данных). Поскольку прав у этих пользователей и так всегда достаточно, то обычно команда SETUSER применялась только для проверки, как та или иная команда будет выполняться от имени определенного пользователя. Выражение EXECUTE AS могут использовать любые пользователи, поэтому основное назначение этого выражения — "подключение" пользователю дополнительных прав на время выполнения определенных команд;
- выражение EXECUTE AS, в отличие от команды SETUSER, позволяет возвращаться "обратно" — к исходной учетной записи, которая использовалась до смены контекста выполнения. Для этого используется команда REVERT. При этом с помощью EXECUTE AS вы вполне можете поменять контекст выполнения несколько раз. Каждая выполненная команда REVERT вернет контекст выполнения на один "уровень" назад.

Работа с выражением EXECUTE AS очень проста. Предположим, что у вас есть пользователь базы данных User1, у которого нет прав на таблицу Table2, и пользователь User2, у которого такие права есть. До начала применения команды EXECUTE AS необходимо предоставить пользователю User1 право IMPERSONATE на объект пользователя User2, чтобы он получил возможность выполнять команды от имени этого пользователя.

На графическом экране это можно сделать так:

- откройте свойства пользователя User1 в Management Studio и перейдите на вкладку **Securables**;
- нажмите кнопку **Add** под списком **Securables**, в открывшемся окне установите переключатель в положение **All objects of the types** (Все объекты типа) и нажмите кнопку **OK**;
- в списке типов объектов выберите **Users** и нажмите кнопку **OK**;
- в загруженном списке пользователей выберите пользователя User2 и установите для него флагок в столбце **Grant** напротив разрешения **IMPERSONATE** (рис. 5.5).

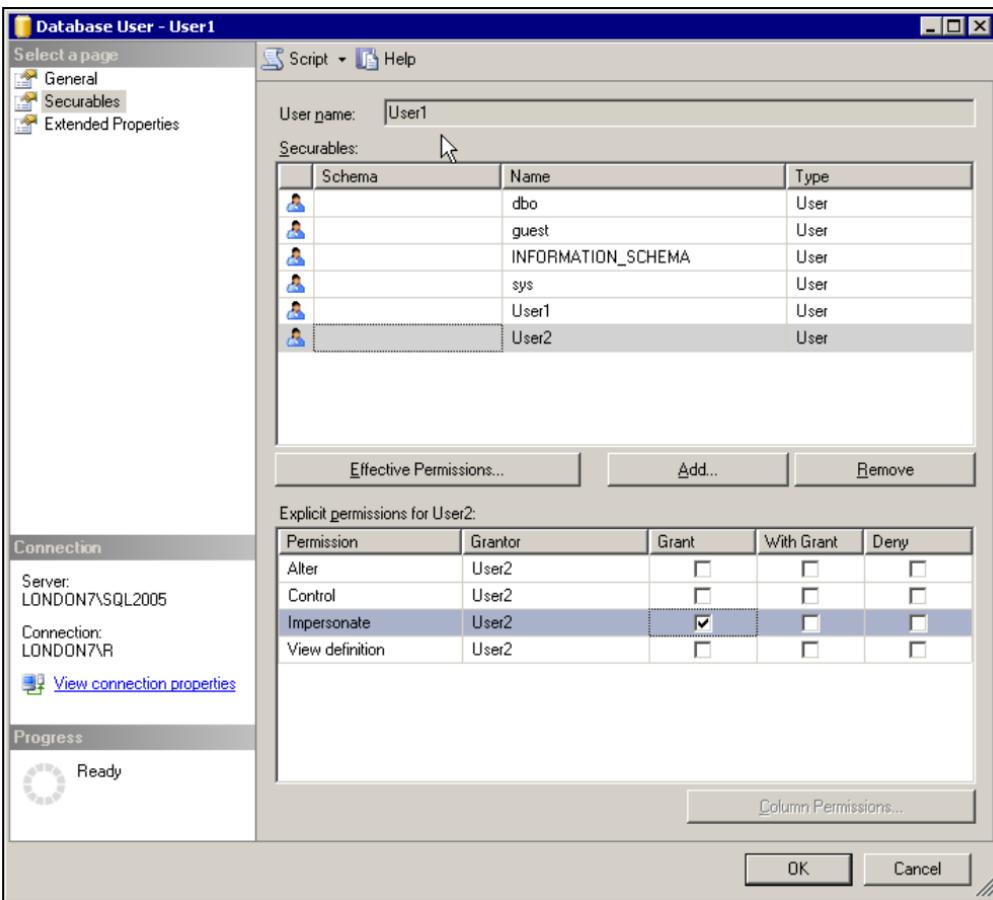


Рис. 5.5. Предоставление права IMPERSONATE

Соответствующая команда Transact-SQL может выглядеть так:

```
GRANT IMPERSONATE ON USER::User2 TO User1;
```

После этого можно использовать возможности выражения EXECUTE AS, например, так:

```
-- Переключаемся в контекст выполнения User2  
EXECUTE AS USER = 'User2';  
-- Обращаемся к таблице Table2  
SELECT * FROM dbo.Table2;  
-- Возвращаемся в контекст выполнения User1  
REVERT;
```

Чаще всего выражение EXECUTE AS используется в определениях программных модулей Transact-SQL (процедур или функций). Это позволяет менять контекст выполнения только на время работы данной хранимой процедуры или функции, например:

```
CREATE PROCEDURE SelectFromTable2  
    WITH EXECUTE AS 'User2'  
    AS SELECT * FROM dbo.Table2;
```

Если у пользователя есть право EXECUTE на хранимую процедуру SelectFromTable2, он может выполнять ее от имени пользователя User2.

В такой ситуации проще бы было сделать так, чтобы владельцем хранимой процедуры и таблицы был один и тот же пользователь, тогда вам не нужно будет прибегать к таким приемам, но это возможно не всегда.

Если вы используете выражение EXECUTE AS в хранимых процедурах и функциях, ему можно передать несколько зарезервированных значений, которые подставляются вместо имени пользователя:

- EXECUTE AS CALLER** — все команды в хранимой процедуре или функции будут выполнены с правами текущего пользователя (при этом неважно, кто является владельцем этой хранимой процедуры или функции);
- EXECUTE AS SELF** — все команды в хранимой процедуре или функции будут выполняться от имени пользователя, который создает или изменяет данную процедуру или функцию. Это значение используется в редких случаях, когда клиентскому приложению требуется создавать хранимые процедуры или функции и запускать их от имени разных пользователей;
- EXECUTE AS OWNER** — все команды хранимой процедуры или функции будут выполняться от имени ее текущего владельца. Это значение тоже используется редко: обычно только в тех ситуациях, когда владелец хранимой процедуры или функции может часто меняться.

Выражение EXECUTE AS можно использовать не только для изменения контекста выполнения на уровне базы данных, но и на уровне сервера (т. е. логинов). Для этого ключевое слово USER меняется на LOGIN, например:

```
EXECUTE AS LOGIN = 'Login2';
```

5.6. Применение сертификатов и шифрование данных в SQL Server 2005

5.6.1. Основы применения сертификатов и шифрования данных

Прежде чем начать разговор о возможностях шифрования данных и сертификатов в SQL Server 2005, необходимо рассказать о некоторых концепциях и терминах, связанных с шифрованием данных.

Вначале остановимся на основных концепциях шифрования.

В настоящее время применяются два принципиально разных способа шифрования: **шифрование с использованием симметричного ключа** и **шифрование с использованием асимметричного ключа** (точнее, пары ключей). При шифровании с использованием симметричного ключа ключ, который применяется для шифрования данных, используется и для расшифровки. Это традиционный и очень хорошо отработанный метод шифрования данных. Он требует минимального количества ресурсов компьютера и очень распространен. В качестве примеров использования симметричных ключей можно привести шифрование при помощи паролей архивов ZIP, файлов Office и т. п.

Второй способ шифрования — применение асимметричных ключей, т. е. пары **общий** (*public*) и **частный** (*private*) ключи. При использовании этого метода данных шифруются первым ключом из пары (общим), а для расшифровки нужен второй ключ (частный). При этом если кто-то получит доступ к зашифрованным данным и общему ключу, который использовался для шифрования, расшифровать данные он не сможет: расшифровку можно произвести только при помощи частного ключа. Проиллюстрируем механизм работы этого метода на простом примере.

Предположим, что пользователю **А** необходимо передать информацию в зашифрованном виде пользователю **Б**. Для этого пользователь **Б** (а не **А**!) генерирует у себя пару общий/частный ключи и передает общий ключ пользователю **А**. Тот шифрует полученным общим ключом данные и передает их в зашифрованном виде пользователю **Б**. А он уже расшифровывает их при помощи второго ключа из пары (частного), который находился у него на компьютере и никуда не передавался.

У шифрования с использованием асимметричных ключей есть очевидные преимущества. При применении симметричных ключей для обмена информацией пользователям необходимо вначале обменяться этим симметричным ключом, который в процессе передачи могут перехватить. Если же вы используете асимметричные ключи, то это позволяет вам надежно защищать данные при передаче по незащищенным каналам связи.

Надо отметить, что у метода с применением асимметричных ключей есть не только достоинства, но и недостатки. Главный из них заключается в том, что шифрование с использованием этого метода требует очень больших ресурсов центрального процессора. Поэтому на практике чаще всего используется комбинированный вариант. Пример этого варианта может выглядеть так.

Предположим, что пользователю **A** необходимо вновь передать информацию в зашифрованном виде пользователю **B**. Пользователь **B**, как и в прошлом случае, генерирует пару общий/частный ключи и передает общий ключ пользователю **A**. Пользователь **A**, вместо того чтобы напрямую шифровать данные при помощи этого ключа, генерирует еще один ключ — симметричный. После этого пользователь **A** шифрует общим ключом сгенерированный им симметричный ключ и передает его пользователю **B**. После того как пользователь **A** получает подтверждение, что симметричный ключ получен и расшифрован пользователем **B**, пользователь **A** начинает передачу данных, зашифрованных симметричным ключом.

И еще один момент, связанный с использованием асимметричных ключей. Предположим, что в процесс обмена информации между пользователем **A** и пользователем **B** вмешался зловредный хакер **B**. Он прислал общий ключ от имени пользователя **B**, а затем расшифровал всю секретную информацию, которую передал ему обознавшийся пользователь **A**. Чтобы такого не происходило, присланные общие ключи принято проверять в специальном месте, которое называется центр сертификации (*Certificate Authority*). Обычно центр сертификации и генерирует пары общий/частный ключи. В нашем примере пользователь **A**, получив общий ключ, должен обратиться в центр сертификации, который сгенерировал этот ключ, чтобы убедиться, что данный ключ действительно выдан пользователю **B**, что он не устарел, не отозван и т. п. Конечно же, пользователь **A** должен доверять этому центру сертификации.

Список центров сертификации, которые заплатили деньги компании Microsoft и которым по умолчанию доверяет ваш компьютер, можно просмотреть в Internet Explorer (выбрать пункт меню **Сервис | Свойства обозревателя**, в открывшемся окне перейти на вкладку **Содержание**, нажать кнопку **Сертификаты** и в одноименном окне перейти на вкладку **Доверенные корневые центры сертификации**).

Еще одна концепция шифрования данных — **сертификат**. Сертификат — это контейнер для хранения общего ключа. В сертификат помещается информа-

ция о версии (в соответствии со стандартом International Telecommunication Union (Международного союза телекоммуникаций) X.509), серийном номере, кому выдан данный сертификат, для каких целей, сколько времени он будет действовать и т. п. Эта информация математически связывается с открытым ключом (при помощи цифрового отпечатка — *thumbprint*), так что исказить ее будет нельзя.

В SQL Server 2005 шифрование используется в разных ситуациях.

Первая ситуация — **шифрование данных, передаваемых между клиентом и сервером** (или, например, между двумя серверами при зеркальном отображении баз данных или репликации). Для этого используется технология SSL (Secure Socket Layer), которая применяется также для шифрования трафика между Web-браузером и Web-сервером. Эта технология была и в SQL Server 2000, но в SQL Server 2005 появились дополнительные возможности, а настройка в целом стала более удобной. Подробно про шифрование сетевого трафика при подключении к SQL Server 2005 будет рассказано в разд. 5.6.2.

Вторая ситуация — **шифрование данных в таблицах баз данных**. Это совершенно новая и очень важная возможность SQL Server 2005. Про нее будет рассказываться в разд. 5.6.3.

Сертификаты можно использовать и для других целей, например, для создания цифровых подписей для программных модулей SQL Server 2005 (например, для хранимых процедур). Защита при помощи сертификата гарантирует защиту целостности кода хранимой процедуры или функции. Если в этот код будут внесены несанкционированные изменения, то такая хранимая процедура просто не будет запускаться на выполнение.

5.6.2. Защита сетевого трафика SQL Server 2005

Одна из больших проблем безопасности при работе с SQL Server любых версий, в том числе и 2005, заключается в том, что данные запросов пользователей и ответов на них сервера возвращаются в абсолютно открытом виде формата пакета TDS (Tabular Data Stream — поток табличных данных). Это означает, что, перехватывая пакеты в локальной сети, можно перехватить информацию, которую получают пользователи с SQL Server (рис. 5.6). Пароли логинов SQL Server передаются изначально в защищенном виде, но если пользователь решит поменять свой пароль командой `ALTER USER`, то такой пароль будет передан по сети открытым текстом. Отметим также, что в Интернете можно найти множество программ, которые умеют собирать данные и парольные хэши SQL Server и расшифровывать их. Пример одной из таких программ — Cain&Abel.

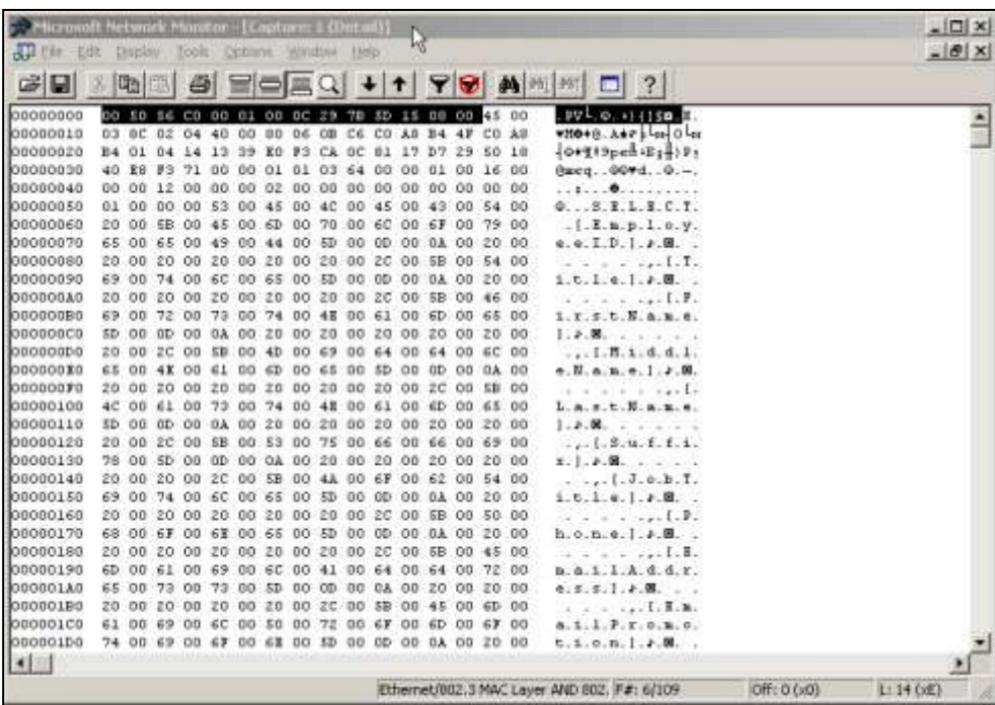


Рис. 5.6. Просмотр перехваченных по сети пакетов
(таким образом можно прочитать текст запросов и возвращаемые результаты)

Некоторые слушатели на курсах говорили, что они не беспокоятся о возможности перехвата данных SQL Server по сети, поскольку в сети их предприятий используются свитчи. Теоретически свитчи должны не позволять одним компьютерам просматривать трафик других компьютеров, который их не касается. Однако на практике свитчи не обеспечивают практически никакой реальной защиты из-за особенностей протокола ARP, который входит в стек протоколов TCP/IP. Доступ к трафику любого компьютера в своем сегменте сети можно получить, например, при помощи программы AntiSwitch (ее также можно скачать из Интернета).

Если вы не хотите неприятностей, связанных с разглашением информации баз данных SQL Server, сетевой трафик необходимо защищать. Стоит отметить, что многое здесь зависит от разработчиков. Вполне может оказаться так, что используемые ими программные интерфейсы не дают возможности использовать встроенные средства SQL Server 2005 для защиты сетевого трафика. В этом случае можно порекомендовать использовать прозрачные для приложений средства IPSec или просто выделить самых важных пользователей и SQL Server в отдельный сетевой сегмент, отгороженный от остальной сети маршрутизатором. Рекомендуется также использовать средства актив-

ной защиты, регулярно выявляя работающие снiffeры в вашей сети специальными программами (например, ProDetect и PromiScan). Однако на многих предприятиях за это ответственен администратор сети, власти над которым администратор SQL Server не имеет.

В этом разделе мы рассмотрим средства защиты только для стандартных сетевых библиотек SQL Server 2005. Эти средства позволяют защитить большинство приложений, в том числе Excel и Access. Не поддерживаются только клиенты, использующие сетевую библиотеку DBLibrary, которая была унаследована от SQL Server 6.5, и клиенты, использующие набор драйверов MDAC версии до 2.53 включительно. Поддерживаются два уровня шифрования — 40 и 128 бит, при этом выбирается максимально возможный уровень, поддерживаемый операционной системой клиента и сервера.

Отметим также, что сетевая библиотека Multiprotocol, которую можно было использовать для шифрования данных в предыдущих версиях SQL Server, в SQL Server 2005 не поддерживается. Единственная возможность зашифровать данные в SQL Server 2005 — воспользоваться технологией SSL. Эта технология доступна для любых сетевых библиотек.

В SSL для защиты передаваемых между клиентом и сервером данных обязательно используются сертификаты. В SQL Server 2005 можно использовать два типа сертификатов:

- **сертификат, который автоматически генерируется и подписывается самими SQL Server 2005** (в терминологии, которая используется в документации — *self-signed* (самоподписанный)). Такой сертификат будет автоматически создаваться и использоваться в ситуации, когда шифрование включено, но какой именно сертификат использовать, явно не указано. Кроме того, такой сертификат всегда используется в автоматическом режиме для шифрования пароля при подключении пользователей от имени логинов SQL Server, даже если вы не настраивали никакого шифрования;
- **сертификат от внешнего центра сертификации** (*Certificate Authority*). Чтобы можно было нормально работать с таким сертификатом, центру сертификации должны доверять и сервер SQL Server 2005, и клиентские компьютеры (хотя для клиентских компьютеров можно установить параметр **Доверять любым серверным сертификатам**).

Настроить применение самоподписанного сертификата для любых клиентских подключений очень просто. Для этого достаточно запустить SQL Server Configuration Manager, раскрыть контейнер **SQL Server 2005 Network Configuration** и щелкнуть правой кнопкой мыши по узлу **Protocols for имя_сервера**, например, **Protocols for SQL2005**. Затем в контекстном меню нужно открыть свойства этого узла и на вкладке **Flags** (Флаги) для параметра

ForceEncryption (Принудительное шифрование) установить значение **Yes**. Если на соседней вкладке, которая называется **Certificate** (Сертификат), не будет выбран никакой сертификат, SQL Server автоматически перейдет в режим использования самоподписанного сертификата (при этом придется перезапустить службу SQL Server). Клиенты, использующие подключения по OLE DB и ODBC с последними версиями MDAC (после 2.53) продолжат работу без каких-либо проблем. Шифрование будет производиться автоматически (в этом можно убедиться, перехватив пакеты снiffeром). Если вы используете не обычный драйвер ODBC, а драйвер ODBC для SQL Native Client, то он выдаст сообщение, что применено шифрование без проверки серверного сертификата (рис. 5.7).

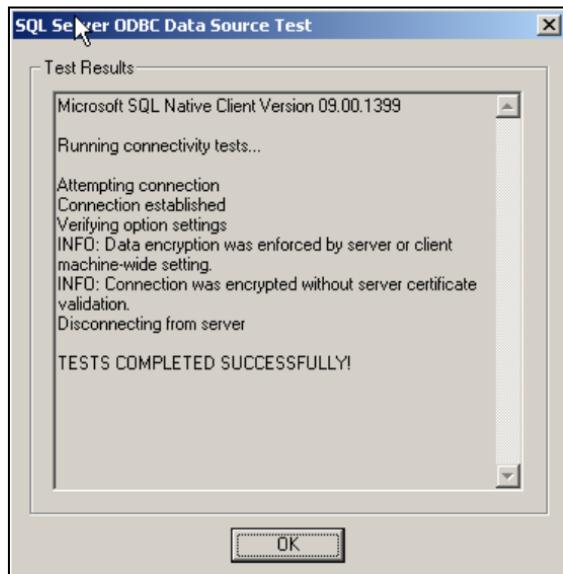


Рис. 5.7. Подключение по ODBC
с применением самоподписанного серверного сертификата

Такой режим очень удобен и вполне может применяться в большинстве ситуаций. Однако Microsoft предупреждает, что при использовании самоподписанного сертификата вполне возможны хакерские атаки типа "человек в середине" (*man-in-the-middle*). При этой атаке хакер представляется сервером SQL Server, принимает клиентские пакеты от имени сервера и перенаправляет их на сервер, возвращая затем полученные с сервера данные пользователям. В результате у хакера останется копия всех данных, которыми обменивались клиенты и сервер. Такая атака становится возможной из-за того, что сертификат, передаваемый сервером (или хакером) клиенту, никем не проверяется.

Кроме того, если на клиенте настроено "требовать принудительного шифрования при подключении к SQL Server", то самоподписанный сертификат использовать не может.

Поэтому в такой ситуации вам придется использовать для шифрования данных сертификат не самоподписанный, а выданный центром сертификации, которому доверяют и клиент, и сервер. Вряд ли вы захотите платить деньги за сертификаты платным центрам сертификации, которым Windows доверяет автоматически (отметим также, что для проверки выданных ими сертификатов потребуется соединение с Интернетом как для клиента, так и для сервера). Поэтому вам нужно будет создать свой собственный центр сертификации. Все, что для этого нужно, поставляется вместе с дистрибутивами Windows 2000 Server и Windows Server 2003.

Опишем последовательность применения сертификата для защиты трафика SQL Server 2005.

Первое, с чего нужно начать, — установка центра сертификации. Для этого надо установить дополнительный необязательный компонент Windows, который называется Certificate Services (Службы сертификации).

Его установка производится точно так же, как и установка других компонентов Windows — при помощи консоли **Установка и удаление программ** в **Панели управления**. После окончания копирования файлов запустится мастер настройки центра сертификации. В нем нужно будет выбрать несколько важных параметров. Первый из них — тип центра сертификации. В вашем распоряжении четыре типа:

- корневой центр сертификации предприятия.** Такой тип центра сертификации предприятия будет доступен только в том случае, если компьютер, на котором производится установка служб сертификации, входит в домен. Этот центр сертификации предназначен для выдачи сертификатов, используемых в домене Active Directory в Windows 2000/2003. Для работы с SQL Server 2005 он не подходит;
- подчиненный центр сертификации предприятия.** Точно так же генерируются сертификаты для использования в Active Directory. Обычно такие подчиненные центры используются в филиалах и подразделениях предприятий;
- изолированный корневой центр сертификации.** Этот центр сертификации может создавать сертификаты для защиты Web-серверов, аутентификации клиентов при подключении к Web-серверам, цифровой подписи для драйверов и т. п. Именно этот тип центра сертификации нужен для защиты трафика SQL Server 2005;
- изолированный подчиненный центр сертификации.** Такой центр сертификации генерирует те же типы сертификатов, что и предыдущий, но

требует обязательного наличия изолированного корневого центра сертификации. Обычно используется в филиалах и подразделениях.

В подавляющем большинстве случаев для работы с SQL Server 2005 вам потребуется изолированный корневой центр сертификации.

На экране **Сведения о центре сертификации** вам потребуется ввести имя и суффикс для центра сертификации. К вводу этой информации нужно отнестись очень внимательно: именно по указанному здесь имени другие компьютеры в вашей сети будут разыскивать центр сертификации для проверки сертификатов. Имя должно соответствовать имени DNS (*hostname*) с указанием основного DNS-суффикса, например, LONDON2.nwtraders.msft. Для всех остальных параметров можно оставить значения, предлагаемые по умолчанию.

Следующее, что нужно сделать после окончания установки, — создать сертификат, который будет использоваться SQL Server 2005. Заказ сертификата и его установка производится при помощи Web-интерфейса. Нужно обратиться из Internet Explorer на тот компьютер, на котором работают службы сертификации, например, <http://london2.nwtraders.msft/certsrv>. Желательно сделать это с компьютера, на котором работает SQL Server 2005, т. к. потом вам потребуется установить полученный сертификат на этот компьютер. Причем это обращение нужно производить, войдя на сервер локально от имени той учетной записи, от которой работает SQL Server 2005.

На первой странице Web-интерфейса служб сертификации нужно выбрать пункт **Запрос сертификата**, на следующей странице щелкнуть по ссылке **Расширенный запрос сертификата**. А на следующем экране нужно перейти по ссылке **Создать и выдать запрос к этому ЦС**.

Самый важный экран для настройки свойств создаваемого сертификата — **Расширенный запрос сертификата** (рис. 5.8).

Нужно обратить внимание на следующие моменты:

- в поле **Имя** нужно привести имя компьютера в формате FQDN (Fully qualified domain name — полностью квалифицированное доменное имя), например, LONDON2.NWTRADERS.MSFT вместо LONDON2;
- в списке **Нужный тип сертификата** обязательно выберите тип **Сертификат проверки подлинности сервера**.

Если вы нарушите любое из этих требований, то SQL Server 2005 просто "не увидит" этот сертификат и не даст его назначить.

После того как сертификат будет запрошен, ваша следующая задача — утвердить выдачу сертификата. Для этого нужно будет на компьютере, на котором работают службы сертификации, в меню **Пуск | Программы | Администрирование | Сертификация** выбрать команду **Выдать сертификат**.

стрирование запустить консоль Центр сертификации, раскрыть узел **Запросы в ожидании**, щелкнуть правой кнопкой по созданному вами запросу о выдаче сертификата и в контекстном меню выбрать **Все задачи | Выдать** (рис. 5.9).

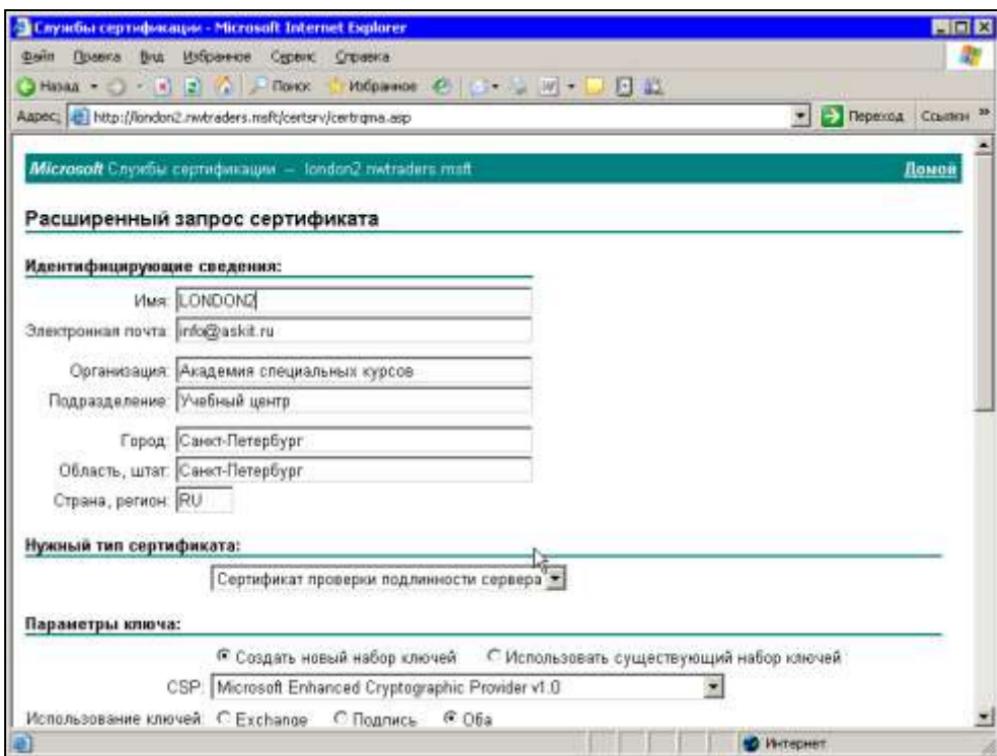


Рис. 5.8. Экран для заполнения свойств создаваемого сертификата

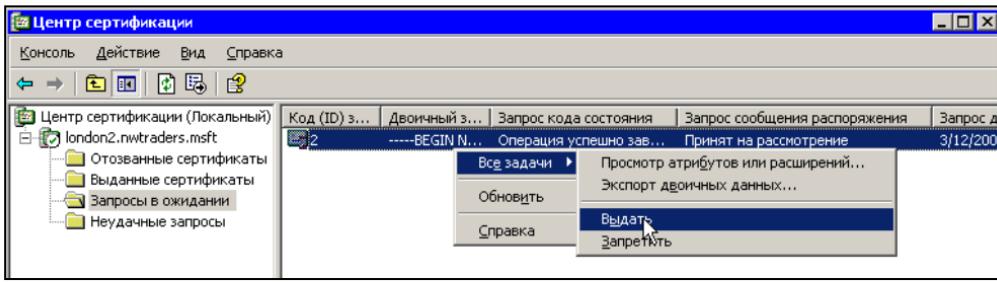


Рис. 5.9. Выдача сертификата

Следующее действие — получить утвержденный сертификат и установить его на компьютер, на котором работает SQL Server 2005. Для этого нужно

с того компьютера, с которого вы заказывали сертификат, войдя локально от имени той же учетной записи, опять обратиться к Web-интерфейсу служб сертификации (<http://london2.nwtraders.msft/certsrv>). Затем на первой странице нужно перейти по ссылке **Просмотр состояния** ожидаемого запроса сертификата, выбрать ваш сертификат и установить его при помощи кнопки **Установить сертификат**.

Просмотреть информацию об установленных на компьютере сертификатах можно при помощи Internet Explorer. Для этого в меню **Сервис** нужно выбрать пункт **Свойства обозревателя**, перейти на вкладку **Содержание** и нажать кнопку **Сертификаты**. Нужный сертификат будет находиться на вкладке **Личные**.

Следующее, что нужно сделать, — назначить созданный вами сертификат SQL Server 2005. Для этого нужно открыть SQL Server Configuration Manager, раскрыть узел **SQL Server 2005 Network Configuration** и щелкнуть правой кнопкой мыши по контейнеру **Protocols for имя_сервера**. Затем нужно открыть свойства этого контейнера, на вкладке **General** настроить для параметра **ForceEncryption** значение **Yes**, а на вкладке **Certificate** выбрать ваш сертификат (рис. 5.10). Чтобы изменения вступили в силу, потребуется перезапустить службу SQL Server.

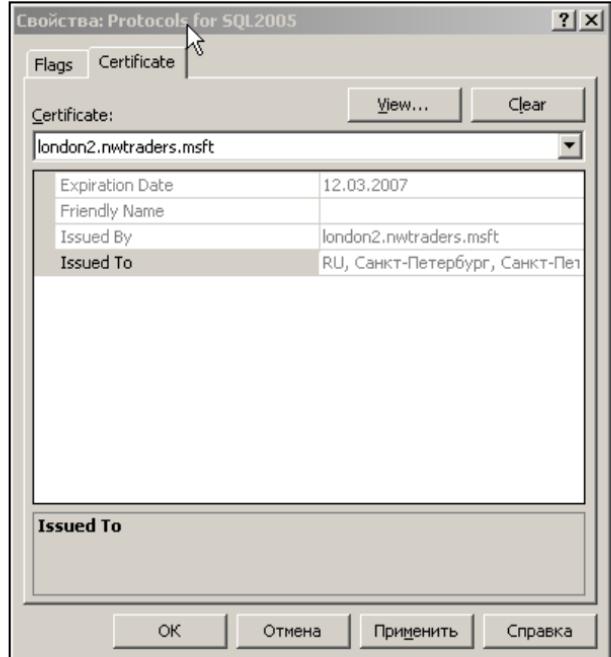


Рис. 5.10. Назначение сертификата серверу SQL Server 2005

Последнее, что вам осталось сделать, — это объяснить компьютеру, на котором работает сервер SQL Server, и всем клиентам, что им следует доверять созданному вами центру сертификации, который выдал сертификат. Это можно сделать двумя способами: вручную на каждом компьютере или при помощи групповой политики (если на вашем предприятии используется домен Active Directory).

Чтобы произвести эту операцию вручную, нужно экспортировать сертификат центра сертификации в файл. Это можно сделать при помощи консоли **Центр сертификации**, которая использовалась нами для утверждения выдачи сертификата. Нужно из контекстного меню открыть свойства самого центра сертификации, на вкладке **Общие** выбрать сертификат центра сертификации, нажать кнопку **Просмотр сертификата**, перейти на вкладку **Состав** и нажать кнопку **Копировать в файл**. После этого на том компьютере, который должен доверять вашему центру сертификации, нужно открыть Internet Explorer, в меню **Сервис** выбрать **Свойства обозревателя**, перейти на вкладку **Содержание**, нажать кнопку **Сертификаты**, затем перейти на вкладку **Доверенные корневые центры сертификации** и нажать кнопку **Импорт**, чтобы импортировать созданный вами файл.

При использовании групповых политик необходимый параметр можно найти в контейнере редактора объектов групповых политик **Конфигурация пользователя | Конфигурация Windows | Параметры безопасности | Политики открытого ключа | Доверительные отношения в предприятии**. Далее нужно щелкнуть правой кнопкой мыши по этому контейнеру и импортировать созданный вами сертификат.

Настройки могут показаться достаточно сложными, но стоит учесть, что достаточно произвести их один раз, чтобы надежно защитить трафик клиентов SQL Server 2005 в вашей сети.

5.6.3. Принципы шифрования информации в базах данных SQL Server 2005

Еще одна большая проблема, связанная с безопасностью SQL Server, заключается в том, что информация хранится в базах данных в незащищенном виде. Фактически единственный рубеж защиты — это разрешения самого SQL Server. Как только они перестают действовать, становится возможным получение доступа к любой информации в базе данных. Приведем несколько обычных ситуаций, когда может произойти нарушение системы безопасности SQL Server:

- администратор сети недосмотрел за безопасностью учетных записей в домене. В результате злоумышленник смог подключиться к SQL Server при

помощи учетной записи, от имени которой работают службы SQL Server (эта учетная запись обладает правами встроенной серверной роли **SYSADMIN**);

- злоумышленник смог получить физический доступ к компьютеру, на котором работает SQL Server, и скопировал файлы базы данных (это можно сделать разными способами, например, загрузившись с компакт-диска или подключив RAID-массив сервера к другому компьютеру). Затем злоумышленник сможет присоединить эти файлы к другому серверу SQL Server, на котором он обладает правами системного администратора;
- в чужие руки попал носитель с резервной копией базы данных. Злоумышленник также сможет восстановить эту резервную копию на другом сервере, на котором он обладает правами системного администратора, и получить таким образом доступ ко всем данным.

Для того чтобы обезопасить себя от подобных ситуаций, следует использовать шифрование. В SQL Server предыдущих версий никаких встроенных средств для шифрования данных в базах данных не было (кроме специальных ситуаций, например, шифрование определений объектов или паролей логинов SQL Server). В SQL Server 2005 такие возможности появились.

Для шифрования данных в SQL Server 2005 предусмотрено четыре способа:

- шифрование при помощи сертификатов. Сертификат при этом должен присутствовать в виде объекта в базе данных (в SQL Server Management Studio можно просмотреть имеющиеся сертификаты в контейнере **Databases | Имя_базы_данных | Security | Certificates**);
- шифрование при помощи асимметричных ключей. Алгоритм здесь такой же, как и при использовании сертификатов. От сертификата асимметричный ключ отличается тем, что в нем не предусмотрено дополнительных полей с информацией о том, кому он выдан, для каких целей, до какого времени действителен и т. п. Имеющиеся асимметричные ключи можно просмотреть в контейнере **Asymmetric Keys**, который находится там же, где и контейнер **Certificates**;
- шифрование при помощи симметричных ключей. Используются намного более быстрые алгоритмы, по сравнению с асимметричными ключами. Сами симметричные ключи также создаются в виде объектов базы данных и могут быть защищены сертификатом, другим симметричным ключом, асимметричным ключом или просто паролем. Просмотреть их можно при помощи контейнера **Symmetric Keys**;
- простое шифрование при помощи паролей.

Отметим также, что вам совсем не обязательно ограничиваться только встроенными средствами SQL Server 2005. Он также позволяет использовать в коде

Transact-SQL вызовы к сборкам .NET. Поэтому для шифрования данных можно использовать классы из пространства имен System.Security.Cryptography в .NET Framework или свои собственные сборки.

5.6.4. Создание сертификатов и ключей и применение криптографических функций

Представьте себе простую задачу. У вас есть база данных db1, а в ней — таблица dbo.SecretTable с единственным столбцом Secret типа nvarchar:

```
USE DB1;
GO
CREATE TABLE dbo.SecretTable(Secret nvarchar(100));
```

Вам нужно поместить в эту таблицу текстовую информацию, зашифрованную при помощи сертификата.

Первое, что вам потребуется сделать, — создать сертификат. Это можно сделать при помощи команды CREATE CERTIFICATE. Самый простой вариант этой команды выглядит так:

```
USE DB1;
CREATE CERTIFICATE SelfSignedCert1
    ENCRYPTION BY PASSWORD = 'P@ssw0rd'
    WITH SUBJECT = 'Проверка шифрования',
    START_DATE = '03/10/2006';
```

Обратите внимание, что для создания сертификата вам не требуется никакой центр сертификации — все необходимые средства уже встроены в SQL Server. Однако вы вполне можете загрузить в базу данных сертификат, который был сгенерирован внешним центром сертификации и сохранен в файле (в отдельном файле должен находиться частный ключ), например:

```
USE DB1;
CREATE CERTIFICATE ExternalCert1
    FROM FILE = 'C:\Certificates\Cert1.cer'
    WITH PRIVATE KEY (FILE = 'C:\Certificates\Cert1Key.pvk',
    DECRYPTION BY PASSWORD = 'P@ssw0rd');
GO
```

Кроме этого, уже готовый сертификат можно также извлечь из подписанный этим сертификатом сборки .NET или из подписанныго исполняемого файла. Параметр DECRYPTION BY PASSWORD позволяет указать пароль, который был использован для защиты данного сертификата.

Параметр ENCRYPTION BY PASSWORD определяет пароль, который потребуется для расшифровки данных, защищенных сертификатом (для шифрования данных он не нужен). Если этот параметр пропустить, то создаваемый сертифи-

кат будет автоматически защищен главным ключом базы данных (*Database Master Key*). Автоматически этот ключ не создается. Чтобы получить возможность работать с ним, нужно предварительно его создать:

```
USE DB1;
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'P@ssw0rd';
```

Кроме пароля, главный ключ базы данных автоматически защищается также главным ключом службы (*Service Master Key*). Этот ключ автоматически генерируется SQL Server 2005 в процессе установки. Будьте очень внимательны при использовании главного ключа базы данных: если вы переустановите сервер (а следовательно, изменится главный ключ службы), зашифрованные данные могут быть потеряны. Чтобы этого не случилось, нужно производить регулярное копирование базы данных *master* или экспорттировать главный ключ службы в файл при помощи команды `BACKUP SERVICE MASTER KEY`.

Обязательный параметр SUBJECT команды CREATE CERTIFICATE определяет цель выдачи сертификата (его значением заполняется соответствующее поле сертификата в соответствии со стандартом X.509v1).

После того как сертификат создан, его можно использовать для шифрования данных. Для этой цели применяется специальная функция `EncryptByCert`:

```
INSERT INTO SecretTable
```

```
values(EncryptByCert(Cert_ID('SelfSignedCert1'),  
N'Секретные данные'));
```

N' Секретные данные'));

Если какой-нибудь пользователь после этого произведет запрос к таблице SecretTable, результаты могут его удивить (рис. 5.11).



Рис. 5.11. Результат выполнения запроса к зашифрованным данным

Обратите внимание, что функция `EncryptByCert` принимает в качестве первого параметра не имя сертификата, а его идентификатор. Требуемый идентификатор легко получить при помощи функции `Cert_ID`.

Расшифровка зашифрованных данных производится при помощи функции DecryptByCert. Единственная проблема при работе с этой функцией заключается в том, что она возвращает расшифрованную информацию с использова-

нием типа данных varbinary, поэтому нужно будет произвести преобразование этого типа данных в nvarchar:

```
SELECT (Convert(Nvarchar(100), DecryptByCert(Cert_ID('SelfSignedCert1'), Secret, N'P@ssw0rd'))) FROM SecretTable;
```

Первый параметр, который принимает функция DecryptByCert, — идентификатор сертификата, возвращаемый той же функцией Cert_ID, второй параметр — строковое значение (или переменная, или, как в нашем случае, имя столбца), третий параметр — пароль, которым был зашифован сертификат при его создании.

Работа с асимметричным ключом выглядит практически аналогично. Вначале нужно создать асимметричный ключ:

```
CREATE ASYMMETRIC KEY AsymKey1
    WITH ALGORITHM = RSA_512
    ENCRYPTION BY PASSWORD = 'P@ssw0rd';
```

Обратите внимание, что при создании асимметричного ключа, кроме пароля, требуется указать длину создаваемого ключа. В вашем распоряжении три варианта: 512, 1024 и 2048 бит.

После этого при помощи созданного ключа вы можете производить шифрование данных:

```
INSERT INTO SecretTable
    values(EncryptByAsymKey(AsymKey_ID('AsymKey1'), N'Секретные данные'));
```

И их расшифровку:

```
SELECT (Convert(Nvarchar(100),
DecryptByAsymKey(AsymKey_ID('AsymKey1'),Secret, N'P@ssw0rd')))
    FROM SecretTable;
```

При использовании симметричных ключей шифрование производится быстрее, чем при применении асимметричных алгоритмов, поэтому при работе с большими объемами данных рекомендуется использовать именно их. Применение симметричных ключей выглядит очень похоже. Правда, есть и небольшие отличия. Во-первых, при создании симметричного ключа его можно защищать не только паролем, но и другим симметричным ключом, асимметричным ключом или сертификатом. Во-вторых, при создании симметричного ключа вы можете указать один из восьми алгоритмов шифрования, поддерживаемых SQL Server 2005 (DES, TRIPLE_DES, RC2, RC4, DESX, AES_128, AES_192, AES_256). Само создание симметричного ключа может выглядеть так:

```
CREATE SYMMETRIC KEY SymKey1
    WITH ALGORITHM = AES_128
    ENCRYPTION BY PASSWORD = 'P@ssw0rd';
```

Перед использованием ключа (для шифрования или расшифровки данных) его нужно обязательно открыть. Это достаточно сделать только один раз в течение сеанса работы пользователя:

```
OPEN SYMMETRIC KEY SymKey1 DECRYPTION BY PASSWORD = 'P@ssw0rd';
```

А дальше используем его привычным способом. Отличаются только названия функций:

```
INSERT INTO SecretTable  
    values(EncryptByKey(Key_GUID('SymKey1'), N'Секретные данные'));  
GO
```

Обратите внимание, что при расшифровке данных нет необходимости передавать функции DecryptByKey имя симметричного ключа и пароль. Будут автоматически подставляться данные открытого при помощи команды OPEN SYMMETRIC KEY ключа:

```
SELECT (Convert(Nvarchar(100), DecryptByKey(Secret))) FROM SecretTable;
```

SQL Server 2005 позволяет производить шифрование данных также просто при помощи пароля. Для этого используется функция EncryptByPassPhrase. В самом простом варианте она принимает только пароль и данные, которые необходимо зашифровать:

```
INSERT INTO SecretTable  
    values(EncryptByPassphrase('P@ssw0rd', N'Секретные данные'));  
GO
```

Расшифровка производится при помощи функции DecryptByPassphrase:

```
SELECT (Convert(Nvarchar(100), DecryptByPassphrase('P@ssw0rd', Secret)))  
FROM SecretTable;
```

Задание для самостоятельной работы 5.1. Назначение прав на объекты SQL Server 2005 и изменение контекста выполнения

Задание:

1. Создайте два логина SQL Server 2005. Для первого логина используйте имя Login1 и пароль P@ssw0rd1, а для второго — Login2 и P@ssword2.
2. Предоставьте логину Login1 права на схему HumanResources в базе данных AdventureWorks и убедитесь, что он может выполнять запросы к любым таблицам этой схемы. Проверьте также отсутствие у логина Login2 прав на выполнение запросов к таблицам в схеме HumanResources.

3. Предоставьте логину Login2 права на выполнение запроса от имени логина Login1. Напишите код запроса с использованием конструкции EXECUTE AS, в ходе которого пользователь Login2 смог бы выполнять запрос к таблице HumanResources.Employee от имени пользователя Login1.

Решение:

К пункту 1 задания — создание логинов:

1. Запустите SQL Server Management Studio и подключитесь к своему локальному серверу SQL Server 2005 при помощи аутентификации Windows. Затем нажмите кнопку **New Query**, чтобы открыть редактор кода Transact-SQL.
2. Введите и выполните в окне редактора кода следующие команды:

```
USE master;
GO
CREATE LOGIN Login1 WITH PASSWORD = N'P@ssw0rd1', DEFAULT_DATABASE =
master, CHECK_EXPIRATION = OFF, CHECK_POLICY = OFF;
GO
CREATE LOGIN Login2 WITH PASSWORD = N'P@ssw0rd2', DEFAULT_DATABASE =
master, CHECK_EXPIRATION = OFF, CHECK_POLICY = OFF;
GO
```

Примечание

Создание логинов SQL Server 2005 можно также производить при помощи графического интерфейса — из контейнера **Security | Logins** в **Object Explorer**.

К пункту 2 задания — предоставление разрешений на схему и проверка прав:

1. Чтобы создать объект пользователя в базе данных AdventureWorks для логина Login1, можно выполнить следующий код:

```
USE AdventureWorks;
GO
CREATE USER Login1 FOR LOGIN Login1;
GO
```

Примечание

Создание объекта пользователя можно также произвести при помощи графического интерфейса — из контейнера **Databases | AdventureWorks | Security | Users** в **Object Explorer**.

2. Чтобы предоставить созданному пользователю Login1 права на чтение для объектов схемы HumanResources, можно выполнить код:

```
GRANT SELECT ON SCHEMA::HumanResources TO Login1;
```

Примечание

При помощи графического интерфейса эту операцию можно произвести на вкладке **Permissions** свойств объекта схемы HumanResources в контейнере **Databases | AdventureWorks | Security | Schemas** в **Object Explorer**.

- Для проверки полученных прав можно подключиться от имени логина Login1 (в меню **File | New** выбрать **Database Engine Query** и открывшемся окне ввести имя пользователя и пароль для логина Login1). Затем в открывшемся от имени Login1 окне редактора кода ввести и выполнить запрос, например:

```
USE adventureworks;
GO
SELECT * FROM HumanResources.Employee;
```

Для пользователя Login1 запрос будет выполнен успешно. Если вы попробуете выполнить такой запрос от имени пользователя Login2, то вернется ошибка.

К пункту 3 задания — предоставление права на выполнение с помощью выражения **EXECUTE AS**:

- Для того чтобы предоставить право логину Login2 выполнять команды от имени логина Login2, можно выполнить следующий код (от имени администратора сервера):

```
USE master;
GO
GRANT IMPERSONATE ON LOGIN::Login1 TO Login2;
GO
```

Примечание

При помощи графического интерфейса такое предоставление прав можно произвести из свойств логина Login2 (не Login1!) на вкладке **Securables**.

- Для проверки предоставленных прав и изменения контекста выполнения можно воспользоваться следующим запросом (его нужно выполнить, подключившись к серверу от имени логина Login2):

```
EXECUTE AS LOGIN = 'Login1';
GO
USE Adventureworks;
GO
SELECT * FROM HumanResources.Employee;
GO
```

Задание для самостоятельной работы 5.2. Шифрование информации в таблицах баз данных

Ситуация:

В соответствии с новой политикой вашей организации информация о зарплатной плате сотрудникам за предыдущие периоды должна храниться только в зашифрованном виде с использованием алгоритма шифрования AES с длиной ключа не менее 256 бит.

Задание:

1. Создайте в базе данных AdventureWorks симметричный ключ PayKey для применения с алгоритмом AES_256. Ключ должен быть защищен паролем P@ssw0rd.
2. Создайте в базе данных AdventureWorks копию таблицы HumanResources.EmployeePayHistory. Новая копия должна называться HumanResources.EmployeePayHistoryEncrypted, и все данные в ней должны быть зашифрованы при помощи созданного вами симметричного ключа.
3. Выполните запрос, который бы вернул все данные из зашифрованной таблицы HumanResources.EmployeePayHistoryEncrypted.

Примечание

Зашифрованную информацию нельзя поместить в столбцы типа int, money и т. п. Поэтому создание таблицы EmployeePayHistoryEncrypted придется производить вручную. Для этого задания используйте для всех столбцов этой таблицы один и тот же тип данных — nvarchar(100). Кроме того, несимвольные типы данных необходимо преобразовать в символьные (например, nvarchar(100)) перед передачей их шифрующей функции.

Решение:

К пункту 1 задания — создание симметричного ключа:

1. Команда на создание симметричного ключа в соответствии с поставленными условиями может выглядеть так:

```
USE AdventureWorks
GO
CREATE SYMMETRIC KEY PayKey
    WITH ALGORITHM = AES_256
    ENCRYPTION BY PASSWORD = 'P@ssw0rd'
```

К пункту 2 задания — создание зашифрованной копии таблицы:

1. Код для создания зашифрованной копии таблицы HumanResources.

EmployeePayHistory может выглядеть так:

```
USE AdventureWorks
```

```
GO
```

```
-- Создаем таблицу для вставки шифрованных данных
```

```
CREATE TABLE HumanResources.EmployeePayHistoryEncrypted
(EmployeeID nvarchar(100),
 RateChangeDate nvarchar(100),
 Rate nvarchar(100),
 PayFrequency nvarchar(100),
 ModifiedDate nvarchar(100))
```

```
GO
```

```
-- Открываем симметричный ключ
```

```
OPEN SYMMETRIC KEY PayKey DECRYPTION BY PASSWORD = 'P@ssw0rd';
```

```
-- Вставляем данные в таблицу при помощи INSERT INTO
```

```
INSERT INTO AdventureWorks.HumanResources.EmployeePayHistoryEncrypted
(EmployeeID, RateChangeDate, Rate, PayFrequency, ModifiedDate)
```

```
SELECT
```

```
EncryptByKey(Key_Guid('PayKey'), CONVERT(nvarchar(100), EmployeeID)),
EncryptByKey(Key_Guid('PayKey'), CONVERT(nvarchar(100),
RateChangeDate)),
```

```
EncryptByKey(Key_Guid('PayKey'), CONVERT(nvarchar(100), Rate)),
```

```
EncryptByKey(Key_Guid('PayKey'), CONVERT(nvarchar(100), PayFrequency)),
```

```
EncryptByKey(Key_Guid('PayKey'), CONVERT(nvarchar(100), ModifiedDate))
FROM AdventureWorks.HumanResources.EmployeePayHistory
```

К пункту 3 задания — запрос к зашифрованным данным:

1. Запрос, возвращающий данные из зашифрованной таблицы

EmployeePayHistory, может выглядеть так:

```
OPEN SYMMETRIC KEY PayKey DECRYPTION BY PASSWORD = 'P@ssw0rd';
```

```
SELECT
```

```
Convert(Nvarchar(100), DecryptByKey(EmployeeID)) AS EmployeeID,
```

```
Convert(Nvarchar(100), DecryptByKey(RateChangeDate)) AS RateChangeDate,
```

```
Convert(Nvarchar(100), DecryptByKey(Rate)) AS Rate,
```

```
Convert(Nvarchar(100), DecryptByKey(PayFrequency)) AS PayFrequency,
```

```
Convert(Nvarchar(100), DecryptByKey(ModifiedDate)) AS ModifiedDate
```

```
FROM AdventureWorks.HumanResources.EmployeePayHistoryEncrypted
```

Примечание

В реальной жизни вам, возможно, потребовалось бы произвести дополнительные преобразования, чтобы вернуть информацию в виде значений с типами данных int, money и т. п. В этом примере для простоты все данные возвращаются как nvarchar(100).



ГЛАВА 6

Резервное копирование и восстановление баз данных SQL Server 2005

В предыдущих главах вы познакомились с тем, как производится установка SQL Server 2005, научились создавать и настраивать базы данных, предоставлять пользователям разрешения и защищать SQL Server. Обычно следующее, что нужно сделать администратору, — это продумать и настроить систему резервного копирования данных на SQL Server. Как это сделать в SQL Server 2005, рассматривается далее в этой главе.

6.1. Основы резервного копирования SQL Server 2005

Есть замечательная фраза: "Кто не проводит резервное копирование, тот искушает судьбу". И это действительно так. Никакие RAID-массивы и кластеры не уберегут вас, например, от ошибки пользователя, который удалил важные данные, или от присланного разработчиками обновления, которое может привести базу данных в неработоспособное состояние. Поэтому, как правило, на предприятиях не стоит вопрос, проводить резервное копирование баз данных SQL Server 2005 или нет. Речь идет о том, как лучше всего это сделать.

Конечно, существует множество факторов, которые нужно учитывать при принятии решения о конкретном способе проведения резервного копирования: требования к скорости восстановления, размер базы данных, бюджет, который есть в вашем распоряжении, требования к производительности и выбранные режимы восстановления базы данных и т. п. Эти факторы будут рассмотрены в следующих разделах этой главы. Пока отметим только некоторые моменты.

В этой главе рассматривается только резервное копирование стандартными средствами SQL Server 2005. В принципе, значительному числу предприятий этих возможностей вполне достаточно. Однако использовать для резервного копирования только встроенные средства SQL Server 2005 — это все равно, что для работы с документами использовать только Блокнот, поставляемый с операционной системой. Готовить документы, конечно, в нем можно, но у таких программ, как Word, возможностей значительно больше.

Что нельзя сделать стандартными средствами резервного копирования SQL Server 2005?

Приведем краткий перечень недостающих возможностей:

- невозможно производить резервное копирование на удаленный стриммер (только на подключенный к локальному компьютеру);
- если у вас несколько серверов SQL Server 2005, вы не сможете управлять процессом резервного копирования из единого центра: запускать резервное копирование придется на каждом сервере отдельно;
- нет возможности шифровать резервную копию или производить сжатие помещаемых на резервную копию данных (стандартными средствами можно включить только аппаратное сжатие, если оно поддерживается стриммером);
- нет возможности производить резервное копирование отдельных объектов базы данных (только целиком базу данных или отдельные файлы данных и файловые группы);
- нет возможности генерировать отчеты о проведении резервного копирования в пользовательском формате.

Поэтому на многих предприятиях используются коммерческие программные продукты для выполнения резервного копирования. По наблюдениям автора, на отечественных предприятиях чаще всего используются Veritas NetBackup, ArcServe и Legato. У каждой из этих программ есть свои возможности, преимущества и недостатки. Рассматривать их здесь мы не будем, отметим только, что для проведения резервного копирования баз данных SQL Server 2005 необходимо, чтобы в комплекте поставки продукта был предусмотрен программный модуль для резервного копирования именно этой версии SQL Server.

6.2. Планирование резервного копирования

6.2.1. Лента или жесткий диск?

Первый вопрос, который нужно решить при реализации системы резервного копирования, — куда помещать резервные копии.

В распоряжении администраторов на предприятиях есть три варианта:

- самый лучший с технической точки зрения, но и самый дорогой вариант — это использование ленточных библиотек. Эти устройства отличаются высокой скоростью работы и надежностью. Часто с ними поставляется специальное программное обеспечение для проведения резервного копирования. Главный недостаток очевиден: не каждое предприятие может приобрести устройство стоимостью в десятки тысяч долларов. Поэтому обычно ленточные библиотеки используются только для серверов баз данных, на которых работают критически важные приложения;
- наиболее традиционный вариант — проведение резервного копирования на стриммер (устройство записи с магнитной лентой), подключенный локально к компьютеру, на котором работает SQL Server 2005;
- третий вариант — проведение резервного копирования на жесткий диск или RAID-массив, подключенный либо локально к тому компьютеру, на котором работает SQL Server, либо к другому компьютеру. Во втором случае резервное копирование может производиться по сети.

Что же выбрать?

Конечно, если есть возможность, лучше всего приобрести ленточную библиотеку. Если же бюджет IT-подразделения таких расходов не предусматривает, выбор между стриммером и жестким диском может оказаться непростой задачей. В пользу стриммера говорит его дешевизна, простота и надежность, удобство замены и транспортировки (например, в специальное хранилище за пределами офиса). Главный аргумент в пользу жесткого диска — скорость работы. Даже не очень быстрые диски с IDE-интерфейсом все равно работают намного быстрее, чем стриммеры. Выигрыш по скорости достигается и за счет того, что жесткий диск — это устройство произвольного, а не последовательного доступа, как магнитные ленты (не надо ничего перематывать). Кроме того, на жесткие диски помещается больше информации. На момент написания этой книги жесткие диски IDE размером 200—300 Гбайт продавались по ценам, вполне доступным не только предприятиям, но и домашним пользователям, а такого объема вполне достаточно для проведения резервного копирования большинства баз данных.

Самое лучшее решение — это объединить преимущества жестких дисков и магнитных лент. Имеет смысл всегда производить резервное копирование только на жесткий диск (локально или по сети), и всегда хранить на жестком диске только последнюю резервную копию (или несколько последних копий). А уже с этого жесткого диска можно производить копирование данных на магнитную ленту.

При этом достигаются сразу несколько целей:

- резервное копирование (и восстановление последней резервной копии) производится только с использованием диска, а значит, очень быстро;
- архивные копии (на начало месяца, квартала и т. п.) будут храниться на дешевых и приспособленных для длительного хранения стриммерах;
- перенос данных на стриммер (который иногда требует смены картриджей) можно производить в дневное время, не мешая работе SQL Server.

Есть и другие возможности для создания резервных копий, например, резервное копирование с использованием магнитооптики. Но они рассматриваться не будут, потому что SQL Server 2005 поддерживает резервное копирование только на диски (локальные или сетевые) и стриммеры.

Предыдущие версии SQL Server поддерживали еще одно назначение резервного копирования: копирование в именованный канал (*Named Pipe*). SQL Server 2005 такое назначение не поддерживает: вместо него разработчикам средств резервного копирования предложено использовать специальный программный интерфейс, встроенный в SQL Server.

6.2.2. Логические устройства или явно указанный путь?

После того как вы определили, куда будете помещать резервные копии, следующее решение, которое вам нужно принять, — будет ли явно указываться путь для размещения резервной копии или для этой цели будут создаваться вспомогательные объекты, которые называются устройствами резервного копирования.

Устройства резервного копирования (*backup devices*) — это специальные объекты, которые хранятся в базе данных `master`. Их единственное назначение — хранить информацию о пути к физическому файлу в операционной системе или о стриммере. Создать такое устройство можно:

- на графическом интерфейсе — из контейнера **Server Objects | Backup Devices** (Объекты сервера | Устройства резервного копирования) в Management Studio;
- из кода Transact-SQL — при помощи хранимой процедуры `sp_addumpdevice`, например:

```
USE [master];
GO
EXEC sp_addumpdevice @devtype = 'disk', @logicalname =
    'BackupDevice1', @physicalname = 'D:\SQLBackups\BackupFile1.bak';
```

После создания логическое устройство можно использовать для резервного копирования. Например, команда на выполнение резервного копирования базы данных db1 без использования логического устройства может выглядеть так:

```
BACKUP DATABASE db1 TO DISK = 'D:\SQLBackups\BackupFile1.bak';
```

Если же вы создали логическое устройство резервного копирования, то можно использовать такую команду:

```
BACKUP DATABASE db1 TO BackupDevice1;
```

Серверу все равно, используете вы логическое устройство резервного копирования или нет. Его применение никак не повлияет на скорость резервного копирования. Вы также не сможете указать для логического устройства какие-то дополнительные параметры. Однако рекомендуется все-таки использовать устройства резервного копирования по одной простой причине: проще будет изменять скрипты резервного копирования. Предположим, что место назначения резервных копий изменилось (на сервер добавился новый диск или был заменен стриммер). Без устройств резервного копирования вам придется исправлять каждый скрипт. Если же вы заранее позаботились о логическом устройстве, достаточно будет исправить указанный в нем путь (строго говоря, путь исправить для логического устройства нельзя, но можно удалить старое устройство и создать новое — с таким же названием, но другим путем).

6.2.3. Типы резервного копирования

В SQL Server 2005 предусмотрено четыре главных типа резервного копирования.

Первый тип — **полное резервное копирование** (*full backup* или *base backup*, в предыдущих версиях SQL Server — *full database backup*). Это самый очевидный тип резервного копирования. В резервную копию записываются все данные, которые есть в базе данных. Пустые страницы при этом не копируются, поэтому если, например, файлы вашей базы данных имеют размер в 1 Гбайт, а реально данные в них занимают всего лишь 200 Мбайт, то получится резервная копия размером 200 Мбайт.

Конечно, полное резервное копирование, как и все другие типы резервного копирования, производится в оперативном режиме (*online*), без отключения пользователей. Стандартными средствами SQL Server 2005 нельзя произвести резервное копирование тех баз данных и файлов, которые находятся в автономном режиме (*offline*). Их резервное копирование следует производить средствами операционной системы.

Обратите внимание на один момент, с которым часто возникает путаница. Предположим, что полное резервное копирование базы данных началось в 7 часов, а закончилось в 8. Данные по состоянию на какое время оказались помещены в резервную копию — на 7 или 8 часов?

Правильный ответ — на 8 часов. Действительно, в момент начала резервного копирования база данных стабилизируется (т. е. никакие изменения в ее файлы не вносятся для обеспечения целостности резервной копии). Однако после того, как перенос самой базы данных завершен, к резервной копии дописывается информация о всех изменениях, которые были внесены в базу данных во время резервного копирования, т. е. с 7 до 8 часов. При восстановлении резервной копии эта информация используется в автоматическом режиме.

Второй тип резервного копирования — **разностный** (*differential backup*, другое название, которое появилось в SQL Server 2005, — *full differential backup*). В этом случае на резервную копию записываются все изменения, которые были произведены с момента полного резервного копирования. Обратите внимание, что именно с момента последнего **полного** резервного копирования, а не другого разностного! Если вы производите полное резервное копирование по понедельникам, а во все остальные дни недели — разностное, то во вторник в резервную копию будут помещены изменения, которые произошли с понедельника по вторник, в среду — с понедельника по среду (а не со вторника по среду) и т. п.

Конечно же, разностное резервное копирование можно использовать только в дополнение к полному.

Третий тип резервного копирования — **резервное копирование журналов транзакций** (*transaction log backup*). Если вы используете режим восстановления **Full** или **Bulk-logged**, то выполнение такого резервного копирования практически обязательно. Причина проста: если вы не будете производить резервное копирование журналов транзакций, то не будет производиться и их очистка. В результате место в файлах журналов транзакций может закончиться (а если для них установлен неограниченный размер, то закончится и место на диске).

Это резервное копирование можно использовать как аналог добавочного (*incremental*) резервного копирования, которое предусмотрено в операционной системе. Например, можно производить полное резервное копирование по понедельникам, а в каждый другой день недели — резервное копирование журналов транзакций. В результате во вторник в резервную копию будут записаны те изменения, которые произошли с понедельника по вторник, в среду будут записаны изменения со вторника по среду и т. п. Такое резервное копирование будет выполняться быстрее, чем разностное, но придется пожертвовать скоростью восстановления. Если сбой произойдет, например, в

четверг, то при использовании разностного резервного копирования вам потребуются две резервные копии: за понедельник (полная) и за четверг (разностная). В той же ситуации при использовании резервного копирования только журналов транзакций вам потребуются: полная копия за понедельник и копии журналов транзакций за вторник, среду и четверг.

Четвертый тип резервного копирования появился только в SQL Server 2005. Это так называемое **копирующее резервное копирование** (*copy-only backups*). Оно предназначено, в первую очередь, для переноса данных между компьютерами в виде резервных копий. Такой тип резервного копирования разделяется на полное (в резервную копию будут помещены те же данные, что и при обычном полном резервном копировании) и разностное (аналог обычного разностного копирования). Этот тип резервного копирования отличается только тем, что в столбце `is_copy_only` таблицы `backupset` базы данных `msdb` (в эту таблицу помещаются данные о всех созданных резервных копиях) такие резервные копии помечаются специальным флагом. За счет этого флага резервные копии, созданные в копирующем режиме, не учитываются в последовательности обычных резервных копий. Например, если после обычной разностной копии будет создана полная копирующая, то при следующем разностном резервном копировании эта полная копия, созданная в режиме `COPY_ONLY`, будет проигнорирована. И в разностную копию будет помещена информация, измененная с момента обычного полного резервного копирования.

Резервное копирование в режиме `COPY_ONLY` (а также восстановление созданных этим способом копий) невозможно произвести при помощи графического интерфейса Management Studio. Вместо этого вам нужно будет использовать ключевое слово `COPY_ONLY` в командах `BACKUP` и `RESTORE`.

В качестве дополнительного типа резервного копирования можно рассматривать **резервное копирование файлов** (*file backup*) и **файловых групп** (*filegroup backup*). Применяется оно достаточно редко и обычно только в двух ситуациях:

- когда резервная копия всей базы данных не помещается на картридж стриммера. В этом случае размер файлов или файловых групп подбирается под размер картриджа;
- когда в базе данных таблицы можно условно поделить на две группы: таблицы-справочники, в которые изменения практически не вносятся, и пользовательские таблицы, которые изменяются активно. В этом случае каждая группа таблиц помещается в свою файловую группу, а затем проводится резервное копирование каждой файловой группы со своей частотой.

Для обеспечения целостности информации при проведении резервного копирования файлов и файловых групп SQL Server 2005 автоматически определя-

ет поколения резервных копий. Пока не будет завершено резервное копирование всех файлов или файловых групп в рамках одного поколения, журнал транзакций очищаться не будет. Поэтому администраторы применяют этот тип резервного копирования редко.

Необходимо также учитывать, что не все типы резервного копирования баз данных всегда доступны. Выбор типа резервного копирования зависит от режима восстановления базы данных. Подробно про режимы восстановления рассказывалось в разд. 4.5. Здесь мы отметим только, что поскольку при использовании режима восстановления **Simple** журнал транзакций очищается автоматически, то в этом режиме резервное копирование журнала транзакций проводить нельзя. Кроме того, в этом режиме можно производить резервное копирование только тех файловых групп, которые доступны только на чтение.

В документации SQL Server 2005 упоминается также концепция **резервных копий-снимков баз данных** (*snapshot backups*). Создание таких резервных копий производится при помощи службы Volume Shadow Copy (Теневое копирование тома) в Windows Server 2003, но стандартными средствами SQL Server 2005/Windows Server 2003 создать такие резервные копии не удастся. Для этого требуются специальные программные и/или аппаратные средства третьих фирм.

Резервные копии-снимки баз данных (*snapshot backups*) не следует путать со снимками баз данных на определенный момент времени (*database snapshots*), которые создаются обычными средствами SQL Server 2005 и также могут использоваться для восстановления (точнее, для отката базы данных к состоянию на определенное время в прошлом).

6.2.4. Расписание резервного копирования

Последнее, что вам осталось решить при планировании резервного копирования, — какое расписание вы будете использовать. Конечно, выбор здесь зависит от конкретной ситуации. Как минимум, нужно учитывать:

- размер базы данных. Если она очень большая, то стоит подумать о применении разностного резервного копирования;
- режим восстановления (об этом говорилось в предыдущем разделе);
- интенсивность внесения изменений в базу данных. Обычно чем больше изменений вносится в нее, тем чаще нужно производить резервное копирование. Кроме того, при очень больших объемах изменяемых данных применение разностного резервного копирования может оказаться неэффективным;
- насколько быстро в случае сбоя необходимо ввести базу данных в строй. Если она должна быть восстановлена быстро, то нет смысла использовать длинные цепочки резервных копий журналов транзакций.

По опыту работы автора и его общения со слушателями самых разных предприятий можно отметить, что в 9 из 10 случаев на отечественных предприятиях используется самое простое расписание резервного копирования: каждую ночь производится полное резервное копирование всей базы данных, и сразу после этого — резервное копирование журналов транзакций (для очистки). Если база данных не очень большая, то такой режим вполне можно считать оптимальным.

При этом обычно хранятся все резервные копии за последнюю неделю или две недели, а также резервные копии на начало месяца (чтобы можно было при необходимости восстановить базу данных и использовать ее для создания отчетов). Носители с ненужными резервными копиями используются повторно.

Корпорация Microsoft считает идеальным вариантом другое расписание резервного копирования (такой вывод можно сделать из официальных учебных курсов и сертификационных экзаменов):

- раз в неделю — полное резервное копирование;
- раз в сутки (каждую ночь) — разностное резервное копирование;
- несколько раз в день — резервное копирование журналов транзакций.

Этот вариант наилучшим образом подходит для больших баз данных (десятки и сотни гигабайт), которые активно изменяются.

Можно использовать и другие варианты резервного копирования в зависимости от конкретных условий предприятия. Например, для не очень важных баз данных полное резервное копирование можно проводить раз в неделю и раз в день выполнять резервное копирование журналов транзакций.

В любом случае резервное копирование лучше производить тогда, когда нагрузка на сервер минимальна (обычно ночью).

Рекомендуется всегда одновременно с резервным копированием пользовательских баз данных производить резервное копирование баз данных master и msdb. По сравнению с пользовательскими базами данных места потребуется совсем немного, а восстанавливаться в случае каких-то проблем будет намного проще.

6.3. Проведение резервного копирования

6.3.1. Средства для выполнения резервного копирования

Резервное копирование чаще всего производится при помощи скрипта Transact-SQL с командой BACKUP, который запускается по расписанию в авто-

матическом режиме (при помощи задания SQL Server Agent). Однако, кроме скрипта, в вашем распоряжении есть и другие варианты:

- графический интерфейс Management Studio. Отметим, что с его помощью очень удобно сгенерировать скрипт Transact-SQL автоматически. Для этого на экране резервного копирования можно настроить нужные параметры и нажать кнопку **Script** в верхней части экрана;
- план обслуживания базы данных (*database maintenance plan*). При помощи плана обслуживания вы можете не только создать задание на резервное копирование базы данных, но и запланировать его для выполнения по расписанию, настроить генерацию отчета о резервном копировании (и, например, его пересылку по электронной почте). Те же возможности доступны в SQL Server 2005 при помощи пакетов SSIS (DTS). Подробнее про работу с планами обслуживания баз данных будет рассказываться в разд. 8.3, а про работу с пакетами SSIS — в гл. 10;
- утилита командной строки операционной системы `sqlmaint`. Эта утилита оставлена для целей обеспечения обратной совместимости, но, тем не менее, она остается самым простым способом проведения резервного копирования баз данных SQL Server по расписанию с одновременным созданием отчета;
- скрипт или программа, использующие объектные модели SQL-DMO или SMO. Речь об этих объектных моделях пойдет в гл. 9.

Однако какое бы средство для резервного копирования вы не выбрали, основные возможности все равно будут одинаковыми (на графическом экране Management Studio будут недоступны некоторые вспомогательные возможности).

6.3.2. Параметры резервного копирования

При проведении резервного копирования можно указать множество параметров. Эти параметры перечислены далее. Для каждого из них приведено название на графическом интерфейсе Management Studio и соответствующий синтаксис команды `BACKUP`. Окно резервного копирования в Management Studio можно открыть из контекстного меню **Tasks | Backup** для соответствующей базы данных. Другой вариант — воспользоваться контекстным меню для контейнера **Server Objects | Backup Devices**.

Параметры резервного копирования следующие:

- **Database** — это, конечно, имя базы данных, резервное копирование которой вы будете производить. В команде `BACKUP` указывается как `BACKUP DATABASE имя_базы_данных`;

- **Recovery model** (Режим восстановления) — это просто справочная информация о текущем режиме восстановления базы данных. Режим восстановления меняется из свойств базы данных;
- **Backup type** (Тип резервного копирования) — тип резервного копирования. Для полного или разностного резервного копирования используется команда `BACKUP DATABASE` (для разностного еще указывается параметр `WITH DIFFERENTIAL`), для резервного копирования журнала транзакций — команда `BACKUP LOG`. Обратите внимание, что в скрипте вы можете также указать параметр `WITH COPY_ONLY`, который определяет копирующий режим (см. разд. 6.2.3). На графическом экране вы его выбрать не сможете;
- **Backup component** (Компонент для резервного копирования) — позволяет выбрать резервное копирование всей базы данных или отдельных файловых групп (отдельных файлов). В команде `BACKUP` для указания файлов или файловых групп используются ключевые слова `FILE` и `FILEGROUP`, например:

```
BACKUP DATABASE db1 FILEGROUP = 'PRIMARY' TO DISK =
'D:\SQLBackups\BackupFile1.bak';
```

- **Backup set name** (Имя резервной копии) — имя резервной копии (*backup set* — это набор носителей, которые относятся к одной резервной копии). В команде `BACKUP` указывается параметр `NAME`. Получить информацию об имени резервной копии можно при помощи кнопки **Contents** (Содержимое) на том же экране или при помощи команды `RESTORE HEADERONLY`;
- **Description** (Описание) — описание резервной копии. Указывается при помощи параметра `DESCRIPTION`. Просмотреть можно так же, как и имя резервной копии;
- **Backup set will expire** (Резервная копия устареет) — позволяет указать срок (дату), после которой резервная копия будет считаться устаревшей и будет автоматически перезаписываться SQL Server. Значение 0 этого параметра означает, что копия никогда не будет считаться устаревшей. В команде `BACKUP` для указания срока/даты устаревания используются параметры `RETAINDAYS` (сколько времени сохранять) и `EXPIREDATE` (дата устаревания).

Конечно, при помощи этого параметра вы определяете только возможное поведение SQL Server. Вы вполне можете принудительно перезаписать копию, которая не успела устареть, или не перезаписывать никакие копии (так ведет себя SQL Server по умолчанию);

- **Destination** (Назначение) — место назначения резервной копии в виде файла на диске, стриммера или логического устройства резервного копирования (см. разд. 6.2.2). Можно указать несколько назначений одновре-

менно (но только одного типа). В этом случае запись резервной копии будет производиться параллельно в несколько файлов или на несколько стриммеров для повышения производительности. Конечно, при восстановлении данных вам понадобятся все части резервной копии, созданной таким образом. В команде `BACKUP` для указания файла на диске используется ключевое слово `TO DISK`, а для указания стриммера — `TO TAPE`.

Если вы используете команду `BACKUP`, а не графический интерфейс, то в вашем распоряжении еще одна возможность: при помощи ключевого слова `MIRROR TO` можно указать файл, стриммер или логическое устройство, на котором будет создаваться второй экземпляр вашей резервной копии (полностью идентичный первому). Такое зеркалирование при создании резервных копий используется, конечно, для повышения надежности резервного копирования. Это новая возможность SQL Server 2005;

□ **Overwrite media** (Перезаписать носитель) — параметры (на графическом экране они расположены на вкладке **Options**), позволяющие определить режим перезаписи носителя (файла на диске или магнитной ленты). В вашем распоряжении следующие варианты:

- **Append to the existing media set** (Добавить к существующему набору носителя). Соответствует параметрам `NOFORMAT` (не перезаписывать заголовок носителя) и `NOINIT` (не перезаписывать старую резервную копию);
- **Overwrite all existing backup sets** (Перезаписать все существующие наборы носителя). Соответствует параметрам `NOFORMAT` и `INIT` (заголовок носителя сохранится, но все старые резервные копии будут перезаписаны);
- **Check media set name and backup set expiration** (Проверить имя набора носителей и устаревание резервной копии). Соответствует параметрам `NOFORMAT`, `INIT`, `NOSKIP` (перезапись будет произведена только в том случае, если имя резервной копии совпадает с существующей на носителе, но существующая резервная копия устарела);
- **Backup to a new media set, and erase all existing backup sets** (Произвести резервное копирование на новый набор носителей и удалить все существующие резервные копии). Соответствует параметрам `FORMAT`, `INIT`, `SKIP` (независимо от заголовков и времени устаревания существующей резервной копии носитель (лента или файл) будет полностью перезаписан, включая заголовок).

При использовании команды `BACKUP` у вас есть еще одна возможность контролировать перезапись — определение пароля для резервной копии при помощи параметра `MEDIAPASSWORD`. В отличие от большинства других паро-

лей этот пароль не используется для защиты данных. Данные не шифруются, а при восстановлении резервной копии на другой сервер этот пароль вообще спрашиваться не будет. Единственное назначение такого пароля — защита от перезаписи важных резервных копий и ошибочных восстановлений. Перезаписать резервную копию, защищенную паролем или использовать ее для восстановления на том же сервере будет невозможно, пока вы не введете правильный пароль. На графическом интерфейсе задать пароль для резервной копии нельзя;

- **Verify backup then finished** (Проверить резервную копию после завершения) — проверка целостности резервной копии после завершения резервного копирования. Проверяются размер, структура заголовка и соответствие контрольной сумме (если она была записана). Если запись производилась на стриммер, на проверку может потребоваться время, сопоставимое с самим резервным копированием.

Никакой параметр команды BACKUP этому параметру не соответствует. Вместо этого после окончания резервного копирования нужно будет выполнить команду RESTORE VERIFYONLY;

- **Perform checksum before writing to media** (Выполнять проверку контрольной суммы перед записью на носитель) — при установке этого параметра SQL Server будет, во-первых, проверять контрольную сумму (или контрольный бит) для каждой страницы базы данных, а во-вторых, записывать свои контрольные суммы для резервной копии. Этому параметру соответствует ключевое слово CHECKSUM в команде BACKUP. В предыдущих версиях SQL Server такого параметра не было;
- **Continue on error** (Продолжать при возникновении ошибки) — параметр, определяющий продолжать или нет резервное копирование, если при проверке контрольных сумм в базе данных были обнаружены ошибки. Соответствует параметру CONTINUE_AFTER_ERROR. По умолчанию используется параметр STOP_AFTER_ERROR — при обнаружении подобных ошибок резервное копирование останавливается;
- **Truncate the transaction log** (Очищать журнал транзакций) — переключатель устанавливается в это значение по умолчанию, если вы производите резервное копирование журнала транзакций. Он означает "очистить журнал транзакций после резервного копирования". Поскольку этот режим используется по умолчанию, то в команде BACKUP LOG ему ничего не соответствует;
- **Back up the tail of the log, and leave the database in the restoring state** (Провести резервное копирование остатка журнала и оставить базу данных в режиме восстановления) — этому значению переключателя соот-

ветствуют параметры NOTRUNCATE (не очищать журнал) и NORECOVERY (без восстановления) в команде `BACKUP LOG`. Они означают "произвести резервное копирование журнала без его очистки и перевести базу данных в состояние RESTORING" (при этом она станет недоступной для пользователей). Такой режим используется только в ситуации, когда для ваших серверов настроена автоматическая передача журналов транзакций (*log shipping*), и вы хотите поменять ролями главный и резервный серверы;

- **Unload the tape after backup** (Выгружать ленту после резервного копирования) — соответствует параметру `UNLOAD` команды `BACKUP`. При использовании этого параметра по окончании резервного копирования картридж автоматически извлекается из стриммера (очень удобно в качестве сигнала об окончании). Этот параметр устанавливается по умолчанию при использовании стриммера. Запретить автоматическое извлечение можно при помощи параметра `NOUNLOAD`;
- **Rewind the tape before unloading** (Перемотать ленту перед выгрузкой) — соответствует параметру `REWIND` и устанавливается по умолчанию. Перед извлечением картридж стриммера будет перемотан на начало. Запретить перемотку можно при помощи параметра `NOREWIND`.

В SQL Server 2005 появилась новая команда, которая позволяет просто перемотать ленту в картридже на начало, не выполняя других операций: `RESTORE REWINDONLY`.

Далее представлено еще несколько параметров резервного копирования, которые можно использовать в команде `BACKUP`, но которым нет аналогов на графическом экране:

- **BLOCKSIZE** — позволяет указать оптимальный размер блока для стриммера. Параметр необязательный и влияет только на производительность (в некоторых ситуациях);
- **STATS** — через сколько процентов от общего объема резервного копирования будут выдаваться информационные сообщения. По умолчанию — через каждые 10%;
- **COPY_ONLY** — копирующий тип резервного копирования (см. разд. 6.2.3);
- **RESTART** — в предыдущих версиях SQL Server этот параметр позволял продолжить приостановленную операцию резервного копирования (например, после вставки нового картриджа, когда на старом закончилось место). В SQL Server 2005 этот параметр игнорируется;
- **READ_WRITE_FILEGROUPS** — задает резервное копирование только файловых групп, доступных для записи (открытые только на чтение будут игнорироваться).

В команде BACKUP LOG доступны еще несколько важных параметров, которых нет на графическом экране:

- **STANDBY** — используется вместо NORECOVERY в той же ситуации. Отличается тем, что база данных для пользователей будет открыта только на чтение, т. е. в режиме STANDBY;
- **NO_LOG** и **TRUNCATE_ONLY** (эти ключевые слова являются синонимами) — используются для очистки журнала транзакций без проведения резервного копирования. Обычно используются тогда, когда место в журнале транзакций внезапно закончилось, и вам нужно как можно быстрее обеспечить пользователям возможность нормальной работы.

Приведем несколько примеров команды BACKUP в самых распространенных случаях. Чтобы провести обычное полное резервное копирование базы данных db1 на диск, можно использовать команду вида:

```
BACKUP DATABASE db1 TO DISK = 'D:\SQLBackups\BackupFile1.bak';
```

Чтобы произвести разностное резервное копирование той же базы данных, можно использовать команду:

```
BACKUP DATABASE db1 TO DISK = 'D:\SQLBackups\BackupFile1.bak'  
WITH DIFFERENTIAL;
```

Команда на проведение резервного копирования журнала транзакций этой базы данных в самом простом варианте может выглядеть так:

```
BACKUP LOG db1 TO DISK = 'D:\SQLBackups\BackupFile1.bak';
```

Еще раз напомним, что нужный скрипт для выполнения резервного копирования можно сгенерировать автоматически при помощи кнопки **Script** в Management Studio. Если воспользоваться пунктом **Script Action to Job** (Отскриптовать действие в задание) раскрывающегося меню для этой кнопки, то можно автоматически создать задание SQL Server Agent, которое будет выполнять резервное копирование по расписанию. Достаточно на вкладке **Schedules** (Расписания) свойств созданного задания настроить для него расписание, и задача автоматизации резервного копирования будет решена.

Поговорим также про ограничения, которые налагаются на базы данных при выполнении резервного копирования. В это время нельзя:

- создавать новые файлы базы данных и удалять старые;
- нельзя уменьшать размер существующих файлов.

Нельзя также производить резервное копирование базы данных, которая находится в автономном режиме (*offline*). Конечно, рекомендуется производить резервное копирование баз данных только в то время, когда нагрузка на сервер со стороны пользователей минимальна.

Если SQL Server используется как настольное приложение на компьютерах пользователей (такая ситуация встречается на предприятиях), то можно научить пользователей производить резервное копирование самостоятельно. Можно показать им возможности Management Studio или команду BACKUP, но проще всего научить их переводить базу данных в автономный режим и выполнять резервное копирование файлов баз данных и журналов транзакций просто при помощи копирования файлов средствами операционной системы.

6.3.3. Получение информации о резервном копировании и создание отчетов

Очень часто администраторам на предприятии необходимо не только проводить резервное копирование, но и отчитываться об этом (например, если они работают в филиалах). Конечно, информационные сообщения выдаются и при помощи команды BACKUP, но использовать их для создания отчетов неудобно. В распоряжении администраторов два варианта:

- воспользоваться стандартным форматом отчетов о резервном копировании;
- создать отчет в своем собственном формате при помощи информации из таблиц с историей резервного копирования.

Для первого варианта проще всего использовать возможности утилиты командной строки sqlmaint. Например, чтобы провести полное резервное копирование базы данных db1 с созданием отчета о резервном копировании в файле D:\BackupReport.html, можно воспользоваться следующей командой:

```
sqlmaint -S "LONDON7" -D "db1" -BkUpDB "D:\SQLBackups" -BkUpMedia
DISK -HtmlRpt "D:\BackupReport.html"
```

В результате будет произведено резервное копирование и создан отчет, аналогичный представленному на рис. 6.1.

Database	Action	Date	Result	Message
db1	Backup database	04.01.2006 19:44:01	Success	Backup Destination: [D:\SQLBackups\db1_db_200601041944.BAK]

Рис. 6.1. Стандартная форма отчета о резервном копировании

Если такая форма вас не устраивает, то придется создавать свои отчеты на основе таблиц с историей резервного копирования. Проще всего это сделать при помощи средств разработки отчетов, таких как Access, Crystal Reports или Reporting Services (Reporting Services входит в состав SQL Server 2005). Скорее всего, вам потребуется информация из таблицы `backupset` в базе данных `msdb`. В этой таблице есть информация о имени сервера и базы данных, времени проведения резервного копирования, пользователе, который выполнял резервное копирование, имени резервной копии и т. п. Дополнительную информацию можно найти в таблицах `backupfile`, `backupfilegroup`, `backupmediaset` и `backupmediafamily` базы данных `msdb`.

6.4. Основы восстановления баз данных

Резервное копирование обычно производится для того, чтобы в случае сбоя можно было произвести восстановление. Но прежде чем знакомиться с процедурой восстановления баз данных SQL Server, необходимо уточнить значение нескольких терминов.

Первый термин — *restore*. С точки зрения SQL Server, этот термин можно перевести как "восстановление с носителя". Чаще всего восстановлением с носителя приходится заниматься в ситуациях, когда база данных повреждена физически или нужно исправить большую ошибку пользователя. Во время этого процесса производится перенос данных из резервной копии на сервер базы данных.

Второй термин — *recovery*. Его можно перевести как "восстановление работоспособности". Во время процедуры *recovery* устраняются все проблемы, которые могут быть с базой данных (например, возникшие из-за незавершенных транзакций), и база данных открывается для доступа пользователей. Процедура *recovery* должна быть произведена после восстановления с носителя — *restore*, однако она запускается и в других ситуациях. Например, если работа сервера была завершена некорректно (например, пропало питание), то эта процедура вернет все базы данных в работоспособное состояние.

Термин *failure* (сбой) обычно означает сбой в работе базы данных, например, появились ошибки на диске, на котором была расположена база данных. Операционная система и программные файлы сервера при этом остались в рабочем состоянии, и вам нужно произвести восстановление только базы данных.

Термин *disaster* (катастрофа) означает катастрофический отказ сервера, например, из-за скачка напряжения, пожара, затопления и т. п. При восстановлении в случае такого катастрофического отказа вам придется вначале установить операционную систему и программное обеспечение SQL Server, а потом уже производить восстановление рабочих баз данных.

Общий план восстановления обычно выглядит следующим образом:

1. Вначале производится процедура *restore* — необходимая информация восстанавливается с носителя. Вы можете восстановить только полную резервную копию, а уже после этого произвести восстановление разностной резервной копии и резервных копий журналов транзакций. Официальное название этого этапа — **фаза копирования данных** (*data copy phase*).
2. Если производится также восстановление журналов транзакций, то следующим действием SQL Server записывает в базу данных всю информацию о завершенных транзакциях из журнала транзакций. Эта операция называется *roll forward* (завершение). Сам этап называется **фазой повтора** (*redo phase*), а оба первых этапа вместе — **этап завершения** (*roll forward step*).
3. Только в редакции SQL Server 2005 Enterprise Edition пользователям открывается доступ к базе данных. Открытие доступа на этом этапе — это новая возможность SQL Server 2005. Она имеет свое название — *fast recovery* (быстрое восстановление). Если же пользователь попытается обратиться к данным, измененным незавершенными транзакциями, то доступ ему будет закрыт за счет механизма блокировок.
4. Затем SQL Server обнаруживает в журнале все незавершенные транзакции и отменяет их. Эта операция называется *rollback* — откат транзакций, а сам этап называется **этапом отката** (*rollback phase*).
5. После этого к базе данных открывается доступ в обычном режиме во всех версиях SQL Server.

Отметим еще несколько моментов, связанных с процессом восстановления SQL Server 2005:

- для восстановления вы можете использовать не только резервные копии, которые были созданы в версии SQL Server 2005, но и те, которые были созданы на версиях 7.0 и 2000. Обновление до версии 2005 произойдет автоматически. Конечно, такую возможность следует рассматривать как еще один вариант обновления баз данных;
- создатели SQL Server 2005 активно рекламируют новую возможность — восстановление на открытой базе данных. На самом деле такое восстановление можно производить только тогда, когда в базе данных несколько файловых групп, и восстанавливаемая файловая группа находится в автономном режиме (*offline*). Поэтому на самом деле пользователи работать с восстанавливаемыми данными, конечно, не смогут;
- в SQL Server 2005 восстановление полнотекстовых индексов производится вместе с базами данных;

- информация о восстановлении, как и информация о резервном копировании, записывается в служебные таблицы базы данных msdb. Используются таблицы `restorehistory`, `restorefile` и `restorefilegroup`.

6.5. Подготовка к восстановлению

Перед восстановлением вам, скорее всего, потребуется произвести некоторые подготовительные действия.

Первое, что нужно сделать, — это запретить пользователям доступ к базе данных, подлежащей восстановлению. Это можно сделать разными способами:

- для большинства баз данных достаточно установить для параметра **Restrict Access** (Ограничить доступ) свойств базы данных значение `Restricted`. Если же пользователи вашей базы данных могут подключаться с правами `dbo`, то для этого параметра можно установить значение `Single`;
- если на сервере имеется только одна рабочая база данных, то лучше просто на время восстановления отключить сетевой доступ к SQL Server. Для этого можно, например, на время восстановления отключить протокол TCP/IP в контейнере **SQL Server 2005 Network Configuration** в SQL Server Configuration Manager.

Если к базе данных в настоящий момент подключены пользователи, то их соединения придется закрыть.

Может случиться так, что база данных повреждена настолько сильно, что изменить ее свойства не удается. Она при этом может находиться в состоянии *suspect* (подозрительное) или в автономном режиме *offline* (информацию о состоянии можно просмотреть, например, из контейнера **Databases** в Management Studio). Если база данных находится в автономном режиме, то запустить ее восстановление вам не удастся. В этой ситуации самый простой выход — отсоединить (*detach*) поврежденную базу данных и произвести восстановление с резервной копии так, как будто эта база данных отсутствует на сервере вообще. Отметим, что для того, чтобы отсоединить базу данных, помеченную как подозрительная (*suspect*), ее необходимо вначале перевести в состояние "экстренной необходимости" (*emergency*):

```
ALTER DATABASE db1 SET emergency;
```

Если база данных находится в рабочем режиме, а времени у вас есть, то лучше, конечно, подстражоваться, создав еще одну, самую свежую резервную копию этой базы данных и журнала транзакций (или только журнала транзакций в зависимости от ситуации).

После того, как доступ пользователей к базе данных закрыт, рекомендуется проверить заголовки ваших резервных копий. Для этого можно использовать следующие команды:

- RESTORE FILELISTONLY** — возвращает информацию о списке файлов и журналов транзакций, которые помещены в данную резервную копию. Эта информация берется из таблицы `backupfile` базы данных `msdb`;
- RESTORE HEADERONLY** — возвращает информацию об имени резервной копии, ее типе, описании, времени создания и времени устаревания и т. п. Эта информация берется из таблицы `backupset` базы данных `msdb`;
- RESTORE LABELONLY** — выводит служебную информацию о метке носителя. В основном метка нужна для картриджей стриммеров, но может применяться и для файлов. Информация берется в том числе и из таблицы `backupmediaset` базы данных `msdb`.

Пример выполнения команды на просмотр информации о резервной копии может выглядеть так:

```
RESTORE FILELISTONLY FROM backupdevice1;
```

Конечно, вы можете обратиться к таблицам с историей резервного копирования в базе данных `msdb` и напрямую.

6.6. Проведение восстановления

После того как подготовка завершена, можно приступить к самому восстановлению. Запустить восстановление можно при помощи графического интерфейса Management Studio (контекстное меню **Restore Database** для контейнера **Databases** или контекстное меню **Tasks | Restore** для контейнера базы данных) или при помощи команды **RESTORE**. Как обычно, опишем возможности, которые представляет графический интерфейс, и приведем информацию о тех параметрах команды **RESTORE**, которым они соответствуют:

- Destination to restore ... To database** (Назначение восстановления ... в базу данных) — это, конечно, имя восстанавливаемой базы данных. Обратите внимание, что вместо выбора базы данных из списка вы можете ввести свое имя. В этом случае из резервной копии на сервере будет создана новая база данных. В некоторых случаях может быть удобно восстановить копию существующей базы данных под другим именем, а затем при необходимости старую базу данных удалить, а восстановленную переименовать, присвоив ей старое название.

Команда на восстановление базы данных в самом простом варианте может выглядеть так:

```
RESTORE DATABASE db2 FROM DISK = 'D:\SQLBackups\BackupFile1.bak';
```

При этом резервная копия вполне могла быть создана для базы данных db1, а не db2;

- **To a point of time** (На момент времени) — позволяет задать восстановление на определенный момент времени. Обычно используется только в ситуации, когда пользователь совершил ошибку (например, удалил важные данные) и вы знаете примерно, когда это произошло. Используется только при восстановлении журналов транзакций. Этот переключатель соответствует параметру STOPAT команды RESTORE, например, WITH STOPAT = '01/06/2006 12:14:24'. Для команды RESTORE можно указать еще два параметра:

- восстановление на метку транзакции. Обычно метка транзакции применяется перед выполнением рискованных операций (применение исправлений от разработчиков, очистка или массовая загрузка данных и т. п.). Создать метку транзакции можно очень просто:

```
BEGIN TRAN mark1 WITH MARK;  
COMMIT TRAN;
```

Для восстановления потребуется использовать параметр WITH STOPATMARK = 'mark1', чтобы остановиться точно на этой метке или WITH STOPBEFOREMARK = 'mark1' для остановки точно перед этой меткой;

- восстановление на номер последовательности в журнале транзакций (log sequence number, LSN). Номер LSN есть у каждой операции, которая зафиксирована в журнале транзакций. К сожалению, стандартными средствами просмотреть журнал транзакций и найти LSN для транзакции, с которой начались проблемы, невозможно. Для этой цели придется использовать утилиты третьих фирм, например, Lumigent Log Explorer. После того как номер LSN найден, можно использовать те же параметры STOPATMARK и STOPBEFOREMARK, но синтаксис будет немного другим, например:

```
RESTORE LOG db1 FROM DISK = 'D:\SQLBackups\BackupFile1.bak'  
WITH STOPATMARK = 'lsn:120';
```

- **From database** (Из базы данных) — для обнаружения резервных копий будет использоваться история резервного копирования из таблиц базы данных msdb. В списке можно выбрать не только текущую базу данных, но и другие базы данных, которые есть на этом сервере;

- **From device** (Из устройства) — вам потребуется указать местонахождение резервной копии явно. Эта возможность используется в тех ситуациях, когда вам нужно восстановить базу данных на другой сервер или местонахождение резервной копии изменилось. В любом случае вам потребуется выбрать логическое устройство резервного копирования, картридж

стриммера или файл на диске. Еще одна возможность (доступная только в Enterprise Edition и только при полном восстановлении базы данных) — использовать в качестве источника снимок базы данных (*database snapshot*);

- **Select the backup sets to restore** (Выбрать резервную копию для восстановления) — в этом списке вам потребуется установить флажки напротив тех резервных копий, которые вы планируете восстановить. Обратите внимание, что флажки можно поставить напротив нескольких резервных копий. В этом случае для каждой выбранной резервной копии будет выполнена отдельная команда `RESTORE`.

При помощи этого же списка можно получить множество дополнительной информации о восстанавливаемых резервных копиях (если прокрутить список вправо): о времени резервного копирования, о размере резервной копии, о пользователе, который производил это копирование, и т. п.

Дополнительные и очень важные параметры восстановления представлены на вкладке **Options** окна восстановления базы данных Management Studio:

- **Overwrite the existing database** (Перезаписывать существующую базу данных) — установленный флажок позволяет перезаписать существующую базу данных. Фактически он отменяет проверки, которые призваны не допустить потери данных в случае ошибочного восстановления. Таких проверок предусмотрено три:

- запрещено восстанавливать резервную копию чужой базы данных на сервер, если под этим именем на сервере есть своя база данных;
- запрещено перезаписывать файлы, которые относятся к базам данных, находящимся в автономном режиме (*offline*), и, кроме этого, вообще любые файлы, которые не относятся к SQL Server;
- запрещено производить восстановление базы данных, если на ней осталась часть журнала транзакций, резервное копирование которой еще не производилось (*tail-log*). Это новая проверка, которая появилась только в SQL Server 2005.

Чтобы эти проверки отменить, нужно установить указанный флажок или использовать параметр `WITH REPLACE` в команде `RESTORE`;

- **Preserve the replication settings** (Сохранить настройки репликации) — сохранить настройки репликации при восстановлении. Соответствует параметру `KEEP_REPLICATION` команды `RESTORE`. Обычно используется только тогда, когда база данных одновременно участвует и в репликации, и в автоматической доставке журналов (*log shipping*).

- **Prompt before restoring each backup** (Выводить приглашение перед каждым восстановлением) — выводить приглашение перед восстановлением

каждой следующей резервной копии из выбранного вами списка. Обычно этот параметр используется только тогда, когда каждая копия лежит на своем картридже стриммера, и вам нужно их менять. Этот параметр можно настроить только на графическом экране Management Studio, поскольку в коде Transact-SQL для восстановления каждой резервной копии вам придется использовать свою собственную команду RESTORE;

- **Restrict access to the restored database** (Ограничить доступ к восстанавливаемой базе данных) — после восстановления доступ будет открыт только членам роли базы данных db_owner и членам серверных ролей dbcreator и sysadmin. Этот параметр обычно применяется в тех случаях, когда после восстановления базы данных вам необходимо произвести дополнительные проверки или внести исправления. Ему соответствует параметр команды RESTORE WITH RESTRICTED_USER;
- **Restore the database files as** (Восстановить файлы базы данных как) — очень важный параметр, который позволяет определить новый путь для восстанавливаемых файлов баз данных. Без него не обойтись, например, в тех ситуациях, когда вы восстанавливаете базу данных на другой сервер, на котором конфигурация дисков выглядит по-другому. Этому флагку в команде RESTORE соответствует параметр MOVE, например:

```
RESTORE DATABASE db1 FROM DISK = 'D:\SQLbackups\BackupFile1.bak'  
    WITH MOVE 'db1' TO 'D:\db1.mdf', MOVE 'db1_log' TO 'D:\db1_log.mdf';
```

Здесь db1 и db1_log — это логические названия файлов базы данных и журнала транзакций соответственно, а 'D:\db1.mdf' и 'D:\db1_log.mdf' — это новые места для файлов, которые будут восстановлены с резервной копии;

- **Recovery state** (Состояние восстановления) — еще один важнейший параметр, который определяет, будет ли база данных открыта для пользователей после окончания восстановления с носителя. В вашем распоряжении три варианта:
 - **WITH RECOVERY** — восстановление в обычном режиме. После окончания восстановления начнется процедура RECOVERY, все незавершенные транзакции будут отменены, и в итоге база данных будет открыта для пользователей. Этот параметр используется по умолчанию;
 - **WITH NORECOVERY** — после окончания процесса восстановления с носителя процедура RECOVERY не начнется. Базы данных останутся в нерабочем состоянии восстановления. Этот параметр используется тогда, когда после восстановления резервной копии вы хотите восстановить дополнительные копии, например, после восстановления полной резервной копии восстановить резервную копию журнала транзакций;

- **WITH STANDBY** — процедура RECOVERY начнется, но вся информация о всех отмененных незавершенных транзакциях будет записана в файл отмены (его нужно будет указать). В результате пользователи смогут обращаться к восстановленной базе данных для чтения (например, для создания отчетов), но в то же время сохраняется возможность применения следующих резервных копий журналов транзакций. Такое решение используется обычно только при применении автоматической доставки журналов на резервный сервер (*log shipping*).

Как и в случае с командой BACKUP, некоторые возможности команды RESTORE доступны только из кода Transact-SQL. Про некоторые из них (например, про возможность восстановления до метки транзакции или LSN) уже рассказано. Далее представлено еще несколько параметров, которые нельзя выбрать при помощи графического интерфейса:

- PAGE** — этот параметр позволяет указать определенные страницы в базе данных, которые будут восстанавливаться. Эта новая возможность SQL Server 2005 будет рассмотрена в разд. 6.7.2;
- CHECKSUM | NOCHECKSUM** — позволяет включить или отключить проверку контрольных сумм при восстановлении. По умолчанию такая проверка производится, а в случае выявления расхождений восстановление прекращается и выдается сообщение об ошибке;
- CONTINUE_AFTER_ERROR | STOP_ON_ERROR** — будет ли остановлено восстановление в случае обнаружения ошибок в контрольной сумме. По умолчанию установлен параметр **STOP_ON_ERROR**;
- MEDIANAME** — позволяет указать имя носителя, с которого производится восстановление. Используется только для дополнительных проверок;
- MEDIAPASSWORD** и **PASSWORD** — при помощи этих параметров вам потребуется указать пароли для носителя и резервной копии соответственно, которые были использованы при резервном копировании. Эти параметры также следует отнести к категории дополнительных проверок. Если вы производите восстановление резервной копии на другой сервер (по отношению к тому, на котором была создана резервная копия), то пароль указывать не нужно;
- PARTIAL** — определяет, что в ходе данного сеанса восстановления будет производиться восстановление только одной файловой группы (если резервное копирование производилось по файловым группам). Процедура восстановления базы данных по частям (т. е. по файловым группам) называется *piecemeal restore*;
- RESTART** — позволяет продолжить операцию восстановления с того момента, когда она была прервана (например, необходимо вставить следующий картридж в стриммер);

- REWIND | NOREWIND** — производить ли после окончания восстановления перемотку ленты в картридже или нет. По умолчанию используется значение **REWIND**, т. е. производить;
- STATS** — так же, как и для команды **BACKUP**, этот параметр определяет частоту появления информационных сообщений. По умолчанию информация о ходе восстановления выводится после восстановления приблизительно каждой 10% резервной копии;
- UNLOAD | NOUNLOAD** — выгружать картридж из стриммера после окончания восстановления или нет. По умолчанию используется значение **UNLOAD**, т. е. выгружать. **UNLOAD** включает в себя также и перемотку ленты на начало, поэтому вместе с параметром **REWIND** использоваться не может.

6.7. Специальные ситуации восстановления

6.7.1. Восстановление базы данных в оперативном режиме

Во всех предыдущих версиях SQL Server можно было выполнять восстановление базы данных, только отключив от нее всех пользователей. В SQL Server 2005 появилась новая возможность — восстановление на работающей базе данных. Другое название такого типа восстановления — **оперативное восстановление** (*online restore*).

Конечно, на практике обойтись совсем без ограничения доступа пользователей не удастся. При восстановлении на работающей базе данных вам в любом случае придется перевести в автономный режим (*offline*) тот файл или файловую группу, восстановление которого вы производите в данный момент. Остальные файлы или файловые группы могут оставаться в рабочем режиме.

Для восстановления на открытой базе данных предусмотрены и другие ограничения:

- резервное копирование на работающей базе данных может использоваться только для баз данных, которые работают в режиме восстановления **Full** или **Bulk-logged**;
- оперативное восстановление первого файла базы данных или первичной файловой группы (в которых находятся системные таблицы и карта размещения данных) производить нельзя.

Если это возможно, SQL Server автоматически применяет режим оперативного восстановления при восстановлении отдельных файлов, файловых групп и страницочном восстановлении (но не при обычном восстановлении всей базы данных). Если вы хотите запретить применение оперативного восстановления

и производить восстановление файлов, файловых групп и отдельных страниц в обычном автономном режиме, то можно перед восстановлением выполнить команду `BACKUP LOG WITH NORECOVERY`. Эта команда, которая обычно применяется только при использовании автоматической доставки журналов (*log shipping*), позволяет создать резервную копию журнала транзакций и перевести базу данных в специальное состояние `RESTORING`. В этом состоянии доступ к базе данных пользователей будет закрыт, а восстановление будет производиться только в автономном режиме.

Синтаксис команд для выполнения оперативного восстановления ничем не отличается от обычного. Например, чтобы в оперативном режиме восстановить файл `db1file2`, уже переведенный в автономный режим, можно использовать следующие команды:

```
RESTORE DATABASE db1 FILE = 'db1file2' FROM DISK =
'D:\SQLBackups\BackupFile1.bak' WITH NORECOVERY;
RESTORE LOG db1 FROM DISK = 'D:\SQLBackups\BackupLogFile1.bak';
```

6.7.2. Восстановление отдельных страниц базы данных

Еще одна новая возможность SQL Server 2005, связанная с восстановлением, — **восстановление отдельных страниц данных** (*page restore*). Теперь в некоторых ситуациях можно вместо восстановления всей базы данных или каких-то файлов, ограничиться восстановлением лишь отдельных страниц. Это позволит:

- сэкономить время;
- произвести восстановление в оперативном режиме, без отключения пользователей от базы данных. Недоступными для пользователей будут только восстанавливаемые страницы.

Чаще всего ошибки на страницах баз данных возникают из-за сбоев дисков или дисковых контроллеров. Поэтому перед таким восстановлением лучше убедиться, что с дисковой подсистемой сервера у вас все в порядке.

Восстановление отдельных страниц базы данных можно производить только при соблюдении следующих условий:

- вы используете редакцию Enterprise Edition;
- восстанавливаемые страницы не относятся к журналу транзакций, к служебным страницам базы данных и к полнотекстовым каталогам;
- база данных работает в режиме **Full** или **Bulk-logged**;
- файловые группы, к которым относятся восстанавливаемые страницы, доступны и на чтение, и на запись.

Как выглядит процедура восстановления отдельных страниц базы данных?

Порядок действий обычно такой:

1. Вначале вы обнаруживаете, что некоторые страницы в базе данных повреждены. Такую информацию можно получить при просмотре журналов событий SQL Server, при помощи команд DBCC (например, DBCC CHECKDB) и просто при помощи анализа сообщений, которые возвращаются клиентскому приложению. Сам SQL Server выявляет поврежденные страницы при помощи анализа контрольных сумм или контрольных бит.
2. Перед восстановлением вам нужно получить информацию о номерах поврежденных страниц и номерах файлов, в которых эти страницы находятся. Эта информация хранится в таблице suspect_pages базы данных msdb (она заносится в эту таблицу автоматически). Номера страниц находятся в столбце page_id, а номера файлов — в столбце file_id. Надо отметить, что в таблице suspect_pages не может быть более 1000 записей. По достижении этого предела запись в таблицу просто прекращается. Поэтому рекомендуется в случае физического повреждения баз данных после восстановления очистить эту таблицу.
3. Затем запускаете команду на восстановление базы данных, например:

```
RESTORE DATABASE db1 PAGE = '1:51, 1:52, 1:55' FROM DISK =
'D:\SQLBackups\BackupFile1.bak';
```

По умолчанию восстановление запускается в оперативном режиме, без отключения пользователей от базы данных. Больше 1000 поврежденных страниц восстанавливать нельзя.

6.7.3. Восстановление системных баз данных

В некоторых ситуациях может потребоваться произвести восстановление системной базы данных master, в которой хранится служебная информация всего сервера (например, информация о логинах, пользовательских базах данных, настройках сервера и т. п.).

Перед тем как производить восстановление базы данных master, подумайте об альтернативных возможностях. Если пострадала не только эта база данных, но и пользовательские базы данных, то возможно легче и надежнее будет просто переустановить весь сервер, а затем восстановить пользовательские базы данных с резервных копий. Если повреждена база данных master, а пользовательские базы данных не пострадали, то можно думать о том, чтобы переустановить сервер или перестроить базу данных master, а пользовательские базы данных присоединить. Такой вариант будет наиболее надежным.

Восстановление базы данных `master` отличается от восстановления обычных баз данных некоторыми особенностями:

- производить восстановление базы данных `master` можно только после перезапуска сервера в однопользовательском режиме. Проще всего сделать это, запустив SQL Server из командной строки. Для этого нужно перейти в каталог, в котором находится файл `sqlservr.exe` (по умолчанию это `C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn`), а затем выполнить команду:

```
sqlservr.exe -m
```

- если база данных `master` повреждена, то сервер вполне может не запуститься. В этом случае, чтобы все-таки можно было запустить сервер и произвести процедуру восстановления, нужно перестроить базу данных `master`. При перестроении база данных `master` возвращается к своему исходному состоянию (когда сервер был только что установлен). В предыдущих версиях SQL Server для перестройки базы данных `master` использовалась специальная утилита `rebuildm`. В SQL Server 2005 для этой цели используется программа установки SQL Server;
- для базы данных `master` доступен только один тип резервного копирования — полное резервное копирование всей базы данных. Поэтому восстановить вы можете только всю базу данных `master` целиком. Резервное копирование и восстановление журналов транзакций, а также любые другие операции восстановления (файлов, файловых групп, отдельных страниц и т. п.) для этой базы данных не предусмотрены;
- после восстановления базы данных `master` сервер автоматически перезагрузится.

После того как восстановление базы данных `master` завершится, очень рекомендуется проверить, не возникло ли каких-то проблем на SQL Server. Могут обнаружиться проблемы:

- с логинами. Для проверки можно использовать хранимую процедуру `sp_validatelogins`;
- с пользователями баз данных. Проверку можно произвести при помощи команды:

```
sp_change_users_login @Action = 'Report';
```

- со списком баз данных на сервере. Если какой-то базы данных в списке нет, но файлы ее остались на диске, эту базу данных можно заново присоединить к серверу.

Если вы произвели перестройку базы данных `master`, то после завершения восстановления этой базы данных обязательно нужно произвести восстанов-

ление баз данных `model` и `msdb`. В остальном, резервное копирование и восстановление этих баз производится так же, как и пользовательских.

Произвести резервное копирование базы данных `tempdb` невозможно. Поскольку эта база данных создается заново при каждом запуске SQL Server, то восстанавливать ее не нужно.

Задание для самостоятельной работы 6.1. Резервное копирование и восстановление базы данных

Задание:

1. Переведите базу данных `AdventureWorks` в режим восстановления **Full**.
2. Создайте на диске C: каталог `Backup` и произведите в него полное резервное копирование базы данных `AdventureWorks`. Файл резервной копии должен называться `AdventureWorksFull.bkp`.
3. Проведите разностное резервное копирование базы данных `AdventureWorks`. Файл резервной копии должен называться `AdventureWorksDiff.bkp`.
4. Проведите резервное копирование журнала транзакций базы данных `AdventureWorks`. Файл резервной копии должен называться `AdventureWorksLog.bkp`.
5. Произведите последовательное восстановление всех созданных вами резервных копий. При этом:
 - восстановление должно производиться для новой базы данных `AdventureWorks1`;
 - файлы этой базы данных должны находиться в корневом каталоге диска C:.

Решение:

В данном решении используются только команды Transact-SQL. Однако все действия можно выполнить и средствами графического интерфейса SQL Server Management Studio.

К пункту 1 задания — перевод базы данных `AdventureWorks` в режим восстановления **Full**:

1. Соответствующая команда может выглядеть так:

```
ALTER DATABASE AdventureWorks SET RECOVERY FULL;
```

К пункту 2 задания — проведение полного резервного копирования базы данных AdventureWorks:

1. Команды на проведение резервного копирования в соответствии с поставленными условиями могут выглядеть так:

```
USE master;
GO
BACKUP DATABASE AdventureWorks TO DISK =
N'c:\Backup\AdventureWorksFull.bkp' WITH NOFORMAT, NOINIT, NAME =
N'AdventureWorks-Full Database Backup';
GO
```

К пункту 3 задания — проведение разностного резервного копирования:

1. Код для выполнения разностного резервного копирования может быть следующим:

```
BACKUP DATABASE AdventureWorks TO DISK =
N'C:\Backup\AdventureWorksDiff.bkp' WITH DIFFERENTIAL, NOFORMAT,
NOINIT, NAME = N'AdventureWorks-Differential Backup';
```

К пункту 4 задания — проведение резервного копирования журнала транзакций:

1. Соответствующий код может быть таким:

```
BACKUP LOG AdventureWorks TO DISK = N'C:\Backup\AdventureWorksLog.bkp'
WITH NOFORMAT, NOINIT, NAME = N'AdventureWorks-Log Backup';
```

К пункту 5 задания — восстановление резервных копий в другую базу данных:

1. Соответствующий код может быть таким:

```
USE master;
GO
RESTORE DATABASE AdventureWorks1 FROM DISK =
N'c:\Backup\AdventureWorksFull.bkp' WITH NORECOVERY,
MOVE N'AdventureWorks_Data' TO N'C:\AdventureWorks_Data.mdf',
MOVE N'AdventureWorks_Log' TO N'C:\AdventureWorks_Log.ldf';
GO
RESTORE DATABASE AdventureWorks1 FROM DISK =
N'C:\Backup\AdventureWorksDiff.bkp' WITH NORECOVERY;
GO
RESTORE LOG AdventureWorks1 FROM DISK =
N'C:\Backup\AdventureWorksLog.bkp' WITH RECOVERY;
```



ГЛАВА 7

Средства обеспечения отказоустойчивости SQL Server 2005

В предыдущей главе были рассмотрены вопросы, связанные с резервным копированием и восстановлением баз данных SQL Server 2005. Однако в некоторых ситуациях кроме выполнения резервного копирования необходимо дополнительно повысить отказоустойчивость. В SQL Server 2005 для этого можно использовать:

- кластер;
- зеркальное отображение баз данных;
- автоматическую доставку журналов (*log shipping*).

Эти возможности и будут рассмотрены далее в этой главе.

7.1. Работа SQL Server 2005 в кластере

7.1.1. Преимущества кластеров

Еще несколько лет назад SQL Server, работающий в кластере, относился скорее к разряду теории. В настоящее время ситуация изменилась. На момент написания этой книги автору было известно несколько кластерных систем с SQL Server, работающих на предприятиях Санкт-Петербурга. Тем не менее и сейчас кластеры не относятся к разряду распространенных решений.

По опыту автора, у многих администраторов SQL Server, которые не сталкивались с кластерами на практике, существуют некоторые заблуждения относительно их возможностей. Поэтому в этом разделе мы постараемся остановиться на ситуациях, в которых могут применяться кластеры, на преимуществах и недостатках кластерных решений.

Первое, что нужно сказать о кластерах для SQL Server — они не могут применяться для повышения производительности! Часто встречаются статьи, которые рассказывают, как десятки и сотни компьютеров, объединенных в кластер, совместно работают над какой-то задачей: раскрывают шифры, моделируют различные явления для научных исследований, играют в шахматы и т. п. Однако в этом случае речь идет совсем о других кластерах. Если вы создадите кластер для SQL Server, то вы не сможете получить никакого выигрыша в производительности (а напротив, немножко проиграете). Единственный выигрыш, который дает кластер SQL Server, — это выигрыш в отказоустойчивости. Возможно, при реализации кластера вы получите также выигрыш с точки зрения гибкости серверной инфраструктуры: отключить для обслуживания какой-то сервер, используемый на предприятии, будет проще.

Второй момент, который необходимо отметить: кластеры — это вовсе не лекарство от всех болезней. Кластеры не защитят вас:

- от ошибок пользователей;
- от проблем с RAID-массивом;
- от некорректных действий разработчиков приложения;
- от ошибок в программном обеспечении самого SQL Server 2005.

Фактически кластеры решают только проблемы с отказом некоторых аппаратных подсистем сервера: оперативной памяти, процессора, материнской платы, сети, да и то не во всех случаях. При использовании кластеров восстановление работоспособности приложения в случае такого сбоя будет производиться намного быстрее и в автоматическом режиме.

Важной новой возможностью SQL Server 2005 стало то, что в кластере теперь может работать не только сам SQL Server, но и Analysis Services (базы данных OLAP), и служба Microsoft Search (полнотекстовые запросы).

7.1.2. Терминология и варианты конфигурации кластера

Физически **кластер** — это два или более компьютера (близких по параметрам оборудования), которые подключены к внешнему RAID-массиву. У этого внешнего RAID-контроллера обязательно должна быть общая шина SCSI (*shared SCSI bus*), которая обеспечивает подключение к нему более одного сервера одновременно. Также на серверах, которые входят в кластер, желательно установить по дополнительному сетевому адаптеру, который будет использоваться только для обмена информацией между членами кластера. На каждом сервере должно быть установлено программное обеспечение Windows Server 2003 Enterprise Edition или Datacenter Edition, а также должна

быть настроена служба Clustering Services. У каждого сервера есть свое имя, но пользователи при обращении к службе, работающей в кластере (например, SQL Server), видят не какое-то из этих имен, а имя третьего компьютера — виртуального сервера, работу которого и защищает кластер.

Теперь остановимся на некоторых терминах, которые используются для кластеров:

- **узел (node)** — это физический компьютер, который входит в кластер. Кластеры могут состоять из 2, 4, 8 или 16 узлов;
- **виртуальный сервер (virtual server)** — это то, к чему обращаются пользователи при работе с приложением, установленным на кластер. Виртуальный сервер соответствует кластеру, а не одному физическому компьютеру. Например, в вашем кластере могут быть узлы (т. е. компьютеры) с именами NodeA и NodeB, которые будут обеспечивать работу виртуального сервера с именем SQL1. При выходе из строя одного узла в кластере работоспособность виртуального сервера будет обеспечивать другой узел;
- **основной узел (primary node)** — это тот физический сервер, который в обычном режиме выполняет все задачи виртуального сервера, обслуживая запросы пользователей;
- **вторичный узел (secondary node)** — это физический сервер, в обычном режиме выполняющий роль запасного. Служба Clustering Services передает на него информацию о всех процессах, которые работают на основном сервере. В случае выхода из строя основного сервера вторичный узел примет на себя всю нагрузку виртуального сервера;
- **переключение узлов (failover)** — это смена ролей основного и вторичного узла, когда нагрузка перемещается на вторичный узел. Обычно происходит в автоматическом режиме при выходе из строя основного узла, но такое переключение можно произвести и вручную.

Кластер может работать в двух вариантах конфигурации: активный/пассивный и активный/активный.

Конфигурация **активный/пассивный (active/passive)** используется в большинстве случаев. В этом случае в обычном режиме нагрузку со стороны пользователей обслуживает только основной узел. Вторичный узел выполняет роль запасного и не несет какой-то нагрузки. Такая конфигурация является наиболее надежной и удобной, но при этом вторичный сервер будет фактически простоявать.

Чтобы он не стоял без дела, можно настроить конфигурацию **активный/активный (active/active)**. В этой конфигурации два узла, входящие в кластер, обслуживают два виртуальных сервера — каждый со своей задачей. При этом для одной задачи первый сервер выполняет роль основного, а вто-

рой сервер — вторичного. А для второй задачи второй сервер выполняет роль основного, а первый сервер — вторичного. При выходе из строя любого из серверов, входящих в кластер, всю нагрузку принимает на себя оставшийся сервер.

Такое решение является более экономичным, но одновременно более сложным и менее надежным. Настоятельно рекомендуется проверить, сможет ли один сервер справляться одновременно с двумя задачами в случае необходимости.

7.1.3. Установка SQL Server 2005 в кластер

Установка SQL Server 2005 в кластер должна начинаться с настройки кластера в операционной системе. Кластер можно создать только в корпоративных редакциях Windows 2000 и Windows 2003: Windows 2000 Advanced Server и Datacenter Server, Windows Server 2003 Enterprise Edition и Datacenter Edition. При этом максимальное количество узлов в кластере зависит от редакции SQL Server. SQL Server 2005 Standard Edition поддерживает максимум два узла в кластере. Редакции Enterprise Edition и Developer Edition поддерживают кластер из стольких узлов, сколько поддерживает операционная система.

Рекомендуется использовать для кластера два идентичных с точки зрения оборудования сервера. Если такой возможности нет, нужно постараться, чтобы на них было одинаковое количество процессоров и оперативной памяти. Если это количество разное, то использование "лишних" процессоров и оперативной памяти придется запретить (это можно сделать как средствами операционной системы, так и средствами SQL Server после окончания установки). Объединить в кластер 32- и 64-разрядную версию SQL Server 2005 невозможно.

После установки и настройки кластера в операционной системе необходимо подготовиться к установке виртуального сервера SQL Server 2005 в кластер. Для этого необходимо выполнить следующие действия:

- перевести в автоматический режим запуска **Службы криптографии** (Windows Cryptographic Service Provider) и **Планировщик заданий** (Task Scheduler);
- установить службу **Координатор распределенных транзакций** (Distributed Transaction Coordinator), если она не установлена, и перевести ее в автоматический режим запуска. Эта служба должна быть приведена в рабочее состояние до установки SQL Server 2005 в кластер;
- убедиться, что ни один из узлов создаваемого кластера не является контроллером домена. SQL Server 2005 не может работать в кластере на контроллере домена;

- отключить протокол NetBIOS на всех сетевых адаптерах, которые используются для обмена служебной информацией кластера.

После этого можно начинать обычную установку SQL Server 2005, но на экране выбора компонентов нужно установить для самого ядра SQL Server 2005 и для Analysis Services флагок **Create a failover cluster** (Создать отказоустойчивый кластер). Далее потребуется ввести информацию об имени виртуального сервера, IP-адресе, используемом общем диске и т. п.

Администрирование SQL Server 2005, работающего в кластере, практически не отличается от администрирования обычного сервера (только при подключении вам придется указывать имя виртуального сервера, а не физических узлов). Все операции, специфичные для кластера (настройки, смена ролей компьютеров и т. п.), выполняются средствами операционной системы, а не SQL Server, и поэтому здесь они рассматриваться не будут.

7.2. Автоматическая доставка журналов

7.2.1. Что такое "автоматическая доставка журналов"

Автоматическая доставка журналов (*log shipping*) — это еще одна технология, которая предназначена повысить отказоустойчивость вашего приложения и скорость восстановления. Принцип ее работы очень прост: на одном сервере регулярно (например, с интервалом в несколько минут) производится резервное копирование журналов транзакций. Затем эти копии журналов автоматически передаются на другой сервер, где существует копия этой базы данных, и автоматически там восстанавливаются. В результате на втором сервере у вас создается копия рабочей базы данных, которая будет синхронизироваться с рабочей базой с разницей в несколько минут.

Главное назначение данной резервной копии — это, конечно, обеспечение отказоустойчивости. В случае отказа рабочего сервера в вашем распоряжении будет резервная копия, отстающая всего на несколько минут. Правда, в отличие от кластера, автоматической смены ролей не произойдет. Вам придется вручную открыть доступ пользователям на сервер с копией базы данных. При этом еще потребуется "объяснить" клиентским приложениям, что теперь они должны обращаться на новый сервер. Это можно сделать множеством разных способов (в зависимости от текущей ситуации): поменять IP-адрес или имя запасного сервера, изменить записи на сервере DNS, использовать псевдонимы на компьютерах пользователей, перенастроить источники данных ODBC и т. п.

Вполне можно представить себе ситуацию, когда на вашем предприятии работает несколько рабочих серверов, и при этом один резервный сервер при

помощи автоматической доставки журналов поддерживает резервные копии баз данных каждого из этих серверов. Любой рабочий сервер в случае выхода его из строя можно будет заменить на резервный.

Кроме отказоустойчивости, Microsoft предлагает использовать резервный сервер в такой конфигурации также и для снятия нагрузки с основного сервера. Дело в том, что резервные копии журналов транзакций можно восстанавливать на сервер в так называемом режиме *STANDBY* (см. разд. 6.3.2). В этом случае после восстановления каждой копии журнала транзакций резервная база данных будет автоматически открываться пользователям на чтение. Теоретически ее при этом можно использовать для обслуживания запросов пользователей, которые не изменяют данные (например, для генерации отчетов). На практике же использовать эту возможность вряд ли удастся. Причина проста: для восстановления журналов транзакций необходимо, чтобы в базе данных не было пользовательских подключений. Это ставит вас перед двумя не самыми лучшими вариантами: либо разрешить пользователям работать в базе данных постоянно (и в это время восстановление журналов транзакций производиться не будет, т. е. расхождение между рабочей и резервной базами данных будет накапливаться), либо принудительно отключать пользователей для восстановления журналов.

Несмотря на то, что графический интерфейс для настройки автоматической доставки журналов появился еще в SQL Server 2000 (а в предыдущих версиях можно было настроить ее вручную при помощи пакетов SQL Server Agent), автор еще не встречал предприятия, где автоматическая доставка журналов применялась бы для рабочих серверов. Если возникает такая потребность, то администраторы стараются использовать вместо доставки журналов репликацию. В основном это связано с тем, что репликация:

- более привычна;
- более функциональна (можно настроить больше параметров, чем для доставки журналов);
- меньше влияет на работу пользователей на резервном сервере.

В то же время доставка журналов значительно проще в настройке и в администрировании, чем репликация, и ее вполне можно рассматривать в качестве одного из возможных вариантов повышения отказоустойчивости сервера.

7.2.2. Терминология доставки журналов

Доставка журналов предусматривает три роли для серверов. Эти роли могут играть три отдельных сервера, а можно совместить их все, например, на одном сервере.

- Основной сервер** (*primary server*, другое название — сервер-источник) — тот сервер, на котором работает рабочая база данных. На этом сервере бу-

дут создаваться резервные копии журналов транзакций. База данных, для которой настраивается доставка журналов, называется **основной базой данных** (*primary database*).

- **Вторичный сервер** (*secondary server*, другое название — сервер-получатель) и **вторичная база данных** (*secondary database*, база данных получателя) — это, соответственно, тот сервер и та база данных, на которых будет производиться восстановление журналов транзакций.
- **Сервер мониторинга** (*monitor server*) — это сервер, который будет отслеживать доставку журнала и в случае каких-то проблем оповещать о них администратора. Сервером мониторинга можно при желании сделать основной или вторичный сервер.

Термин *failover* означает то же, что и для кластера, — переключение ролей сервера, когда вторичный сервер занимает место основного. Основной сервер при этом может стать вторичным или вообще может быть выведен из сети, например, для ремонта.

7.2.3. Настройка доставки журналов

Настроить доставку журналов можно как при помощи графического интерфейса Management Studio, так и при помощи команд Transact-SQL. Команды Transact-SQL здесь рассматриваться не будут, поскольку, во-первых, это заняло бы много места и времени, а во-вторых, необходимый код можно легко сгенерировать автоматически при помощи кнопки **Script Configuration** (Отскриптовать конфигурацию) на графическом экране Management Studio.

В предыдущей версии SQL Server для настройки доставки журналов использовался Database Maintenance Plan Wizard — мастер настройки планов обслуживания баз данных. В SQL Server 2005 планы обслуживания баз данных не имеют к доставке журналов никакого отношения. Настройка доставки журналов производится из вкладки **Transaction Log Shipping** (Доставка журналов транзакций) свойств базы данных (рис. 7.1). Эту вкладку можно открыть также из контекстного меню базы данных при помощи команды **Tasks | Ship Transaction Logs** (Задачи | Доставлять журналы транзакций).

Остановимся на возможностях настройки доставки журналов:

- флагок **Enable this as a primary database in a log shipping configuration** (Включить как основную базу данных в конфигурации доставки журналов) фактически включает доставку журналов для данной базы данных. Если база данных работает в режиме восстановления **Simple** (см. разд. 4.5), то попытка установить этот флагок приведет к появлению предупреждающего сообщения. Использование доставки журналов для баз данных, которые работают в режиме восстановления **Bulk-logged**, также не реко-

мендуется. Лучше всего использовать доставку журналов только с режимом **Full**;

- нажатие на кнопку **Backup Settings** (Настройки резервного копирования) приведет к открытию еще одного диалогового окна **Transaction Log Backup Settings** (Настройки резервного копирования журнала транзакций), в котором вы сможете настроить параметры резервного копирования журнала транзакций основной базы данных. Следующие параметры настраиваются именно из этого окна;

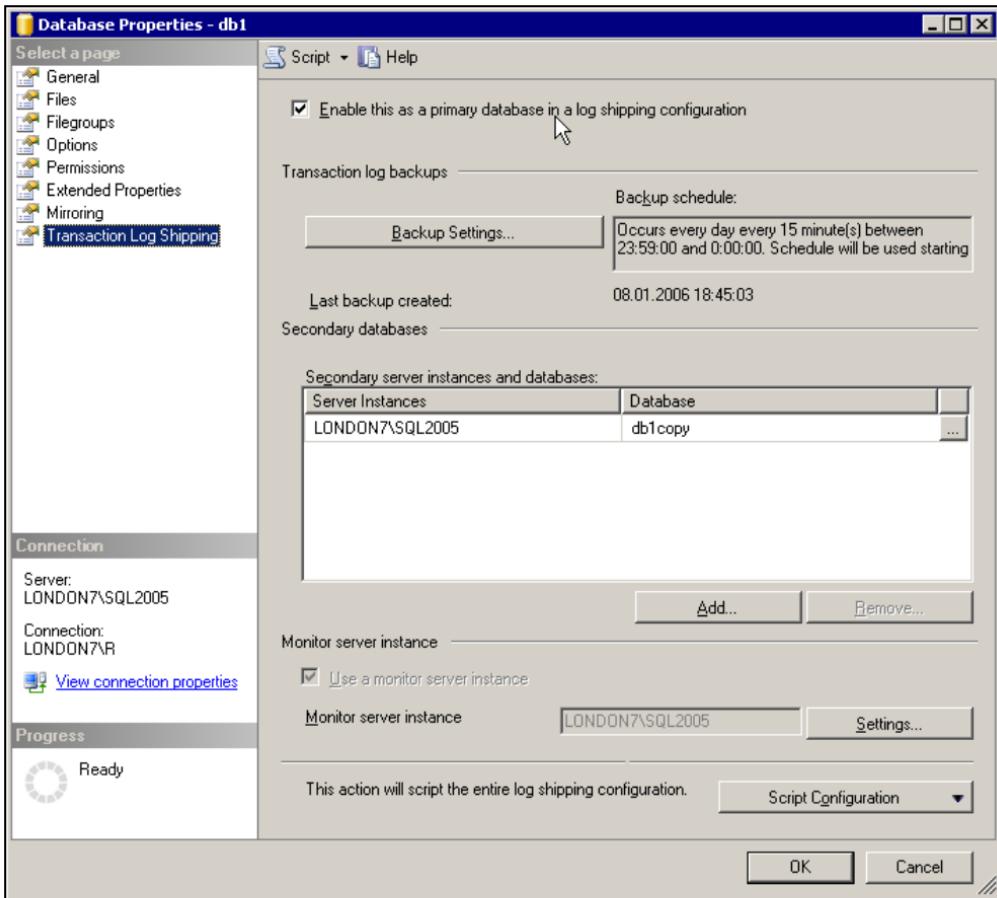


Рис. 7.1. Экран настройки доставки журналов

- **Network path to backup folder** (Сетевой путь к каталогу резервного копирования) и **Local path to the folder** (Локальный путь к каталогу) — соответственно, сетевой и локальный пути к каталогу, в который будут помещаться резервные копии журналов транзакций. Служба SQL Server Agent

вторичного сервера будет забирать файлы резервных копий из этого каталога. Если вы решили помещать резервные копии сразу на сетевой сервер, то поле с локальным путем можно оставить пустым.

Обратите внимание, что если основной сервер работает от имени локальной учетной записи, он не сможет обращаться к сетевым каталогам на другом компьютере. В этом случае вам обязательно потребуется указать локальный путь для резервных копий журналов транзакций;

- **Delete files older than** (Удалять файлы, старше чем) — этот параметр позволяет указать срок (дату), по истечении которого файлы журналов транзакций будут удалены (по умолчанию 3 дня). Даже успешно скопированные и восстановленные файлы журналов транзакций все равно будут оставаться в исходном каталоге в течение указанного времени. То, что они не удаляются сразу, связано с тем, что для одного первичного сервера можно настроить доставку журналов на несколько вторичных;
- **Alert if no backup occurs within** (Оповещать, если резервное копирование не происходит в течение) — для этого параметра по умолчанию используется значение 1 час. Он используется для создания предупреждения **Log Shipping Primary Server Alert** (Предупреждение основного сервера поставки журналов), которое можно увидеть под контейнером **Alerts** (Предупреждения) в контейнере **SQL Server Agent**. По умолчанию это предупреждение не назначается ни одному оператору, а, следовательно, увидеть, что оно сработало, можно только при просмотре логов SQL Server Agent (или таблиц истории доставки журналов). Подробно про работу с предупреждениями и операторами будет рассказываться в разд. 8.1.7;
- **Backup job** (Задание резервного копирования) — при помощи этой группы элементов управления можно просмотреть параметры создаваемого задания SQL Server Agent, которое и будет выполнять резервное копирование журналов транзакций. При помощи кнопки **Edit job** (Изменить задание) можно изменить параметры этого задания, а при помощи флажка **Disable this job** (Отключить это задание) — на время отключить задание. Подробно про работу с заданиями будет рассказываться в разд. 8.1.3.

После того как вы закончите настройку параметров резервного копирования журналов транзакций и нажмете кнопку **OK** на вкладке **Transaction Log Shipping** окна свойств базы данных, вам потребуется настроить параметры восстановления резервных копий на сервере-получателе. Для этого надо нажать кнопку **Add** (Добавить) под списком **Secondary server instances and databases** (Вторичные экземпляры серверов и базы данных) на той же вкладке. В открывшемся окне **Secondary database settings** (Параметры вторичных баз данных) вам потребуется определить следующие параметры:

- **Secondary server instance** (Вторичный экземпляр сервера) — тот сервер, на котором будут восстанавливаться резервные копии журнала транзакций;

- **Secondary database** (Вторичная база данных) — та база данных, для которой будет производиться восстановление копий журналов транзакций. Можно не только выбрать имя существующей базы данных, но и вписать новое имя. В этом случае базу данных можно будет создать автоматически.

В том же окне при помощи вкладки **Initialize Secondary Database** (Инициализировать вторичную базу данных) вам придется определиться, как именно будет создана вторичная база данных. В вашем распоряжении три варианта:

- **Yes, generate a full backup of the primary database** (Да, сгенерировать полную резервную копию основной базы данных) — будет автоматически проведено полное резервное копирование исходной базы данных, и эта резервная копия будет восстановлена на указанном вами сервере под указанным именем. При помощи кнопки **Restore options** (Восстановить параметры) можно будет определить местонахождение файлов базы данных и журналов транзакций создаваемой базы;
- **Yes, restore an existing backup of the primary database** (Да, восстановить существующую резервную копию основной базы данных) — этот вариант используется в ситуации, когда у вас уже есть полная резервная копия основной базы данных и нет необходимости производить ее резервное копирование заново. В этом случае вам потребуется указать сетевой путь к файлу с полной копией;
- **No, the secondary database is initialized** (Нет, вторичная база данных уже инициализирована) — при выборе этого варианта вам потребуется самостоятельно позаботиться о создании вторичной базы данных и ее исходной синхронизации с основной.

Самый простой и надежный вариант — первый, когда копия основной базы данных создается автоматически. Второй и третий варианты имеет смысл выбирать только в специальных ситуациях, например, когда основная база данных очень большая или когда падение производительности, вызванное резервным копированием, недопустимо.

На вкладке **Copy Files** (Копирование файлов) вы определяете параметры копирования файлов резервных копий журналов транзакций:

- **Destination folder to copied files** (Каталог назначения для скопированных файлов) — удобнее всего здесь указать просто локальный каталог на вторичном сервере. Но если это противоречит каким-то правилам безопасности, можно указать и сетевой каталог на файл-сервере;
- **Delete copied files after** (Удалять скопированные файлы после) — скопированные файлы будут удалены только через указанное здесь количество часов (даже если восстановление этих резервных копий прошло успешно). По умолчанию задано также 3 суток (72 часа);

- **Copy job** (Задание копирования) — при помощи этого параметра можно узнать имя создаваемого задания SQL Server Agent, которое будет заниматься копированием файлов резервных копий журналов транзакций, настроить расписание резервного копирования, а также при необходимости отключить это задание.

На вкладке **Restore Transaction Log** (Восстановление журналов транзакций) вы можете настроить параметры восстановления резервных копий журналов транзакций:

- **Database state when restoring databases** (Состояние базы данных при восстановлении) — это очень важный параметр, который определяет, будет ли база данных открываться для пользователей. В вашем распоряжении два варианта:

- **No recovery mode** (Режим без восстановления работоспособности) — база данных открываться для пользователей не будет;
- **Standby mode** (Режим "горячей готовности") — база данных будет открыта в режиме "только чтение". Однако понятно, что если к базе данных подключены пользователи, восстановление журналов транзакций производиться не будет. Поэтому вы можете или смириться с задержками при восстановлении, или принудительно отключать пользователей при восстановлении, установив флажок **Disconnect users in the database when restoring backups** (Отсоединять пользователей от базы данных при восстановлении резервных копий).

Оба варианта достаточно неудобны. Возможно, в некоторых ситуациях имеет смысл открывать доступ к пользователям в режиме STANDBY, настроив расписание для восстановления таким образом, чтобы восстановление происходило только во внерабочее время. Однако при использовании такого решения мы теряем главное преимущество доставки журналов — быструю синхронизацию основной и вторичной баз данных;

- **Delay restoring backups at least** (Отложить восстановление базы данных по крайней мере на) — этот параметр дает возможность определить задержку перед восстановлением резервной копии. По умолчанию задано без задержки;
- **Alert if no restore occurs within** (Предупредить, если восстановления не производятся в течение) — при помощи этого параметра можно указать пороговое время ожидания для восстановления резервных копий. Если в течение указанного времени (по умолчанию оно равно 45 минутам) восстановление по каким-то причинам произведено не будет, сработает предупреждение. Как и предупреждение резервного копирования, это предупреждение можно найти в контейнере **SQL Server Agent | Alerts**. Оно называется **Log shipping Secondary Server Alert** (Предупреждение вто-

ричного сервера доставки журналов). Для того чтобы оно действительно оповещало администратора (например, при помощи сетевого сообщения или письма по электронной почте), ему нужно будет назначить оператора;

- **Restore Job** (Задание восстановления) — можно выбрать имя и расписание для задания, которое будет заниматься восстановлением резервных копий журналов транзакций.

После того как настройка вторичного сервера будет завершена, вам останется только вернуться в окно свойств основной базы данных и настроить параметры сервера мониторинга. В принципе можно обойтись и без сервера мониторинга, но производить диагностику поставок журналов без информации, которая накапливается в таблицах этого сервера, намного сложнее. Включить такой сервер можно при помощи флагка **Use a monitor server instance** (Использовать экземпляр сервера мониторинга). Затем при помощи кнопки **Settings** (Параметры) вам потребуется настроить дополнительные параметры сервера мониторинга:

- **Monitor server instance** (Экземпляр сервера мониторинга) — этот параметр позволяет выбрать сервер, который будет отслеживать доставку журналов;
- **Monitor connections** (Отслеживать соединения) — определяет, как именно задания, которые выполняют резервное копирование, копирование по сети и восстановление, будут "отчитываться" перед сервером мониторинга (т. е. заносить информацию в его таблицы):
 - **By impersonating the proxy account of the job** (При помощи имперсонации учетной записи-прокси задания) — подключение будет производиться от имени специальной учетной записи-прокси. Ее можно определить из вкладки **Job system** (Система заданий) свойств SQL Server Agent на том сервере, на котором выполняется задание. По умолчанию используется учетная запись, от имени которой работает служба SQL Server Agent.
- Если переключатель **Monitor connections** установлен в это положение, то этой учетной записи нужно предоставить права на запись информации в таблицы на сервере мониторинга;
- **Using the following SQL Server login** (Использование следующего логина SQL Server) — можно явно указать логин SQL Server, который будет использоваться для подключения к серверу мониторинга. Конечно, вам потребуется создать соответствующий логин на сервере мониторинга и предоставить ему соответствующие права;
- **Delete history after** (Удалять историю после) — через сколько дней будут удаляться старые записи из таблиц мониторинга. По умолчанию определено, что через четыре дня;

- **Alert job** (Задание для оповещения) — этот параметр позволяет настроить задание, которое будет выполняться на сервере мониторинга и опрашивать основной и резервный серверы на предмет появления каких-либо проблем с доставкой журналов. Обратите внимание, что опрос по умолчанию производится каждые две минуты, что может повлиять на производительность работы сервера мониторинга.

7.2.4. Мониторинг доставки журналов

Если доставка журналов используется у вас на постоянной основе, то, скорее всего, вам потребуется получать информацию о том, успешно ли она работает.

Самый простой и очень удобный способ получения информации о доставке журналов — использование встроенного отчета Management Studio. Получить отчет о доставке журналов можно так:

- открыть Management Studio и в дереве **Object Explorer** выделить контейнер с именем сервера. Это может быть основной сервер, вторичный сервер или сервер мониторинга: в отчете будет показана только информация о той части доставки журналов, которая выполняется на этом сервере;
- в окне **Document** (Документ) (в главном окне Management Studio) раскрыть список рядом с кнопкой **Report** (Отчет) и выбрать тип отчета **Transaction Log Shipping Status** (Состояние доставки журнала транзакций).

В результате будет создан и загружен в окно **Document** отчет с информацией о доставке журналов, аналогичный представленному на рис. 7.2.

Если вам нужна более подробная информация, то в вашем распоряжении история выполнения заданий резервного копирования, сетевого копирования и восстановления, а также таблицы мониторинга.

Просмотреть информацию об истории выполнения заданий доставки журналов можно так:

1. В окне **Object Explorer** выбрать контейнер **SQL Server Agent** для нужного сервера SQL Server Agent и раскрыть в нем контейнер **Jobs** (Задания). При настройке доставки журналов автоматически создаются четыре задания:
 - **LBackup** — задание для резервного копирования журналов транзакций. Оно создается на основном сервере;
 - **LSCopy** — задание для сетевого копирования созданных файлов резервных копий. Оно создается на вторичном сервере;

Status			Backup		Copy		Restore		
	Primary Database	Secondary Database	Time Since Last	Threshold	Alert Enabled	Time Since Last	Time Since Last	Latency of Last File	Three
Good	[LONDON7\SQL2005] [db1]		11 min	60 min	True				
Good	--[LONDON7\SQL2005] [db1copy]					26 min	11 min	15 min	45 mi

- data in this column is not available or not applicable for this server instance.

Рис. 7.2. Отчет о доставке журналов

- **LSRestore** — задание для восстановления созданных резервных копий. Оно также создается на вторичном сервере;
- **LSAlert** — задание для опроса основного и резервного сервера. Оно создается на сервере мониторинга.

2. Открыть контекстное меню для нужного задания и выбрать команду **View History** (Просмотреть историю). Откроется окно просмотра истории выполнения заданий, аналогичное представленному на рис. 7.3.

При помощи этого экрана можно получить информацию о том, когда выполнялось задание и с каким результатом. Для каждого этапа выполнения задания приводится подробная информация. Если произошел какой-то сбой, то при помощи окна истории можно получить информацию о причинах этого сбоя. История выполнения заданий наиболее полезна для диагностики и исправления ошибок.

Если вам нужны отчеты о доставке журналов в своем собственном формате, можно воспользоваться данными из таблиц и хранимых процедур мониторинга. Такая необходимость возникает редко, поэтому таблицы мониторинга (все они находятся в базе данных msdb на сервере мониторинга) и хранимые процедуры, которые упрощают к ним доступ, рассматриваться не будут. Просто перечислим их названия:

таблицы мониторинга:

```
log_shipping_monitor_alert, log_shipping_monitor_error_detail,
log_shipping_monitor_history_detail, log_shipping_monitor_primary,
log_shipping_monitor_secondary;
```

хранимые процедуры для доступа к таблицам мониторинга:

```
sp_help_log_shipping_monitor_primary,
sp_help_log_shipping_monitor_secondary,
sp_help_log_shipping_alert_job,
sp_help_log_shipping_primary_database,
sp_help_log_shipping_primary_secondary,
sp_help_log_shipping_secondary_database.
```

The screenshot shows the 'Log File Viewer - LONDON7\SQL2005' window. On the left, there's a tree view under 'Select logs' with 'Job History' expanded, showing several log shipping jobs like 'LSAlert_LONDON7\SQL2005', 'LSBackup_db1', etc. Below that is a 'Status' section with 'Last Refresh: 09.01.2006 19:59:48' and a 'View filter settings' link. A 'Progress' section shows 'Done (14 records)' with a checkmark icon. The main area is titled 'Log file summary: No filter applied' and contains a grid table with columns: Date, Step ID, Server, Job Name, Step Name, and Notification. The table lists 14 rows of log shipping events for 'LSBackup_db1'. At the bottom, a 'Selected row details:' section shows specific information for the first row: Date 09.01.2006 19:45:00, Log Job History [LSBackup_db1], Step ID 0, Server LONDON7\SQL2005, Job Name LSBackup_db1, Step Name [Job outcome], Duration 00:00:03, Sql Severity 0, Sql Message ID 0, and Operator Emailed.

Date	Step ID	Server	Job Name	Step Name	Notification
09.01.2006 19:45:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 19:45:00	1	LONDON7\SQL2005	LSBackup_db1	Log shippin...	
09.01.2006 19:45:00	1	LONDON7\SQL2005	LSBackup_db1	Log shippin...	
09.01.2006 19:45:00	1	LONDON7\SQL2005	LSBackup_db1	Log shippin...	
09.01.2006 19:45:00	1	LONDON7\SQL2005	LSBackup_db1	Log shippin...	
09.01.2006 19:45:00	1	LONDON7\SQL2005	LSBackup_db1	Log shippin...	
09.01.2006 19:45:00	1	LONDON7\SQL2005	LSBackup_db1	Log shippin...	
09.01.2006 19:30:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 19:15:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 19:00:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 18:45:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 18:30:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 18:15:00	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	
09.01.2006 18:00:01	0	LONDON7\SQL2005	LSBackup_db1	[Job outcome]	

Рис. 7.3. История выполнения задания

При необходимости создания своих отчетов можно просто просмотреть эти таблицы и информацию, возвращаемую хранимыми процедурами, и выбрать нужные для отчета данные.

7.2.5. Действия в случае сбоя основного сервера

Основное назначение доставки журналов — обеспечение отказоустойчивости. Если произошел сбой основного сервера, то вы можете просто предоставить пользователям для работы вторичную базу данных на другом сервере. Возобновить работу пользователей можно будет очень быстро, поскольку рабочая копия базы данных у вас уже есть (возможно, она на какое-то время отстает от основной базы данных, но повторный ввод данных за последние 15 минут или полчаса обычно не представляет больших проблем для пользователей).

Однако некоторые действия вам все-таки придется выполнить. Отметим, что все они выполняются вручную: автоматического переключения, как при использовании кластера, при доставке журналов не предусмотрено. План действий в случае сбоя основного сервера представлен далее:

1. Первое, которое нужно выполнить, — отключить все четыре задания, которые создаются в процессе настройки доставки журналов. Они были перечислены в предыдущем разделе. Если основной сервер вышел из строя, эти задания будут вам только мешать.

Отключить задания проще всего при помощи команды **Disable** (Отключить) из контекстного меню для объекта задания в **Object Explorer** (контейнер **SQL Server Agent | Jobs**). Отключенные задания помечаются в **Object Explorer** специальной красной меткой. Но правильнее будет отключить для каждого задания расписание. Для этого нужно открыть свойства задания, перейти на вкладку **Schedules** (Расписания), выделить нужное расписание (по умолчанию создается только одно) и нажать кнопку **Edit** (Изменить). Затем в свойствах расписания нужно снять флажок **Enabled** (Включено).

2. Затем нужно вручную скопировать и восстановить те резервные копии журналов транзакций, которые были созданы на основном сервере, но еще не были восстановлены на вторичном сервере. Просмотреть информацию о последнем восстановленном файле проще всего при помощи отчета о доставке журналов для вторичного сервера из Management Studio (*см. разд. 7.2.4*). Если вы отключили расписания у заданий, то резервное копирование и восстановление можно произвести при помощи заданий **LSCopy** и **LSRestore**. Вручную запустить задание можно при помощи команды **Start job** (Запустить задание) из его контекстного меню в **Object Explorer**.
3. Если основной сервер находится в рабочем состоянии, подумайте, не нужно ли вручную снять с него резервную копию последней части журнала

транзакций и также восстановить ее на вторичном сервере, чтобы не потерять никакие пользовательские данные. Если нужно запретить работу с основным сервером (например, на нем нужно будет произвести какие-то работы), то можно произвести резервное копирование остатка журнала с параметром NORECOVERY. В этом случае будет не только создана резервная копия журнала транзакций, но и будет закрыт доступ к базе данных для пользователей.

4. Следующее действие, которое вам потребуется выполнить, — перевести базу данных на вторичном сервере в рабочее состояние из режима NORECOVERY или STANDBY (в зависимости от параметров доставки журналов). Для этого достаточно выполнить команду RESTORE DATABASE с параметром WITH RECOVERY, например:

```
RESTORE DATABASE db1copy WITH RECOVERY;
```

Таким образом, вторичная база данных будет приведена в рабочее состояние. Теперь осталось самое сложное — объяснить клиентским приложениям, что они должны с этого момента обращаться на новый сервер. Как это сделать, зависит от того, как именно клиентское приложение обращается к серверу. В вашем распоряжении следующие варианты:

- если основной сервер больше не вернется в сеть предприятия (или на нем использовалась только база данных, выступавшая в роли основной), то, возможно, самым простым вариантом будет обычное переименование вторичного сервера (и, возможно, второй базы данных). После выполнения этой операции клиентские компьютеры станут обращаться к новому серверу автоматически;
- если для подключения пользователи используют полное доменное имя сервера (например, вида London7.nwtraders.msft), то, скорее всего, поможет изменение записей на сервере DNS. Если клиенты подключаются по IP-адресу (что бывает очень редко), то подойдет изменение IP-адреса вторичного сервера;
- если менять имя сервера и производить другие изменения в сетевой инфраструктуре нельзя, то придется заняться внесением изменений на клиентах. В зависимости от того, как устроены клиенты, можно использовать:
 - изменение имени сервера в клиентском приложении (если оно это позволяет), например, исправление файлов конфигурации;
 - применение псевдонима для SQL Server (псевдонимы придется создавать на каждом клиентском компьютере);
 - изменение свойств источника данных ODBC (также на каждом клиенте).

Если основной сервер нужно вернуть в строй, то проще всего восстановить старую конфигурацию так:

1. Произвести резервное копирование вторичной базы данных (которая в настоящее время выполняет функцию рабочей).
2. Восстановить эту резервную копию на основном сервере.
3. Перенацелить клиентские приложения обратно на основной сервер.
4. Удалить старые задания для резервного копирования, сетевого копирования, восстановления и мониторинга.
5. Настроить доставку журналов заново.

Проще всего сделать это вручную, хотя можно использовать для этой цели и специализированные хранимые процедуры.

7.2.6. Как отменить доставку журналов

Вполне может возникнуть ситуация, когда вам потребуется отменить доставку журналов и убрать всю связанную с ней информацию: задания SQL Server Agent, ведение мониторинга доставки журналов и т. п.

Проще всего сделать это средствами Management Studio: достаточно на той же вкладке **Transaction Log Shipping** основной базы данных снять флажок **Enable this as a primary database in a log shipping configuration**. После того как вы нажмете кнопку **OK**, будут автоматически удалены задания на резервное копирование, сетевое копирование и восстановление журналов транзакций. Будет произведено также удаление информации мониторинга доставки журналов для этой базы данных. Задание для мониторинга придется удалять вручную. Вторичная база данных останется в положении **NORECOVERY** или **STANDBY**, и решать ее судьбу придется вам самим. Для приведения ее в нормальное рабочее состояние нужно будет произвести повторное восстановление последней резервной копии журнала транзакций с параметром **RECOVERY**.

Другой вариант отмены доставки журналов — использовать специальные хранимые процедуры:

- `sp_delete_log_shipping_primary_secondary` и
`sp_delete_log_shipping_primary_database` — эти хранимые процедуры, которые выполняются на основном сервере, убирают с него все настройки доставки журналов для указанной вами базы данных;
- `sp_delete_log_shipping_secondary_database` — эта хранимая процедура выполняется на вторичном сервере. Она убирает всю информацию о доставке журналов с этого сервера вместе со вторичной базой данных.

7.3. Зеркальное отображение баз данных

7.3.1. Что такое "зеркальное отображение баз данных"

В SQL Server 2005 предусмотрено новое средство для повышения отказоустойчивости — **зеркальное отображение баз данных** (*database mirroring*). При использовании этого средства изменения, которые вносятся в базу данных на основном сервере (он называется *сервером-принципалом*), мгновенно (или почти мгновенно, в зависимости от выбранных вами параметров) отображаются в копии базы данных на другом сервере SQL Server 2005.

Зеркальное отображение баз данных имеет преимущества по сравнению и с применением кластеров, и с доставкой журналов.

Преимущества перед использованием кластера следующие:

- зеркальное отображение баз данных не требует применения специального оборудования;
- серверы, которые участвуют в зеркальном отображении баз данных, не обязательно находиться рядом друг с другом;
- в кластере серверы работают с одной физической базой данных, которая находится на внешнем RAID-массиве. Таким образом, выход из строя этого RAID-массива приведет к отказу всего кластера. В зеркальном отображении используются две отдельные копии базы данных, что повышает надежность работы.

По сравнению с доставкой журналов преимущества такие:

- переключение ролей в случае отказа основного сервера может производиться автоматически (при наличии **следящего сервера** (*witness server*));
- при применении зеркального отображения баз данных в некоторых ситуациях вам не потребуется вносить какие-либо изменения в сетевую инфраструктуру или в настройки клиентов. Клиенты при необходимости автоматически переключаются на использование зеркальной копии (это относится только к приложениям, которые используют SQL Native Client или .NET SQL Provider). Информацию о зеркальной копии можно указать в строке подключения (ее также может передать основной сервер при подключении к нему клиента);
- при использовании доставки журналов всегда есть какое-то отставание при синхронизации данных на серверах (по крайней мере, на несколько минут). Применение зеркального отображения баз данных позволяет избежать такой задержки и гарантирует идентичность копий данных на обоих серверах.

Для зеркального отображения можно использовать два режима:

- **синхронный режим (*synchronous mode*)** — при использовании этого режима транзакция не будет завершена, если она не прошла на обоих серверах. При использовании синхронного режима идентичность данных на двух серверах гарантируется. Однако скорость работы транзакций при этом может существенно замедлиться. Этот режим работы подразделяется еще на два:
 - **ориентированный на отказоустойчивость (*high-availability*)** — для этого режима обязательно использование следящего сервера. Этот режим гарантирует отказоустойчивость. В случае, когда основной сервер становится недоступным, следящий сервер автоматически меняет ролями сервер-принципал и сервер-зеркало. Если становится недоступным сервер-зеркало, то продолжается обычная работа. А если, например, стали одновременно недоступными и следящий сервер, и сервер-зеркало, то прекратится и работа базы данных на сервере-принципале — чтобы гарантировать, что сервер будет работать только в отказоустойчивом режиме;
 - **ориентированный на защиту данных (*high-protection*)** — в этом режиме можно обойтись без следящего сервера. Любая транзакция в этом режиме обязательно должна быть завершена на обоих серверах: так гарантируется идентичность данных на основной базе данных и на зеркале;
- **асинхронный режим (*asynchronous mode*, другое название — *high-performance mode* (режим высокой производительности))** — в этом случае транзакция вначале завершается на первом сервере, а затем информация о ней немедленно передается на второй сервер. Задержек при работе транзакций уже не будет, но данные между серверами могут синхронизироваться с небольшим отставанием.

К сожалению, на момент выпуска SQL Server 2005 разработчикам не удалось полностью отладить технологию зеркального отображения баз данных и сделать ее пригодной для использования на рабочих серверах. Об этом разработчики честно предупреждают в документации. Нет никакой гарантии, что зеркальное отображение будет работать надежно. Кроме этого, служба поддержки Microsoft имеет право отказаться от поддержки тех серверов, на которых реализовано зеркальное отображение баз данных, а сама возможность зеркального отображения по умолчанию отключена. Чтобы ее включить, необходимо использовать специальное отладочное средство — флаг трассировки 1400. Его нужно указать в параметрах запуска сервера (как это сделать, будет рассказано в разд. 7.3.3). Тем не менее технология зеркального отображения баз данных, которая фактически открывает преимущества кластеров для ши-

рого применения, очень перспективна, и, скорее всего, через определенное время она будет использоваться очень активно.

7.3.2. Терминология зеркального отображения баз данных

Терминология зеркального отображения баз данных существенно отличается от терминологии кластеров и доставки журналов:

- **база данных-принципал** (*principal database*) — это база данных, с которой в обычном режиме работают пользователи. Она является источником информации для зеркальной копии. Фактически, это аналог основной базы данных в конфигурации доставки журналов;
- **зеркальная база данных** (*mirror database*) — копия базы данных-принципала, которая получает информацию об изменениях и может использоваться в случае выхода из строя принципала для обеспечения отказоустойчивости (к ней будут подключаться пользователи);
- **следящий сервер** (*witness server*) — сервер, отдельный по отношению к серверам, на которых работают база данных-принципал и база данных-зеркало. Он осуществляет мониторинг зеркального отображения и в случае сбоя сервера-принципала может автоматически перевести в рабочее состояние базу данных-зеркало;
- **переключение ролей** (*role switching*) — процесс, при котором сервер-принципал и сервер-зеркало меняются ролями. Обычно такой процесс инициируется администратором;
- **переключение в случае отказа сервера-принципала** (*failover*) — процесс, когда зеркальная база данных приводится в рабочее состояние. Обычно этот процесс возникает автоматически, если с базой данных-принципалом возникли проблемы;
- **сессия зеркального отображения** (*mirroring session*) — этим словом называется работа базы данных-принципала, зеркальной базы данных и следящего сервера в режиме синхронизации баз данных, т. е. в режиме, когда зеркальное отображение включено. Весь набор участников сеанса зеркального отображения называется **кворумом зеркального отображения** (*mirroring quorum*).

7.3.3. Настройка зеркального отображения

Перед тем как настраивать зеркальное отображение, необходимо произвести определенную подготовку.

Во-первых, зеркальное отображение использует технологию *точек подключений по HTTP* (HTTP endpoints), которые работают только под Windows

Server 2003 или Windows XP. Поэтому настроить применение этой технологии под Windows 2000 не удастся.

Во-вторых, зеркальное отображение нельзя настроить между разными экземплярами SQL Server 2005 на одном физическом сервере: вам обязательно потребуется два физических сервера. Если вы планируете использовать следящий сервер, то нужно использовать три физических сервера, поскольку следящий сервер не может быть расположен на одном компьютере ни с сервером-принципалом, ни с зеркальным сервером.

В-третьих, возможность настройки зеркального отображения по умолчанию отключена. Чтобы включить ее, необходимо использовать флаг трассировки 1400. Чтобы этот флаг трассировки устанавливался всякий раз при запуске SQL Server, лучше всего настроить его включение при помощи параметра запуска. Нужный параметр будет выглядеть как `-t1400`. Проще всего настроить его запуск в свойствах службы SQL Server в Configuration Manager (вкладка **Advanced** (Дополнительно), поле редактирования **Startup Parameters** (Параметры запуска)). После настройки этого параметра вам нужно будет перезапустить сервер.

Последнее, в чем необходимо убедиться, — что база данных, для которой настраивается зеркальное отображение, работает в режиме восстановления **Full** (полного протоколирования). Если база данных работает в другом режиме, необходимо изменить режим перед началом настройки.

После этого можно приступать непосредственно к настройке зеркального отображения.

Для начала вам нужно будет произвести полное резервное копирование исходной базы данных (базы данных-принципала) и восстановить эту резервную копию на зеркальном сервере с параметром `NORECOVERY`. Если после создания полной резервной копии на сервере-принципале вы произвели там же резервное копирование журнала транзакций, то эту копию также нужно восстановить на зеркальном сервере с тем же параметром `NORECOVERY`.

Отметим еще два важных момента, которые связаны с созданием зеркальной базы данных. На зеркальном сервере имя базы данных должно быть точно таким же, как и на сервере-принципале (это условие является обязательным!). Кроме того, очень рекомендуется обеспечить на обоих серверах одинаковые пути и имена файлов базы данных и журналов транзакций. В противном случае при добавлении нового файла базы данных на сервере-принципале могут возникнуть проблемы.

После этого можно приступать к настройке параметров зеркального отображения. Это можно сделать двумя способами: при помощи графического интерфейса SQL Server Management Studio или посредством команд Transact-SQL. Первый способ проще, поэтому рассмотрим только его.

Для настройки параметров зеркального отображения в контекстном меню для объекта базы данных-принципала в **Object Explorer** выберите команду **Tasks | Mirror** (Задания | Зеркализовать). Откроется вкладка **Mirroring** (Зеркальное отображение) окна свойств базы данных.

На этой вкладке вначале нужно запустить мастер настройки безопасности зеркального отображения (Configure Database Mirroring Security Wizard). Он запускается нажатием кнопки **Configure Security** (Настроить безопасность). Этот мастер позволяет создать точки подключения по HTTP на всех серверах, участвующих в зеркалировании, а также выбрать учетные записи, которые будут использоваться для взаимодействия серверов. По умолчанию на всех серверах имена точек подключения будут одинаковыми — **Mirroring**. Для точки подключения на сервере-принципале по умолчанию используется порт 5022, а на зеркальном и следящем серверах — 5023.

После того как мастер выполнит настройку точек подключения и учетных записей, вашей следующей задачей будет выбор режима зеркального отображения на вкладке **Mirroring** свойств базы данных. В вашем распоряжении три варианта:

- Synchronous with automatic failover** (Синхронный с автоматическим переключением) — этот режим ориентирован на максимальную отказоустойчивость. Все транзакции применяются одновременно и на сервере-принципале, и на зеркальном сервере. Следящий сервер, который необходимо использовать в этом режиме, производит мониторинг состояния обоих серверов и в случае необходимости выполняет автоматическое изменение статуса зеркального сервера, открывая к нему доступ пользователей;
- Asynchronous (high performance)** (Асинхронный (высокая производительность)) — этот режим ориентирован на максимальную производительность работы. Транзакции изначально применяются только на сервере-принципале, а на зеркальный сервер они передаются в асинхронном режиме с небольшой задержкой;
- Synchronous (high protection)** (Синхронный (максимальная защита)) — режим ориентирован на обеспечение идентичности данных на обоих серверах. Транзакции обязательно должны быть успешно завершены и на сервере-принципале, и на зеркальном сервере. При этом автоматическое переключение зеркального сервера в рабочий режим не производится.

После того, как нужный режим выбран, вам осталось только воспользоваться кнопкой **Start Mirroring** (Начать зеркальное отображение) на вкладке **Mirroring** окна свойств базы данных, чтобы запустить базу данных в режиме зеркального отображения.

7.3.4. Мониторинг зеркального отображения

Мониторинг процессов зеркального отображения в SQL Server 2005 можно производить несколькими способами.

Первый способ — просто воспользоваться той же вкладкой **Mirroring** свойств базы данных. Общая информация о состоянии зеркального отображения будет показана в поле **Status** (Состояние) в нижней части этой вкладки.

Второй способ — воспользоваться счетчиками Системного монитора (про применение Системного монитора подробно будет рассказываться в разд. 11.4.5). Соответствующий объект Системного монитора с набором требуемых счетчиков называется **MSSQL\$имя_экземпляра: Database Mirroring**. К главным счетчикам для этого объекта можно отнести:

- **Transaction Delay** (Задержка транзакций) — на сколько в среднем применение транзакции на зеркальном сервере отстает от применения этой же транзакции на сервере-принципале, т. е. с какой задержкой происходит синхронизация серверов;
- **Redo Queue, KB** (Очередь повтора, Кбайт) — сколько килобайт накопилось в очереди для применения на зеркальном сервере. Этот показатель также показывает задержку синхронизации зеркального сервера по сравнению с сервером-принципалом, но не во времени, а в количестве данных;
- **Bytes Received/Sec** и **Bytes Sent/Sec** (Получено байт/сек и Отправлено байт/сек) — при помощи этих двух счетчиков можно определять, насколько активно происходит обмен информацией между сервером-принципалом и зеркальным сервером.

Третий способ мониторинга — использование специальных системных представлений, которые специально предназначены для этих целей. Все эти представления находятся в базе данных *master*:

- `sys.database_mirroring` — информация по всем базам данных на сервере, которые принимают участие в зеркальном отображении. Это представление имеется как на основном, так и на зеркальном сервере. Для каждой базы данных представлена информация о ее роли в зеркальном отображении, о ее текущем состоянии, о последнем номере последовательности журнала транзакций (*log sequence number, LSN*), которая применена к зеркальному серверу, и множество другой важной информации;
- `sys.database_mirroring_endpoints` — сводная информация по всем точкам подключения по HTTP, которые используются для зеркального отображения баз данных. Это представление также имеется и на основном, и на зеркальном сервере. Для каждой точки подключения выводится информация о роли, которую она играет, о том, включено ли шифрование, об используемом методе аутентификации, об алгоритмах шифрования;

- sys.database_mirroring_witnesses — это представление, предназначенное для получения информации на следующем сервере о сервере-принципале и зеркальном сервере, о режиме зеркального отображения, о текущем состоянии каждой базы данных, принимающей участие в зеркальном отображении, и т. п.;
- sys.dm_db_mirroring_connections — представление, выводящее информацию по сетевым соединениям, которые используются для зеркального отображения. Это представление имеется на всех серверах, принимающих участие в зеркальном отображении. Это представление очень важно для диагностики проблем зеркального отображения. В нем выводится информация об идентификаторах подключений, используемом транспорте, состоянии каждого из соединений, времени соединения, количестве переданных данных.

Четвертый способ мониторинга — использование событий зеркального отображения, информацию о которых можно получить при помощи профилировщика или уведомлений о событиях (*Event Notifications*). Про работу с профилировщиком подробно будет рассказываться в разд. 11.2.3, а про уведомления о событиях — в разд. 11.2.5. Единственное событие, относящееся к зеркальному отображению баз данных, называется **Database Mirroring State Change** (Изменение состояния зеркалирования баз данных). Оно находится в группе **Database** (База данных) на вкладке **Event Selection** (Выбор событий) свойств сеанса трассировки. Главные параметры этого события — **State** (Состояние) и **Text** (Текст). Столбец **State** позволяет узнать, как именно изменилось состояние базы данных, принимающей участие в зеркальном отображении, а столбец **Text** позволяет получить текстовое описание для этого события с дополнительной информацией.

7.3.5. Смена ролей серверов

Главное назначение системы зеркального отображения баз данных — обеспечение отказоустойчивости. Поэтому нужно быть готовым к тому, что однажды вам придется воспользоваться процедурой смены ролей серверов и сделать зеркальный сервер основным.

Отметим сразу такой момент. При переключении ролей серверов необходимо перенаправить клиентов, которые обращались к серверу-принципалу, на сервер, который раньше выполнял роль зеркального сервера, а теперь стал самостоятельным. При использовании обычных подключений по ODBC или OLE DB вам придется выполнить такое переключение вручную. Это можно сделать, например, изменив настройки на всех клиентских компьютерах или поменяв имя бывшего зеркального сервера, изменив его IP-адрес и т. п. Однако если клиентское приложение использовало для подключения к серверу сред-

ства SQL Native Client или ADO.NET Data Provider, клиенты смогут в автоматическом режиме обращаться к новому серверу. Для этого достаточно с самого начала в строке подключения клиентского приложения указать, кроме основного сервера, также и запасной сервер, в роли которого будет выступать сервер с зеркальной копией базы данных. Выглядеть такая строка подключения может так:

```
"server=имя_сервера_принципала; failover partner=имя_зеркального_сервера;  
database=AdventureWorks"
```

Про следящий сервер клиентские компьютеры ничего не знают, и поэтому указывать его в строке подключения не надо.

В SQL Server 2005 предусмотрено три варианта смены ролей серверов при зеркальном отображении баз данных:

- **автоматическое переключение ролей серверов** при выходе из строя сервера-принципала (при потере с ним связи). Для такого автоматического переключения обязательно необходимо, чтобы в системе зеркального отображения использовался следящий сервер;
- **переключение ролей вручную**, когда оба сервера — и сервер-принципал, и зеркальный сервер — остаются доступными. В этом случае серверы просто меняются ролями;
- **переключение ролей вручную в аварийном режиме**, когда сервер-принципал недоступен.

Первый вариант является самым простым. Он требует, чтобы обязательно был настроен следящий сервер, а само зеркальное отображение работало в режиме **Synchronous with automatic failover**. Как только следящий сервер определяет, что сервер-принципал вышел из строя или связь с ним потеряна, он меняет в своих параметрах статус этого сервера на **DISCONNECTED** (отсоединенний), и после завершения применения к зеркальному серверу всех транзакций, которые были отправлены к нему с сервера-принципала, открывает доступ к зеркальной базе данных в обычном режиме. Если связь с бывшим сервером-принципалом будет восстановлена, он автоматически меняет свою роль и начинает выполнять роль зеркального сервера.

Второй вариант — ручное переключение ролей серверов, когда оба сервера доступны, — обычно используется администратором для проверки возможности переключения или для того, чтобы на время отключить сервер-принципал (например, для выполнения каких-то регламентных операций по замене оборудования или установке пакета обновлений). Эту операцию можно произвести двумя способами:

- на вкладке **Mirroring** свойств зеркалированной базы данных нажать кнопку **Failover** (Переключение);

- выполнить команду Transact-SQL:

```
ALTER DATABASE имя_зеркализированной_базы_данных SET PARTNER FAILOVER;
```

Оба сервера просто поменяются ролями.

Третий вариант — ручное переключение серверов, когда сервер-принципал недоступен. Обычно этот вариант используется в аварийной ситуации, когда сервер-принципал внезапно вышел из строя, а администратору нужно как можно быстрее обеспечить восстановление работоспособности сервера и нормальную работу клиентов. В других ситуациях использование этого варианта не рекомендуется, поскольку существует риск потерять данные, которые были переданы с сервера-принципала на зеркальный сервер. Для выполнения этой операции графического интерфейса не предусмотрено. На сервере, который выполняет роль зеркального, нужно выполнить команду Transact-SQL:

```
ALTER DATABASE имя_зеркализированной_базы_данных SET PARTNER  
FORCE_SERVICE_ALLOW_DATA_LOSS;
```

База данных, которая выполняла роль зеркальной копии, будет открыта для пользователей в нормальном режиме.

7.3.6. Приостановка и отмена зеркального отображения

В некоторых ситуациях администратору может потребоваться временно приостановить работу системы зеркального отображения. Например, когда сервер-принципал работает с максимальным количеством пользователей, и администратор хочет на время снять с него дополнительную нагрузку по зеркальному отображению баз данных. Другой случай — когда на некоторое время сетевые соединения между сервером-принципалом и зеркальным сервером станут недоступны.

При приостановке зеркального отображения состояние сеанса изменится на SUSPENDED (отложенное). Изменения будут копиться в журнале транзакций сервера-принципала до тех пор, пока администратор не продолжит работу сеанса зеркального отображения. После этого все накопившиеся изменения будут в обычном порядке переданы на зеркальный сервер.

Приостановить и продолжить работу сеанса зеркального отображения можно при помощи графического интерфейса SQL Server Management Studio или командой Transact-SQL. Для приостановки работы достаточно просто нажать кнопку **Pause** (Пауза) на вкладке **Mirroring** свойств базы данных. Для продолжения работы нужно нажать на ту же кнопку (но она уже будет называться **Resume** (Продолжить)). Для приостановки зеркального отображения можно также воспользоваться командой Transact-SQL:

```
ALTER DATABASE имя_зеркализированной_базы_данных SET PARTNER SUSPEND;
```

а для возобновления — командой:

```
ALTER DATABASE имя_зеркальированной_базы_данных SET PARTNER RESUME;
```

Если вы хотите полностью удалить настроенную систему зеркального отображения, то можно нажать кнопку **Stop Mirroring** (Остановить зеркальное отображение) на той же вкладке или выполнить команду:

```
ALTER DATABASE имя_зеркальированной_базы_данных SET PARTNER OFF;
```

Вся информация о настройках зеркального отображения будет удалена. При этом зеркальная база данных останется в положении NORECOVERY. Ее можно удалить или привести в рабочий режим, вручную восстановив последнюю копию журнала транзакций с базы данных сервера-принципала в режиме RECOVERY. Для того чтобы опять настроить зеркальное отображение базы данных, все процедуры придется повторить заново.

Задание для самостоятельной работы 7.1. Настройка доставки журналов

Задание:

1. Установите на вашем компьютере второй экземпляр сервера SQL Server 2005 со следующими параметрами:

- должен быть установлен полный набор компонентов;
- именованный экземпляр должен называться Server2;
- все службы SQL Server 2005 должны работать от имени локальной системной учетной записи;
- режим аутентификации — **Mixed**;
- пароль для учетной записи sa — P@ssw0rd;
- кодировка по умолчанию — **Cyrillic_General**;
- не посыпать сообщения об ошибках в Microsoft.

Для остальных параметров оставьте значения по умолчанию.

2. Создайте на первом экземпляре сервера SQL Server 2005 (с именем SQL2005) на вашем компьютере базу данных DB1 с параметрами по умолчанию.

3. Создайте в ней таблицу Table1 со следующими параметрами:

```
USE DB1;
GO
CREATE TABLE dbo.Table1(
    Col1 int IDENTITY(1,1) NOT NULL,
    Col2 nvarchar(50));
```

4. Настройте автоматическую доставку журналов для созданной вами базы данных DB1 к базе данных с таким же названием на втором экземпляре SQL Server 2005 (Server2) с интервалом доставки 2 минуты. При этом для доставки журналов должен использоваться каталог C:\LogShippingBackup1, которому нужно назначить сетевой путь *имя_вашего_компьютера*\LogShippingBackup1. Для получения журналов должен использоваться каталог C:\LogShippingBackup2. Сервером мониторинга для поставки журналов должен выступать первый экземпляр SQL Server 2005.
5. Проверьте работоспособность системы доставки журналов при помощи отчетов сервера и истории заданий. Заполните любыми значениями несколько строк в таблице Table1 исходной базы данных и убедитесь, что эти изменения отобразились на второй базе данных.
6. Отмените поставку журналов и убедитесь, что вы можете обращаться с запросами к таблице Table1 на сервере Server2.

Решение:

К пункту 1 задания — установка именованного экземпляра SQL Server 2005:

1. Запустите программу установки SQL Server 2005, примите лицензионное соглашение и на экране **Installing Prerequisites** (Пререквизиты для установки) нажмите кнопку **Next** (Дальше).
2. Убедитесь, что на экране **System Configuration Check** (Проверка конфигурации системы) нет ни ошибок, ни предупреждений, и еще раз нажмите кнопку **Next**.
3. На экране **Registration Information** (Информация о регистрации) введите ваше имя и имя вашей организации.
4. На экране **Components to install** (Компоненты для установки) установите только флажок **SQL Server Database Services** (Службы баз данных SQL Server).
5. На экране **Instance Name** (Имя экземпляра) переставьте переключатель в положение **Named Instance** (Именованный экземпляр) и введите имя создаваемого именованного экземпляра (*Server2*).
6. На экране **Service Account** (Учетная запись служб) переставьте переключатель в положение **Use the built-in System Account** (Использовать встроенную системную учетную запись) и убедитесь, что в правом списке выбрано "Local system". На этом же экране в группе **Start services at the end of setup** (Запускать службы после окончания установки) установите флажок **SQL Server Agent**.
7. На экране **Authentication Mode** (Режим аутентификации) установите переключатель в положение **Mixed Mode** (Смешанный режим) и укажите для учетной записи sa пароль P@ssw0rd.

8. На экране **Collation Settings** (Настройки сопоставления) выберите кодировку **Cyrillic_General**.
9. На экране **Error and Usage Report Settings** (Настройки отчетов об ошибках и использовании) убедитесь, что все флажки сняты и нажмите кнопку **Next**, а затем на экране **Ready To Install** (Готовность к установке) нажмите кнопку **Install** (Установить).

К пункту 2 задания — создание базы данных DB1 на первом экземпляре сервера SQL Server 2005:

1. Для создания базы данных DB1 с параметрами по умолчанию подключитесь к первому экземпляру SQL Server 2005 (который называется SQL2005) и выполните на нем следующую команду:

```
CREATE DATABASE DB1;
```

Пункт 3 задания — создание таблицы — выполняется с помощью кода, приведенного в задание.

К пункту 4 задания — настройка автоматической доставки журналов:

1. Создайте на диске C: вашего компьютера каталог C:\LogShippingBackup1. Откройте свойства этой папки, перейдите на вкладку **Доступ** и сделайте эту папку общей с сетевым именем **LogShippingBackup1**. Затем нажмите кнопку **Разрешения** и предоставьте группе **Все** полный доступ к этому каталогу. Создайте также на диске C: каталог LogShippingBackup2.
2. В SQL Server Management Studio раскройте узел **имя_вашего_сервера\SQL2005\ Databases** и в контекстном меню созданной базы данных DB1 выберите **Tasks | Ship Transaction Logs**. Откроется вкладка **Transaction Log Shipping** свойств этой базы данных.
3. На этой вкладке установите флажок **Enable this as a primary database in a log shipping configuration** и нажмите кнопку **Backup Settings**.
4. В окне **Transaction Log Backup Settings** в поле **Network path to backup folder** введите путь \\имя_вашего_сервера\LogShippingBackup1 (например, \\LONDON2\LogShippingBackup1).
5. Нажмите кнопку **Schedule** и измените расписание резервного копирования таким образом, чтобы оно производилось каждые 2 минуты. Затем в окне **Transaction Log Backup Settings** нажмите кнопку **OK**, чтобы вернуться в окно свойств базы данных.

Примечание

Таким образом, вы настроили интервал в 2 минуты только для резервного копирования исходной базы данных. Для того чтобы изменить интервал копиро-

вания и восстановления (по умолчанию один раз в 15 минут), необходимо изменить свойства заданий на втором сервере. Это можно сделать как при настройке доставки журналов, так и потом.

6. В списке **Secondary server instances and databases** нажмите кнопку **Add**, а затем в открывшемся окне **Secondary Database Settings** нажмите кнопку **Connect**. В окне **Connect to server** (Подключиться к серверу) введите имя второго экземпляра SQL Server 2005 (например, `LONDON2\SERVER2`) и нажмите кнопку **Connect**, чтобы вернуться в окно **Secondary Database Settings**.
7. В окне **Secondary Database Settings** оставьте предлагаемое по умолчанию значение `DB1` в поле **Secondary Database** и на вкладке **Initialize Secondary Database** оставьте для переключателя значение по умолчанию `Yes, generate a full backup of the primary database`.
8. Перейдите на вкладку **Copy Files** и в поле **Destination Folder for copied files** (Каталог назначения для скопированных файлов) введите значение `C:\LogShippingBackup2`. Затем на этой вкладке и на вкладке **Restore Transaction Log** нажмите кнопку **Edit Job**, чтобы открыть свойства создаваемых заданий, перейдите в открывшемся окне на вкладку **Schedules** и измените для них расписание таким образом, чтобы копирование и восстановление также производились один раз в 2 минуты. Оставьте для остальных параметров значения по умолчанию и нажмите кнопку **OK**, чтобы вернуться в окно свойств базы данных.
9. На вкладке **Transaction Log Shipping** свойств базы данных установите флажок **Use a monitor server instance** и нажмите кнопку **Settings**. В открывшемся окне **Monitor Server Instance** нажмите кнопку **Connect** и подключитесь к серверу `имя_вашего_сервера\SQL2005`. Закройте окно базы данных с сохранением внесенных изменений. Убедитесь, что в окне **Save Log Shipping Configuration** (Сохранить конфигурацию поставки журналов) все этапы выполнены успешно.

К пункту 5 задания — просмотр информации о поставке журналов:

1. В окне **Object Explorer** в SQL Server Management Studio выделите строку для сервера, который был назначен сервером мониторинга доставки журналов (`имя_вашего_сервера\SQL2005`), и в меню **View** (Просмотр) выберите пункт **Summary** (Сводное).
2. В открывшемся окне **Summary** нажмите на стрелку рядом со списком **Report**, чтобы открыть список отчетов. Затем в этом списке отчетов выберите отчет **Transaction Log Shipping Status**.
3. В окне **Object Explorer** раскройте узел `имя_вашего_сервера\SQL2005 | SQL Server Agent | Jobs` и просмотрите историю выполнения задания

LSBackup_DB1 (при помощи команды **View history** (Просмотреть историю) в контекстном меню). Подключитесь ко второму серверу (*имя_вашего_сервера\Server2*) и просмотрите историю выполнения заданий **LSCopy** и **LSRestore**. Все эти задания должны выполняться без ошибок.

К пункту 6 задания — отмена доставки журналов:

1. В окне **Object Explorer** раскройте узел *имя_вашего_сервера\SQL2005 | SQL Server Agent | Jobs* и откройте свойства задания **LSBackup_DB1**, а затем на вкладке **General** снимите флажок **Enabled**. Подождите 2 минуты (это время, которое потребуется, чтобы скопировать и восстановить уже созданные резервные копии журнала транзакций), а затем точно так же отключите задания **LSCopy** и **LSRestore** на втором сервере.
2. Откройте историю выполнения задания **LSRestore** (при помощи команды **View History** контекстного меню) на втором сервере и найдите информацию о последнем восстановленном журнале событий.

3. Подключитесь из окна редактора кода SQL Server Management Studio ко второму серверу (*имя_вашего_сервера\Server2*) и выполните команду на повторное восстановление последнего журнала транзакций (который вы определили согласно решению предыдущего пункта задания). Соответствующая команда может выглядеть, например, так:

```
USE master;
RESTORE LOG DB1 FROM DISK =
    C:\LogShippingBackup2\DB1_20060407120603.trn' WITH RECOVERY;
```

4. Откройте свойства базы данных **DB1** на первом сервере (*имя_вашего_сервера\SQL2005*) и перейдите на вкладку **Transaction Log Shipping**.
5. Снимите флажок **Enable this as a primary database in a log shipping configuration** и нажмите кнопку **Yes** в окне подтверждения, а затем — кнопку **OK**. После удаления конфигурации доставки журнала убедитесь, что задания, историю выполнения которых вы просматривали согласно решению предыдущего пункта задания, удалены.
6. Раскройте контейнер **Databases** на втором сервере (*имя_вашего_сервера\Server2*) и убедитесь, что база данных **DB1** находится в обычном состоянии, а в таблице **dbo.Table1** отображаются все изменения, которые вы внесли в исходную таблицу.



ГЛАВА 8

Автоматизация администрирования SQL Server 2005

В предыдущих главах было рассмотрено выполнение основных административных операций на SQL Server 2005. Однако, как знают опытные администраторы, многие административные операции на сервере являются повторяющимися. Например, очень часто изо дня в день приходится производить:

- резервное копирование;
- проверку целостности баз данных;
- загрузку и выгрузку данных;
- перестроение индексов и дефрагментацию,

а также многие другие действия, набор которых зависит от конкретной задачи.

Кроме того, в некоторых ситуациях необходимо сделать так, чтобы администратор был немедленно извещен о каких-то важных событиях на сервере (например, внесены изменения в важные таблицы, выявлена ошибка при проверке целостности данных, возникли проблемы при массовой загрузке и т. п.).

И выполнение определенных действий по расписанию, и отслеживание событий рекомендуется автоматизировать. Чем более опытен и квалифицирован администратор, тем большее количество повседневных операций он старается автоматизировать, чтобы освободить свое время для других дел.

В этой главе речь и пойдет про автоматизацию административных операций. Также будут рассмотрены средства, которые обеспечивают дополнительные функциональные возможности для автоматизации.

Отметим также, что часто для автоматизации административных операций удобно использовать скрипты VBScript или JavaScript (или программы на

любом COM-совместимом языке, например Visual Basic, VBA, C++, Java, Delphi и др., или .NET-совместимые программы), которые работают с объектными моделями SMO, SQL-DMO и WMI. Речь о них пойдет в гл. 9.

8.1. Автоматизация административных операций средствами SQL Server Agent

8.1.1. Что такое SQL Server Agent

SQL Server Agent — это служба SQL Server, основное назначение которой — автоматизация выполнения административных операций. Сама автоматизация осуществляется при помощи:

- **заданий (jobs)** — именованных наборов действий, которые можно выполнять по расписанию;
- **предупреждений (alerts)** — действий, которые выполняются в ответ на событие, произошедшее на SQL Server. Таким действием может быть, например, выполнение задания или отправка сообщения оператору. События — это либо ошибки с определенным номером на SQL Server (их можно определять и генерировать самостоятельно), либо выход счетчика производительности за какие-то границы, либо специальные события WMI (т. е. ответ, пришедший на специальный событийный запрос на языке WQL);
- **операторов (operators)** — записей в адресной книге, на которые будут отправляться сообщения.

Задания, предупреждения и операторы будут подробно рассмотрены в следующих разделах. Пока же отметим только общие моменты, которые связаны с SQL Server Agent.

Первое, что необходимо отметить, — для использования автоматизации административных операций необходимо, чтобы служба SQL Server Agent работала. Будет ли она запускаться автоматически при запуске сервера или ее нужно будет запускать вручную, зависит от параметров, которые вы выбрали при установке сервера. Проверить, работает ли эта служба (и при необходимости запустить или изменить режим запуска), можно при помощи SQL Server Configuration Manager (см. разд. 3.3).

Второй момент — служебная информация SQL Server Agent (в том числе информация о заданиях, предупреждениях и операторах) хранится в специальной служебной базе данных msdb. Если вы активно работаете с заданиями и предупреждениями, не забывайте регулярно производить резервное копирование этой базы данных.

Третье, что нужно отменить, — возможности SQL Server Agent (и его работоспособность) зависят от того, от имени какой учетной записи работает эта служба. Рекомендуется, чтобы:

- SQL Server Agent работал от имени той же доменной учетной записи, от имени которой работает сам SQL Server;
- эта доменная учетная запись обладала бы на компьютере правами локального администратора.

Учетная запись, от имени которой работает SQL Server Agent, определяется при установке SQL Server 2005. Затем ее можно изменить, например, при помощи SQL Server Configuration Manager.

Возможностей настройки безопасности при работе SQL Server Agent в SQL Server 2005 очень много. Подробнее про них будет рассказано в разд. 8.1.4.

И, в-четвертых, для SQL Server Agent предусмотрена своя система журналов, при помощи которой можно получить информацию о всех происходящих с этой службой событиях. Просмотреть журналы можно из контейнера **SQL Server Agent | Error Logs** (Журналы ошибок) в SQL Server Management Studio.

Перед созданием заданий, оповещений и операторов рекомендуется проверить параметры SQL Server Agent: соответствуют ли они вашим потребностям.

8.1.2. Параметры настройки SQL Server Agent

Параметры SQL Server Agent, как и большинства других объектов SQL Server 2005, можно изменить двумя способами:

- при помощи графического интерфейса SQL Server Management Studio. Для этого достаточно открыть свойства контейнера **SQL Server Agent** в **Object Explorer**;
- при помощи специальных хранимых процедур (например, `sp_set_sqlagent_properties`). Рассматривать эти хранимые процедуры мы не будем, поскольку команды на их применение можно очень просто сгенерировать при помощи кнопки **Script** (Скрипт) на экране свойств SQL Server Agent в Management Studio.

Далее приведен перечень параметров, которые можно настроить для SQL Server Agent с краткими комментариями. Вначале рассмотрим параметры на вкладке **General** (Общие) окна свойств SQL Server Agent.

- Auto restart SQL Server if it stops unexpectedly** (Автоматически перезапускать SQL Server при неожиданной остановке) — SQL Server Agent бу-

дет контролировать работу службы SQL Server и при необходимости запускать сервер заново. Конечно, SQL Server без причины обычно не останавливается, поэтому значение этого параметра, в принципе, не очень важно. Тем не менее такой контроль по умолчанию включен;

- **Auto restart SQL Server Agent if it stops unexpectedly** (Автоматически перезапускать SQL Server Agent при неожиданной остановке) — это обратный контроль. Данный параметр определяет, будет ли SQL Server контролировать работу службы SQL Server Agent и при необходимости производить перезапуск этой службы. По умолчанию параметр также включен;
- **File name** (Имя файла) в разделе **Error Log** (Журнал ошибок) — это путь к текстовому файлу протокола работы службы SQL Server Agent. Просмотреть этот протокол можно при помощи контейнера **SQL Server Agent | Error Logs** (Журнал ошибок) (или просто открыть файл в Блокноте);
- **Include execution trace messages** (Включить сообщения трассировки) — если этот флагок установлен, то в файл протокола будет также записываться трассировочная информация для процесса SQL Server Agent. Обычно эта возможность нужна только для отладки (при этом не выполнения заданий, а только самой службы SQL Server Agent). В журнал будет записываться большое количество дополнительной информации, которая администраторам обычно не нужна;
- **Write OEM file** (Записывать файл в формате OEM) — по умолчанию текстовый файл протокола создается в формате UNICODE. Если вам нужен файл в обычном текстовом формате (не UNICODE), можно установить этот флагок. Обычно он нужен только в одной ситуации: когда вы обрабатываете файл журнала какой-то специализированной программой, не понимающей кодировку UNICODE;
- **Net send recipient** (Получатель по net send) — имя компьютера (IP-адрес) или имя пользователя, которому будут автоматически передаваться по команде `net send` (при помощи окон сообщений) записи, регистрируемые в протоколе SQL Server Agent. Здесь нужно отметить два момента:
 - служба Messenger (Служба сообщений), которая ответственна за работу с сообщениями, передаваемыми по `net send`, по умолчанию в Windows Server 2003 отключена. Чтобы получать сетевые сообщения, вам потребуется сначала включить эту службу;
 - в журнал событий SQL Server Agent записывается большое количество сообщений, и всплывающие окна будут сильно мешать работе пользователя, на компьютер которого они отправляются. Поэтому имеет смысл определять получателя только в каких-то специальных ситуациях.

На вкладке **Advanced** (Дополнительно) вы можете настроить параметры перенаправления сообщений и условий, при которых центральный процессор будет считаться бездействующим:

- **Forward events to a different server** (Перенаправлять события на другой сервер) — этот параметр имеет смысл включать тогда, когда у вас на предприятии используются несколько серверов, и вы хотите упростить просмотр журналов, сконцентрировав все сообщения на одном сервере. По умолчанию, если на вашем компьютере установлено несколько экземпляров SQL Server, все сообщения будут передаваться на тот экземпляр, который был установлен первым;
- **Events** (События) — при помощи этой группы параметров вы можете определить, какие именно события будут перенаправляться для записи на другой сервер. В вашем распоряжении следующие параметры:
 - **Unhandled events** (Неперехваченные события) — на другой SQL Server будут передаваться записи только о тех событиях, для которых не настроены предупреждения (про предупреждения будет рассказываться в *разд. 8.1.7*);
 - **All events** (Все события) — будет передаваться информация о всех событиях;
 - **If event has severity at or above** (Если у события важность находится на уровне или выше) — этот параметр позволяет настроить фильтр для передаваемых сообщений по их важности. По умолчанию используется самый низкий уровень 001, поэтому передаваться будут все сообщения;
- **Define idle CPU condition** (Определить условия простоя центрального процессора) — единственное назначение этого набора параметров — возможность определить для задания, что оно должно запуститься, когда процессоры компьютера, на котором установлен SQL Server, бездействуют (т. е. выполнены определенные при помощи этих параметров условия). На практике такие задания используются очень редко. Сами условия, при выполнении которых центральный процессор будет считаться бездействующим, можно определить при помощи следующих параметров:
 - **Average CPU usage falls below** (Средняя загрузка центрального процессора падает ниже) — по умолчанию установлено 10%;
 - **And remains below this level for** (И остается ниже этого уровня в течение) — по умолчанию задано 600 секунд, т. е. 10 минут.

На вкладке **Alert system** (Система предупреждений) вы можете определить общие параметры работы SQL Server Agent с предупреждениями. Обратите внимание, что вы не можете определить адреса тех, кому будут отправляться

предупреждения, из свойств SQL Server Agent — для этого потребуется создать объекты операторов. На этой вкладке вы можете определить лишь общие настройки, которые будут применяться для всех предупреждений:

□ **Enable mail profile** (Включить почтовый профиль) — если этот флагок установлен, то SQL Server Agent получает возможность взаимодействовать с электронной почтой (например, отправлять по электронной почте предупреждения для операторов или результаты выполнения заданий). Если этот флагок установлен, в вашем распоряжении появляются еще две возможности:

- **Mail system** (Почтовая система) — возможность выбрать одну из двух систем для взаимодействия с электронной почтой: SQLMail (унаследованная система, ориентированная на MAPI) или Database Mail (другое название — SQLMail, более современная система, ориентированная на SMTP). Подробнее про работу SQL Server и SQL Server Agent с электронной почтой будет рассказано в разд. 8.2.

При помощи кнопки **Test** (Проверить) можно проверить работоспособность настроенных параметров для работы с электронной почтой, отправив пробное сообщение;

- **Save copies of the sent messages in the Sent Items folder** (Сохранять копии отправленных сообщений в папке Sent Items) — эта возможность доступна только тогда, когда для отправки сообщений используется протокол MAPI (а значит, используется система SQLMail);

Чтобы параметры, связанные с настройками электронной почты, вступили в силу, необходимо перезапустить службу SQL Server Agent.

□ в разделе **Pager e-mails** (Сообщения на пейджер) определяются параметры тех сообщений, которые будут отправляться на пейджер (или на сотовый телефон как сообщения SMS — в зависимости от настроек соответствующего шлюза):

- **To line, CC line, Subject** — возможность определить префиксы и суффиксы для строк **Кому**, **Копия** и **Тема сообщения** соответственно;
- **Include body of e-mail in notification message** (Включить тело письма в уведомляющее сообщение) — если этот флагок установлен, то на пейджер или сотовый телефон будут отправляться уведомления (о перехваченной ошибке или результатах выполнения задания) целиком. Эти сообщения могут быть достаточно большими, поэтому с этим параметром нужно быть осторожнее. Если этот флагок снят, то на пейджер или телефон придет краткое уведомление без подробной информации;
- **Fail-safe operator** — в соответствии с учебными курсами Microsoft, это должно переводиться как "резервный оператор" или "оператор послед-

ней надежды". Однако слушатели на курсах придумали более меткое название — "Кто будет крайним". Смысл этого параметра очень прост: для каждого оператора можно указать рабочие часы. Если получилось так, что событие произошло в тот момент, когда все операторы согласно расписанию отдыхают, сообщение будет отправлено "оператору последней надежды" (вне зависимости от расписания его работы). Вы можете указать соответствующий объект оператора и выбрать способ его уведомления: при помощи электронной почты, пейджера или сетевого сообщения;

- **Replace tokens for all job responses to alerts** (Заменять маркеры для всех заданий, которые запускаются в ответ на оповещения) — этот новый параметр определяет, можно ли будет использовать специальные подстановочные символы (например, для имени базы данных) в параметрах заданий, которые автоматически запускаются при срабатывании предупреждений. Подробнее об этом будет рассказано в *разд. 8.1.7.*

На вкладке **Job System** (Система заданий) вы можете определить общие параметры для выполнения заданий SQL Server Agent:

- Shutdown time-out interval (in seconds)** (Время ожидания при отключении (в секундах)) — если вы дали команду на остановку службы SQL Server Agent, а в это время выполняется задание, то SQL Server Agent даст на его завершение столько секунд, сколько указано в этом параметре (по умолчанию задано 15 секунд). По истечении этого времени работа задания будет прекращена принудительно;
- Job step proxy account** (Учетная запись прокси для этапов заданий) — этот параметр позволяет определить учетную запись Windows, от имени которой в ходе выполнения заданий будут выполняться различные действия в операционной системе. По умолчанию SQL Server Agent выполняет эти действия от имени учетной записи, под которой он работает. Обратите внимание, что этот параметр будет доступен только для служб SQL Server Agent, которые входят в состав SQL Server 7.0 и 2000 (если вы решили администрировать их из Management Studio). Для SQL Server 2005 учетные записи прокси настраиваются из контейнера **SQL Server Agent | Proxies**.

На вкладке **Connection** (Подключение) настраиваются параметры подключения службы SQL Server Agent к SQL Server:

- Alias local host server** (Псевдоним для локального сервера) — псевдоним для экземпляра SQL Server (он обязательно должен быть расположен на том же компьютере), к которому будет подключаться SQL Server Agent. Подробно про псевдонимы рассказывалось в *разд. 3.3.4.* Этот параметр нужно заполнять только в том случае, если для "нормального имени" SQL Server на вашем компьютере уже существует какой-то другой псевдоним, перенаправляющий запросы к нему на другой сервер;

□ **SQL Server connection** (Подключение к SQL Server) — параметр определяет учетную запись, которая будет использоваться службой SQL Server Agent для подключения к SQL Server. Как уже говорилось, очень рекомендуется использовать для работы службы SQL Server и SQL Server Agent одну и ту же учетную запись. Тогда здесь проще всего оставить аутентификацию Windows (которая выбирается по умолчанию). Если SQL Server и SQL Server Agent работают под разными учетными записями или вам нужно подключаться (для целей обеспечения обратной совместимости) от имени логина SQL Server, то в этом параметре вы можете определить соответствующую учетную запись Windows или логин SQL Server Agent.

На вкладке **History** (История) определяются параметры хранения истории выполнения заданий в журналах SQL Server Agent:

□ **Limit size of job history log** (Ограничить размер журнала истории выполнения заданий) — если этот флагок снять, то старая информация о выполнении заданий не будет автоматически удаляться из журналов событий SQL Server Agent. Вам потребуется удалять ее вручную. Если же этот флагок установлен (по умолчанию), то можно настроить два дополнительных параметра:

- **Maximum job history log size (in rows)** (Максимальное количество записей для истории выполнения заданий (в строках));
- **Maximum job history rows per job** (Максимальное количество записей для одного задания);

□ **Automatically remove agent history** (Автоматически удалять историю агента) — этот параметр позволяет определить, через какое время записи о истории выполнения заданий будут удаляться автоматически.

8.1.3. Работа с заданиями SQL Server Agent

Задания (jobs) — это наиболее часто используемое средство автоматизации административных операций. Задания можно определить как именованные наборы действий, которые можно запланировать для выполнения по расписанию (а можно выполнять и вручную). На многих рабочих серверах на предприятиях существует сложная система заданий, при помощи которых выполняется множество операций: резервное копирование, проверка целостности, дефрагментация и перестроение индексов, загрузка и выгрузка данных, генерация страниц HTML и т. п. Задания могут создаваться и в автоматическом режиме, например, при настройке доставки журналов или репликации.

Работа с заданиями обычно производится из контейнера **Jobs** в SQL Server Agent. Создать новое задание можно при помощи команды **New Job** (Новое задание) из контекстного меню для контейнера **Jobs**. Откроется окно, анало-

личное представленному на рис. 8.1, в котором вам потребуется настроить свойства задания. Подробно про работу со свойствами задания будет рассказано ниже в этом разделе.

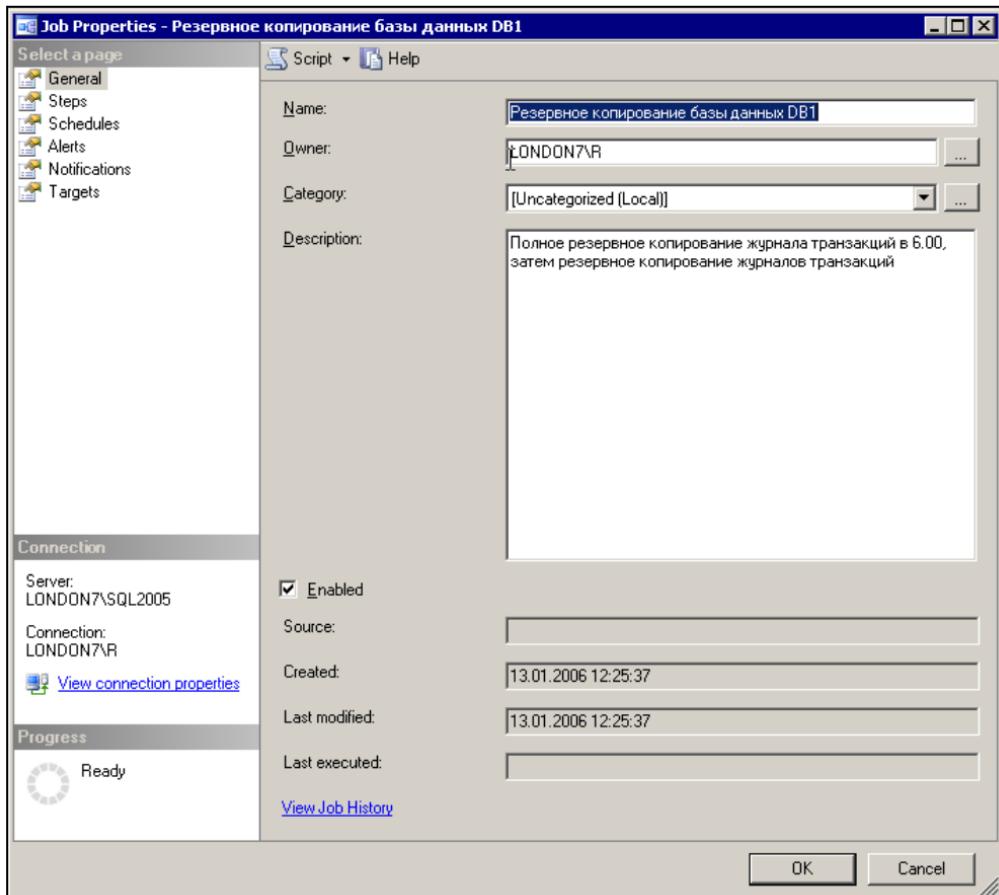


Рис. 8.1. Окно свойств задания SQL Server Agent

Очень часто возникает необходимость скопировать задания с одного сервера на другой, чтобы упростить автоматизацию выполнения схожих операций. Проще всего сделать это так: в контекстном меню для созданного задания нужно выбрать команду **Script Job As | CREATE TO | New Query Editor Window** (Отскриптовать задание как | Создать | Новое окно редактора запросов). В результате в окно редактора кода будет загружен скрипт с командами на создание задания с аналогичными параметрами. Вам останется только исправить некоторые параметры в этом скрипте, сохранить его и запустить на выполнение на другом сервере.

Этот же способ можно использовать и для получения информации о хранимых процедурах, которые можно использовать для создания заданий из кода Transact-SQL. Их синтаксис достаточно сложен и рассматриваться здесь не будет.

Теперь рассмотрим параметры, которые можно настроить для создаваемого задания. На вкладке **General** вы можете настроить или просмотреть общие параметры для задания:

- **Name** (Имя) — это, конечно, имя для создаваемого задания. Ему совершенно не обязательно соответствовать правилам именования объектов SQL Server. Используйте любое удобное для вас словосочетание;
- **Owner** (Владелец) — владелец данного задания. По умолчанию владельцем становится тот пользователь, который это задание создал. Информация о владельце обычно используется для определения прав, с которыми задание может выполнять различные действия на SQL Server, а также по владельцу можно производить фильтрацию при создании отчетов о выполнении заданий;
- **Category** (Категория) — этот параметр ни на что не влияет. Используется для группировки заданий и для их сортировки при отображении в Management Studio;
- **Description** (Описание) — простое описание задания, например заметки, которые администратор делает для самого себя, чтобы не забыть, для чего предназначено это задание;
- **Enabled** (Включено) — если задание в настоящее время вам не нужно, но может потребоваться потом, вы можете просто снять этот флажок. Отключенное задание выполняться не будет. Можно также отключить расписание для этого задания;
- **Source** (Источник) — в этом параметре можно просмотреть сервер, который запускает данное задание на выполнение (*master server*). Параметр используется только для мультисерверных заданий (которые могут выполняться на разных серверах). Подробнее про мультисерверные задания будет рассказываться в разд. 8.1.6;
- **Created** (Создано), **Last modified** (Изменено в последний раз), **Last Executed** (Запускалось в последний раз) — это, соответственно, время создания, время последнего изменения и время последнего запуска на выполнение для задания.

На вкладке **Steps** (Этапы) производится самая важная часть настройки заданий. Здесь нужно будет определить *этапы* (*steps*), т. е. действия, из которых состоит задание. Создание этапа производится при помощи кнопки **New** (Но-

вый). Вам потребуется указать имя создаваемого этапа и выбрать его тип. Если отбросить типы этапов, которые начинаются с префикса **Replication** (задания с такими этапами практически всегда создаются автоматически при настройке репликации), то в вашем распоряжении следующие варианты:

- **ActiveX Script** (Скрипт ActiveX) — это самый функциональный тип этапа. Фактически при помощи этого типа этапа вы можете сделать на SQL Server и в операционной системе (не только на локальных, но и на удаленных компьютерах) абсолютно все (включая возможности любых других типов этапов). Для работы с SQL Server в вашем распоряжении имеются объектные модели SQL-DMO, SMO и поставщика WMI для SQL Server (речь о них пойдет в гл. 9), для работы с операционной системой — объектные модели Windows Script Host, Scripting Runtime, WMI и т. п. Исходной точкой для их освоения может стать сайт www.microsoft.com/scripting.

Отметим некоторые моменты, связанные с этим типом этапов:

- скрипты можно создавать на любом COM-совместимом скриптовом языке. По умолчанию в Windows 2000 и Windows 2003 Server встроены интерпретаторы только для VBScript и JavaScript, но если ваш любимый язык — Perl или TCL, то вам ничего не мешает установить на сервер интерпретатор для этого языка и использовать именно его;
- если вам нужны программные конструкции — циклы, проверки значений и т. п., то нужно использовать этот тип этапа;
- лучше всего, конечно, вначале написать и отладить скрипт при помощи специализированного средства (например, Sapien PrimalScript), а затем скопировать его в окно свойств этапа;

- **Operation System (CmdExec)** (Команда операционной системы) — этот тип задания позволяет просто выполнить какую-либо команду из командной строки операционной системы и проверить для нее код возврата. Это намного менее функциональный тип этапа по сравнению со скриптом ActiveX, но его могут использовать администраторы, которые не знают никаких скриптовых языков программирования. Обычное применение этапов этого типа — копирование файлов в операционной системе, отправка электронной почты из командной строки, подключение сетевых дисков и т. п.;

- **Transact-SQL Script (T-SQL)** (Скрипт Transact-SQL) — это, вероятно, самый распространенный тип этапа (выбирается по умолчанию). Как понятно из названия, он позволяет выполнить на сервере набор команд Transact-SQL. На самом деле, как и при помощи скриптов ActiveX, в заданиях этого типа тоже можно сделать абсолютно все. Доступ к обычным

объектным моделям Windows из кода Transact-SQL можно получить при помощи хранимых процедур автоматизации (`sp_OACreate`, `SP_OAMethod` и т. п.), а к сборкам .NET можно обратиться при помощи новой возможности SQL Server 2005 — подсистемы CLR integration. Чаще всего этот тип этапа используется для резервного копирования баз данных, проверки целостности с использованием команд DBCC, дефрагментации и перестройки индексов и т. п.;

- **SQL Server Integration Services Package** (Пакет SQL Server Integration Services) — этот тип этапов позволяет выполнить по расписанию пакет SSIS (DTS). Это новый тип этапа, которого не было в SQL Server 2000 (в нем пакеты DTS можно было запускать из этапа **CmdExec** при помощи утилиты командной строки `dtsrun`). Функциональность пакетов SSIS также очень велика. Подробнее про работу с SSIS будет рассказано в гл. 10;
- **SQL Server Analysis Services Command** (Команда SQL Server Analysis Services) и **SQL Server Analysis Services Query** (Запрос SQL Server Analysis Services) — эти типы задания позволяют выполнить, соответственно, команду или запрос к Analysis Services (ядру баз данных OLAP на SQL Server).

У каждого этапа есть свой набор свойств (в основном очевидных). Рассмотрим только те из них, которые являются общими для большинства типов этапов. На вкладке **General**, кроме параметров, специфичных для данного типа этапа, вы можете настроить параметр **Run as** (Запустить как), который позволяет запустить этап от имени определенной учетной записи (со своими правами). В соответствующем списке будут доступны SQL Agent Service Account (учетная запись службы SQL Server Agent, т. е. этап будет запускаться от имени той учетной записи, под которой работает SQL Server Agent) и учетные записи-прокси, которые созданы для этапа данного типа из контейнера **SQL Server Agent | Proxies**. Для создания учетной записи прокси вам потребуется вначале определить объект Credential из контейнера **Security | Credentials**. Подробно про эти возможности будет рассказано в разд. 8.1.4.

Другие важные свойства для этапов можно определить при помощи вкладки **Advanced**:

- **On success action** (Действие при успехе) и **On failure action** (Действие при сбое) — эти параметры позволяют определить, что должно произойти, соответственно, после успешного или неуспешного (возникла ошибка) выполнения этого этапа. В вашем распоряжении три варианта (они одинаковые как для успешного завершения, так и для неуспешного):
 - **Go to the next step** (Перейти к следующему этапу) — этот вариант используется по умолчанию для успешно завершившихся этапов;

- **Quit the job reporting success** (Выйти из задания, просигнализировав об успешном выполнении) — этот вариант предлагается использовать для последнего этапа, если он завершился успешно;
- **Quit the job reporting failure** (Выйти из задания, просигнализировав об ошибке) — это значение по умолчанию предлагается выбирать при возникновении ошибки в ходе выполнения этапа.

При помощи этих возможностей на практике вы можете создавать сложные алгоритмы выполнения заданий. Для очень сложных алгоритмов рекомендуется использовать возможности скриптовых языков программирования в этапах ActiveX Script или языка SQL в этапах Transact-SQL Script.

Отметим еще один момент. Определенная вами последовательность выполнения этапов будет работать без каких-либо проблем только в заданиях, которые запускаются на выполнение по расписанию. Если вы запускаете задание с несколькими этапами на выполнение вручную, то вне зависимости от определенной вами последовательности для выбора первого этапа будет открываться окно, аналогичное представленному на рис. 8.2. Избежать его появления обычными средствами нельзя. Можно изменить задание, использовав в нем один этап типа ActiveX Script для всех действий, а можно запустить задание средствами SQL-DMO или SMO и указать программными средствами первый этап для выполнения в задании;

- Retry attempts** (Попытки повтора) — сколько раз SQL Server Agent будет пытаться повторить выполнение данного этапа, если его не удалось выполнить сразу;

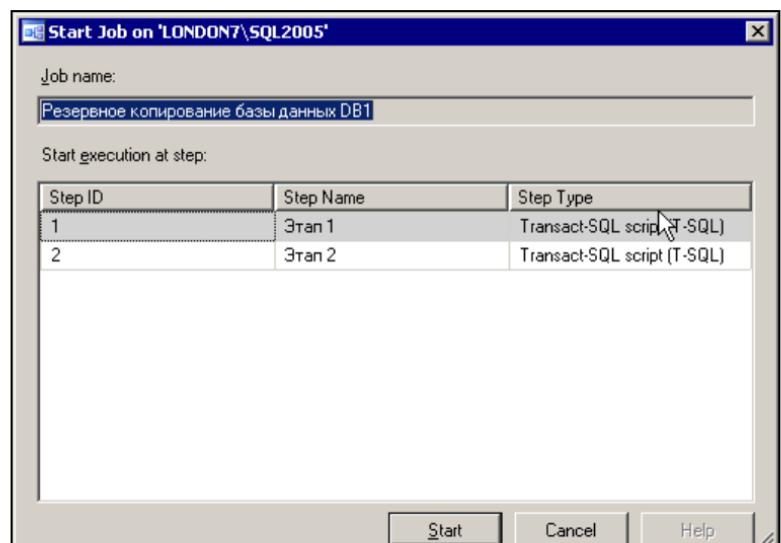


Рис. 8.2. Экран выбора первого этапа при запуске задания

- **Retry intervals** (Интервал повтора) — какая пауза будет сделана SQL Server Agent между попытками повторного выполнения (в минутах);
- **Output file** (Файл вывода) — текстовый файл, в который будет записываться то, что возвращают команды, определенные для этапа. Этот параметр можно использовать для этапов Transact-SQL Script и CmdExec, но не для ActiveX Script (в этапах этого типа такую запись несложно организовать программно, например, при помощи объекта `TextStream` объектной библиотеки Scripting Runtime, имеющейся на любом компьютере).

В принципе, этот параметр можно использовать для простого экспорта данных по расписанию (например, для экспорта результатов выполнения запросов Transact-SQL). Однако для этой цели лучше использовать пакеты SSIS (DTS) — это программное средство специально предназначено для экспорт/импорта данных, и функциональных возможностей в нем значительно больше;

- **Append output to existing file** (Дописать результат к существующему файлу) — обычно этот параметр используется в тех случаях, когда вы хотите накапливать результаты выполнений этапов в каком-то файле;
- **Log to table** (Запротоколировать в таблицу) — параметр позволяет записать протокол выполнения этапа (только протокол, без результатов, возвращаемых, например, командой Transact-SQL) в специальную таблицу `sysjobstepslogs` в базе данных `msdb`. Кнопка **View** (Просмотр) справа от этого флагка позволяет просмотреть не эту таблицу (как можно было бы подумать), а текстовый файл, который вы определили ранее при помощи параметра **Output file**. А кнопка **View** рядом с именем текстового файла в SQL Server 2005 вообще недоступна (она работает только при подключении к SQL Server 2000);
- **Append output to existing entry in table** (Добавлять вывод к существующим записям в таблице) — если этот флагок установлен, то при каждом запуске задания в таблицу `sysjobstepslogs` будут добавляться новые строки. Если он снят, то при запуске задания будут удаляться все существовавшие до этого записи;
- **Include step output in history** (Включать вывод этапа в историю) — если этот флагок установлен, то вывод этапа (например, результат выполнения команд Transact-SQL) будет дописываться в таблицы истории выполнения заданий. С этим параметром нужно быть очень осторожным, потому что размер этого вывода может быть очень большим. В результате чего размер базы данных `msdb` может существенно возрасти.

После того как вы определили нужные этапы задания, можно просмотреть их на вкладке **Steps** окна свойств задания. Обратите внимание, что вы можете

менять этапы местами при помощи стрелок в нижней части экрана, а также определить этап, который будет запускаться первым (при запуске по расписанию).

Обычно следующее, что нужно сделать при создании задания, — подумать, нужно ли вам запускать это задание по расписанию. Если да, то нужно настроить расписание на вкладке **Schedules** (Расписания) свойств задания. Сама настройка расписания вряд ли требует каких-либо комментариев. Отметим только следующие моменты:

- в SQL Server 2005 предусмотрено четыре типа расписания (тип выбирается при помощи списка **Schedule Type** (Тип расписания) в окне свойств расписания):
 - **Recurring** — повторяющееся действие, которое будет выполняться, например, каждый день, или каждую неделю, или каждый месяц;
 - **One time** — действие будет выполнено только один раз;
 - **Start automatically when SQL Server Agent starts** — задание будет запускаться автоматически каждый раз при запуске SQL Server Agent;
 - **Start whenever the CPU become idle** — задание будет запускаться во время простоя центрального процессора. Состояние простоя определяется на вкладке **Advanced** свойств SQL Server Agent;
- чтобы задание сработало в соответствии с настроенным расписанием, необходимо, чтобы служба SQL Server Agent находилась в рабочем состоянии. Проверьте, настроен для нее режим автозапуска;
- с расписаниями можно работать отдельно от заданий. Фактически это отдельный набор объектов. Чтобы открыть список всех имеющихся расписаний (с возможностью создания новых расписаний, изменения существующих и т. п.), можно воспользоваться командой **Manage Schedules** (Управлять расписаниями) контекстного меню для контейнера **SQL Server Agent | Jobs**;
- если у вас уже есть подходящее расписание, но определенное для другого задания, создавать его заново не нужно. Достаточно воспользоваться кнопкой **Pick** (Выбрать) на вкладке **Schedules** и выбрать существующее расписание. Однако необходимо учитывать, что у расписания и у задания должен быть один и тот же владелец, поэтому таким способом можно выбрать только те расписания, владельцы которых совпадают с владельцем задания. Для выбора расписания достаточно выбрать в открывшемся окне **Pick Schedule** (Выбрать расписание) нужную строчку и нажать **OK**;
- для одного задания можно настроить несколько расписаний. Во многих случаях это может быть очень удобно;

- если вам нужно, чтобы на какое-то время расписание перестало действовать (например, во время новогодних праздников проводить резервное копирование базы данных каждый день нет необходимости), то проще всего это расписание отключить. Для этого нужно открыть его свойства и снять флагок **Enabled**.

Возможен другой вариант автоматического запуска задания — не по расписанию, а в ответ на какое-то событие, которое произошло на SQL Server. Например, можно настроить автоматическое выполнение резервного копирования в случае, если место в журнале транзакций закончилось (или его осталось слишком мало). Для отслеживания событий средствами SQL Server Agent применяются предупреждения (*alerts*). Подробно про работу с предупреждениями будет рассказываться в *разд. 8.1.7*. Здесь только отметим, что настроить предупреждение, при срабатывании которого будет автоматически запущено ваше задание, можно на вкладке **Alerts** (Предупреждения) свойств задания.

Кроме того, задания можно запускать и без всякого расписания — вручную. Проще всего это сделать с помощью команды **Start Job** (Запустить задание) контекстного меню задания в Management Studio. Другой вариант — воспользоваться хранимой процедурой `sp_start_job`.

Часто бывает необходимо сделать так, чтобы задание по завершении само отчиталось о своем выполнении. Например, если у вас ночью проводится резервное копирование многих баз данных на нескольких серверах, то вы можете настроить соответствующие задания так, чтобы по окончании резервного копирования каждой базы данных отчет о нем отправлялся на почтовый ящик администратора. Просмотреть почтовые сообщения в почтовом ящике обычно проще, чем смотреть логи резервного копирования на всех серверах.

Настроить параметры "отчета" задания о своем завершении можно при помощи вкладки **Notifications** (Уведомления) свойств задания. На этой вкладке вы можете настроить следующие параметры:

- **E-mail, page и net send** — эти параметры позволяют выбрать объекты операторов (про эти объекты будет рассказано в *разд. 8.1.8*) для отправки предупреждений о выполнении задания соответственно по электронной почте, на пейджер и по сети (средствами службы Messenger). В вашем распоряжении три варианта, когда будет отправляться сообщение:

- **When the job fails** — отправлять предупреждение только тогда, когда при выполнении задания возникла ошибка. Этот вариант выбирается по умолчанию;
- **When the job succeeds** — предупреждение будет отправляться только при успешном выполнении задания;

- **When the job completes** — предупреждение будет отправляться в любом случае;
- Write to the Windows Application event log** (Записывать информацию о выполнении задания в журнал событий приложений Windows) — такое решение можно использовать, например, если журналы событий Windows с разных серверов автоматически переносятся в базу данных или, например, в вашей сети работает приложение, которое автоматически отслеживает появление определенных записей в журналах событий Windows и предпринимает для них какие-либо действия (например, GFI SELM или EventTracker);
- Automatically delete job** (Автоматически удалять задание) — при выборе этого варианта задание после выполнения само себя удалит. Обычно такое решение используется для заданий, которые выполняют разовые операции, например, создание базы данных-копии, в которую будут передаваться данные средствами доставки журналов или репликации.

На вкладке **Targets** (Назначения) свойств задания вы можете определить, на каких именно серверах будет выполняться это задание. Эта возможность доступна только для заданий на серверах, на которых настроен режим мультисерверного выполнения. Подробно про настройку этого режима говорится в разд. 8.1.6.

8.1.4. Безопасность при выполнении заданий

Одна из новых возможностей SQL Server 2005 — очень точная настройка параметров безопасности при выполнении этапов заданий.

В SQL Server 2000 в вашем распоряжении было только два варианта запуска заданий:

- запускать задания с правами учетной записи SQL Server Agent (по умолчанию);
- выбрать другую учетную запись, от имени которой должны были выполняться все задания ActiveX Script и CmdExec. Такую учетную запись можно настроить и средствами Management Studio (на вкладке **Job System** свойств SQL Server Agent), но эта возможность будет доступна только в случае, если вы подключитесь из Management Studio к SQL Server 2000.

В SQL Server 2005 вы можете настроить свою учетную запись для любого этапа любого задания. Таким образом, каждый этап может быть настроен для выполнения от имени учетной записи с минимально необходимыми правами, что обычно и предписывают требования безопасности. Если же вам нет необходимости заниматься такой точной настройкой, что вы вполне можете за-

пускать этапы заданий с правами учетной записи SQL Server Agent, как это обычно делалось в предыдущих версиях SQL Server.

Для того чтобы определить конкретную учетную запись, от имени которой будет выполняться определенный этап задания, нужно выполнить несколько действий.

Первое действие — создать объект Credential. Для этого нужно открыть контейнер **Security | Credentials** в Management Studio и воспользоваться командой **New Credential** (Новая учетная запись) контекстного меню этого контейнера. Откроется окно создания нового объекта Credential. В нем вам потребуется указать:

- Credential name** (Имя учетной записи) — лучше выбрать какое-нибудь значимое имя, чтобы не забыть, для чего был создан этот объект;
- Identity** (Идентификатор) — здесь нужно выбрать локальную или доменную учетную запись Windows. Эта учетная запись должна обладать на локальном компьютере (или в сети) теми правами, которые вы хотите предоставить соответствующему этапу задания;
- Password** (Пароль) и **Confirm Password** (Подтверждение пароля) — это, конечно, пароль для данной учетной записи.

После того, как объект Credential создан, второе действие, которое вам нужно будет выполнить, — создать учетную запись прокси для SQL Server Agent. Сделать это можно из контейнера **SQL Server Agent | Proxies**. Вам потребуется выбрать контейнер для нужного типа этапа и в контекстном меню для вложенного контейнера этого типа воспользоваться командой **New Proxy** (Новая прокси). Откроется окно, аналогичное представленному на рис. 8.3.

В окне **New Proxy Account** необходимо:

- в поле **Proxy name** (Имя прокси) ввести имя создаваемой учетной записи прокси;
- в поле **Credential name** выбрать созданный ранее объект Credential;
- в поле **Description** ввести описание для создаваемой учетной записи (по желанию);
- в списке **Subsystem** (Подсистема) указать типы этапов, для которых можно будет использовать созданную учетную запись прокси.

Осталось выполнить последнее действие — открыть свойства этапа задания и в списке **Run as** на вкладке **General** выбрать созданную вами учетную запись прокси.

Отметим еще два момента:

- для этапов типа ActiveX Script вы можете поменять контекст выполнения (т. е. учетную запись, с правами которой выполняется этот скрипт) прямо

в ходе выполнения скрипта (программными средствами), без использования учетных записей прокси. Пример того, как это можно сделать, легко найти в Интернете, запустив поиск по словосочетанию "vbrunas.vbs";

- для этапов типа Transact-SQL Script вы также можете поменять контекст выполнения в ходе выполнения скрипта. Это можно сделать двумя способами:
 - использовать в коде скрипта конструкцию `EXECUTE AS` (см. разд. 5.5);
 - воспользоваться полем **Run as user** (Запустить от имени пользователя) на вкладке **Advanced** свойств этого этапа. Вы можете выбрать логин, от имени которого будет выполняться этот этап (только если вы сами, как владелец этого задания, обладаете на сервере правами `sysadmin`).

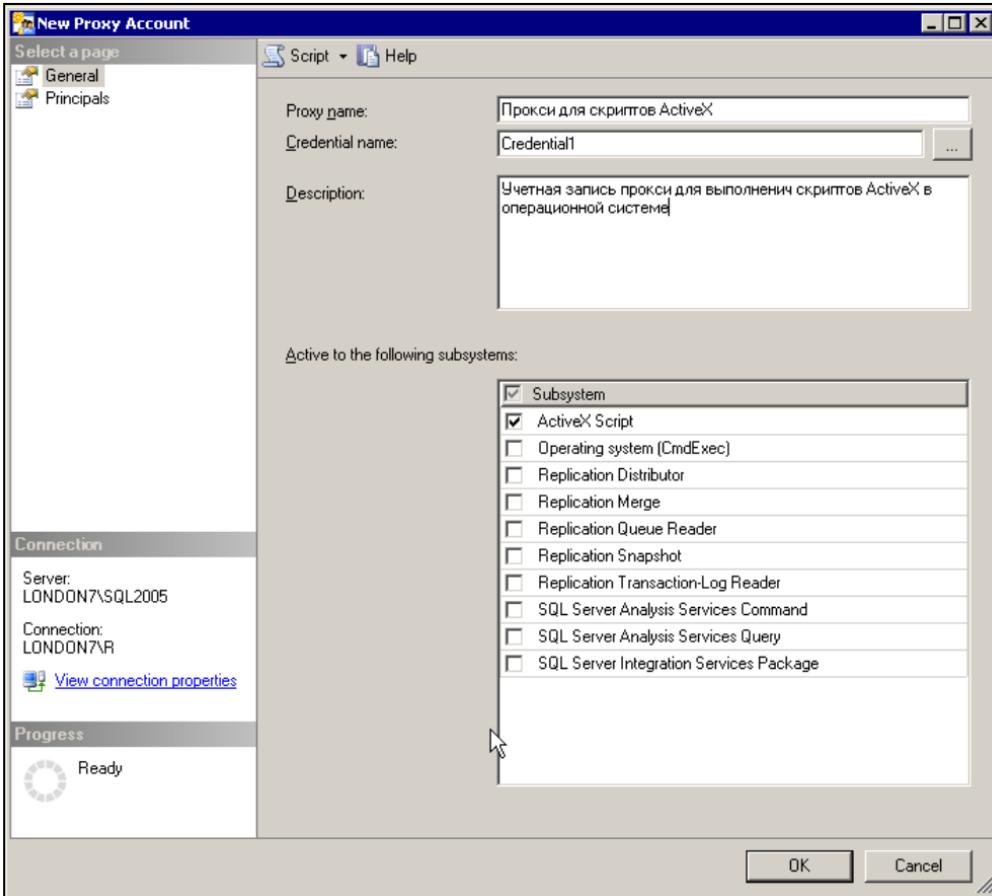


Рис. 8.3. Окно создания новой учетной записи прокси

8.1.5. Просмотр истории выполнения заданий

Так как задания SQL Server Agent чаще всего выполняются по расписанию, то, скорее всего, вам потребуется просматривать историю их выполнения, например, для того, чтобы убедиться, что они выполняются успешно и каких-либо проблем не возникает.

Просмотреть историю выполнения заданий в SQL Server 2005 можно разными способами.

Первый способ — воспользоваться журналами SQL Server Agent. Чтобы просмотреть события, которые относятся к истории выполнения конкретного задания, можно просто воспользоваться командой **View History** (Просмотреть историю) контекстного меню задания. Откроется окно просмотра журналов с настроенным фильтром, аналогичное представленному на рис. 8.4.



Рис. 8.4. Просмотр журнала выполнения задания

Это окно позволяет получить самую полную информацию о выполнении каждого из этапов задания. Если вам нужно понять, отчего возникла ошибка при выполнении, то лучше всего использовать именно этот способ получения информации об истории выполнения заданий.

У этого способа есть только один недостаток: отображается история выполнения только одного задания. Если же заданий у вас несколько десятков, то, вполне вероятно, вам потребуется сводная информация по всем заданиям, чтобы можно было сразу понять, с какими заданиями возникли проблемы.

Для получения такой сводной информации проще всего использовать новое средство SQL Server 2005, которое называется Job Activity Monitor (Монитор активности заданий). Чтобы им воспользоваться, достаточно найти объект **Job Activity Monitor** под контейнером **SQL Server Agent** в Management Studio и в его контекстном меню воспользоваться командой **View Job Activity** (Просмотреть активность заданий). Вам будет предоставлена информация, аналогичная показанной на рис. 8.5.

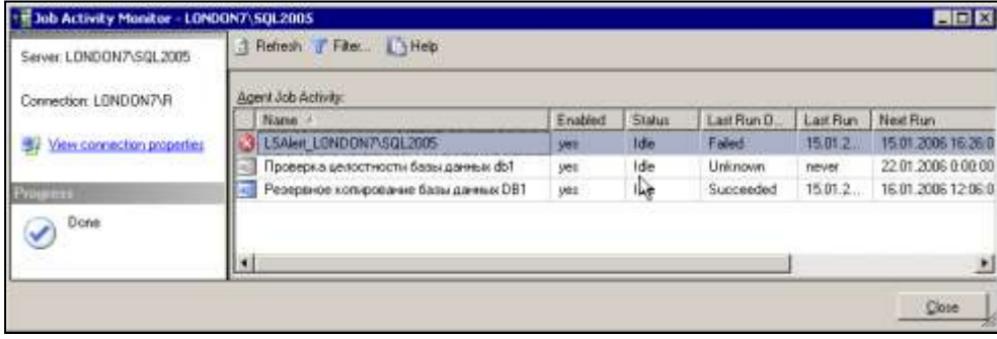


Рис. 8.5. Окно Job Activity Monitor

При помощи этого окна вы можете быстро получить сводную информацию по всем заданиям и, например, найти те задания, при выполнении которых возникли проблемы. Из этого окна вы можете также открыть свойства задания или просмотреть подробную историю его выполнения.

Информацию для представления на экране Job Activity Monitor берет из таблицы `sysjobactivity` базы данных `msdb`. Вы можете обращаться к этой таблице и напрямую. Однако это не очень удобно, поскольку информация в этой таблице нормализована, и, например, задания в ней представлены по их системным идентификаторам, а не по именам. Получить самую полную информацию из этой таблицы (более подробную, чем средствами Job Activity Monitor) можно при помощи хранимой процедуры `sp_help_jobactivity`.

Еще один способ получения информации об истории выполнения заданий — воспользоваться возможностями этапов заданий. Для некоторых типов этапов (например, CmdExec и Transact-SQL Script) можно настроить запись возвращаемых результатов в файл или протоколирование информации о выполнении в таблицу `sysjobstepslogs`. Эти настройки производятся при помощи вкладки **Advanced** свойств этапа (см. разд. 8.1.3).

8.1.6. Мультисерверные задания

Задания SQL Server Agent — это достаточно мощное средство для автоматизации административных задач. Но если вы работаете на большом предпри-

ятия с десятками серверов SQL Server, то скорее всего захотите сделать еще один логичный шаг: вместо того, чтобы настраивать систему заданий (часто однотипных) на каждом сервере отдельно, настроить их все вместе на одном сервере. Этот сервер должен будет "раздавать" задания для выполнения на других серверах и централизованно собирать отчеты об их выполнениях. Для этого вам потребуется настроить возможность выполнения мультисерверных заданий и создать такие задания.

Кроме централизации администрирования, мультисерверные задания могут помочь в синхронизации выполнения административных операций. Предположим, что вам нужно произвести выгрузку данных и их обработку на одном сервере и сразу после этого начать загрузку этих данных на другом сервере. Если вы реализуете эти операции при помощи заданий SQL Server Agent, то настроить последовательное выполнение действий на разных серверах проще всего при помощи именно мультисерверных заданий.

В системе мультисерверных заданий используются серверы двух типов:

- *главный сервер (master server)* — сервер, на котором создаются мультисерверные задания. Этот сервер посылает информацию о заданиях и этапах для выполнения на другие серверы и принимает от них отчеты о выполнении;
- *сервер-получатель заданий (target server)* — это сервер, который принимает задания для выполнения от главного сервера и отчитывается перед ним о выполнении.

Обычно главным сервером на предприятии делают один-единственный сервер, который либо совсем не загружен другими задачами, либо загружен не сильно. Все остальные серверы выступают в роли серверов-получателей заданий. При этом каждому серверу-получателю может быть назначен только один главный сервер: несколько "хозяев" для одного сервера настроить невозможно.

Перед настройкой системы мультисерверных заданий необходимо подготовиться:

1. Вначале нужно проверить, от имени каких учетных записей работают службы самого SQL Server и SQL Server Agent на главном сервере и на серверах-получателях заданий. Они обязательно должны работать от имени доменных учетных записей (иначе серверы не смогут взаимодействовать) и должны обладать необходимыми правами друг на друга. Лучше всего, чтобы учетные записи SQL Server и SQL Server Agent главного сервера обладали административными правами на сервере-получателе заданий, и наоборот. В противном случае настройка прав может оказаться очень сложной.

2. Следующее действие — подключение из Management Studio к главному серверу и выполнение команды **Multi Server Administration | Make this a Master** (Мультисерверное администрирование | Сделать этот сервер главным) контекстного меню службы SQL Server Agent для этого сервера. Откроется мастер настройки главного сервера **Master Server Wizard**.
3. На экране **Master Server Operator** (Оператор главного сервера) необходимо ввести данные оператора мультисерверного администрирования. Это обычный оператор, т. е. запись в адресной книге с информацией об адресе электронной почты, пейджера и сетевым адресом (подробно про работу с операторами будет рассказываться в *разд. 8.1.8*). Единственное его отличие заключается в том, что только этот оператор будет уведомляться о результатах выполнения мультисерверных заданий.
4. Далее на экране **Target Servers** (Серверы-получатели) вам потребуется выбрать те серверы, которые будут назначены серверами-получателями для этого главного сервера. В список серверов-получателей в правой части экрана можно добавить только те серверы, которые известны Management Studio. Если какие-то серверы еще не были зарегистрированы в Management Studio, к ним можно подключиться из этого экрана при помощи кнопки **Add Connection** (Добавить подключение). При нажатии на кнопку **Next** (Дальше) выполнится проверка возможности настройки этих серверов как серверов-получателей.
5. Затем на экране **Master Server Login Credentials** (Логин для подключения к главному серверу) вам потребуется определить, как именно будет производиться подключение сервера-получателя к главному серверу. Если необходимый логин уже существует, вы можете снять единственный флажок на этом экране. Если нужного логина нет или вы сомневаетесь, то флажок нужно сохранить в установленном виде. После этого вам останется прочитать окно сводки и нажать кнопку **Finish** (Завершить), чтобы перевести серверы в режим мультисерверного администрирования.

Если настройка прошла успешно, то после обновления окна **Object Explorer** в Management Studio вы увидите изменения: для службы SQL Server Agent главного сервера в скобках появится надпись **MSX**, а в контейнере **Jobs** появятся еще два контейнера: **Local Jobs** (Локальные задания) и **Multi-Server Jobs** (Мультисерверные задания). На сервере-получателе заданий для службы SQL Server Agent добавится в скобках надпись **TSX: имя_главного_сервера**, кроме того, в контейнере **Jobs** на этом сервере будут отражаться все назначенные ему на главном сервере мультисерверные задания.

Мультисерверные задания создаются из контекстного меню контейнера **Multi-Server Jobs** и выглядят так же, как и обычные, за единственным исключением: на вкладке **Targets** свойств такого задания вы сможете выбрать серверы, на которых эти задания должны выполняться.

Если вы назначили для этапов мультисерверного задания выполнение не с обычными правами учетной записи SQL Server Agent, а от имени специальных учетных записей-прокси (*см. разд. 8.1.4*), то вам придется проделать дополнительные действия:

1. Настроить на сервере-получателе заданий для параметра реестра `\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\Services\SQL Agent\AllowDownloadedJobsToMatchProxyName` значение 1 (по умолчанию он равно 0).
2. Создать на сервере-получателе заданий учетные записи-прокси с точно такими же именами, как на главном сервере (для тех учетных записей-прокси, которые используются в этом задании).

Если вы используете мультисерверные задания, то логично будет для удобства просмотра истории выполнения заданий перенаправить события SQL Server Agent сервера-получателя на главный сервер. Это можно сделать на вкладке **Advanced** свойств SQL Server Agent на сервере-получателе. Если этого не сделать, никакая информация о выполнении заданий на главном сервере собираться не будет (только информация об отправке заданий на сервер-получатель).

Если вы изменили задание на главном сервере, то новое определение задания не будет автоматически загружено сервером-получателем. Чтобы инициировать такую загрузку, нужно использовать хранимую процедуру `sp_post_msx_operator`. Ее параметры зависят от того, как именно изменилось задание.

8.1.7. Работа с предупреждениями

Предупреждения (alerts) — это еще одно важное средство автоматизации работы с SQL Server. Предупреждения позволяют реагировать на события, которые происходят на SQL Server. К таким событиям относятся:

- **события самого SQL Server.** Для удобства любые события SQL Server, которые можно зафиксировать (перехватить), называются ошибками. При этом совершенно не обязательно, чтобы данная ошибка действительно мешала работе сервера: множество ошибок (например, все ошибки с уровнем важности от 0 до 9) являются просто информационными сообщениями о том, что на сервере что-то произошло. Кроме того, пользователь (или разработчик) может сам генерировать ошибку с нужным кодом;
- **выход значения счетчика Системного монитора за указанные вами пределы.** Подробно про счетчики Системного монитора и анализ получаемой с их помощью информации будет рассказываться в *разд. 11.4*;
- **возникновение события WMI.** Про работу с объектной моделью WMI будет рассказываться в *разд. 9.4*.

Работа с предупреждениями производится из контейнера **SQL Server Agent | Alerts**. Чтобы создать новое предупреждение, нужно воспользоваться командой **New Alert** (Новое предупреждение) из контекстного меню этого контейнера. Далее на вкладке **General** свойств создаваемого предупреждения вы можете настроить:

- Name** — имя создаваемого предупреждения;
- Enable** — если этот флажок снят, то предупреждение работать не будет. Обычно это используется для того, чтобы на время отключить оповещение без его удаления;
- Type** — тип предупреждения. От него зависит, что именно будет отслеживать это предупреждение и какие параметры можно будет для него настроить. В вашем распоряжении три варианта:
 - **SQL Server event alert** — предупреждение будет реагировать на встроенные и пользовательские ошибки SQL Server;
 - **SQL Server performance condition alert** — будет отслеживаться выход значений выбранных вами счетчиков производительности в Системном мониторе за указанные вами границы;
 - **WMI event alert** — будут отслеживаться события объектов WMI.

Рассмотрим возможности каждого типа предупреждений.

Чаще всего используются предупреждения типа **SQL Server event alert**. Для них вы можете настроить следующие параметры:

- Database name** (Имя базы данных) — предупреждение будет срабатывать, только если возникла ошибка в определенной базе данных. В вашем распоряжении есть все базы данных на этом сервере и специальное значение **<all databases>** (Все базы данных);
- Alert will be raised on: Error number/Severity** (Предупреждение сработает: При ошибке номер/С важностью) — этот параметр определяет номер ошибки или уровень важности. При возникновении ошибки с этим номером или любой ошибки на указанном вами уровне важности произойдет срабатывание данного предупреждения. Подробнее про номера встроенных и пользовательских ошибок будет рассказано далее в этом разделе;
- Raise alert when message contains** (Срабатывать, если текст ошибки содержит) — проверяет не номер и уровень важности ошибки, а текст сообщения для этой ошибки. Такой вариант рекомендуется выбирать только в том случае, если вы не можете использовать номер ошибки. С точки зрения производительности, использование таких предупреждений — не самый лучший вариант.

Какие именно ошибки чаще всего перехватываются? К сожалению, в справке к SQL Server 2005, в отличие от справки к SQL Server 2000, нет таблиц с номерами ошибок. Вместо этого нам предлагается воспользоваться разделом "Events and Errors Message Center" (Центр сообщений о событиях и ошибках), расположенным на сайте Microsoft (по адресу <http://go.microsoft.com/fwlink/?LinkId=47660>). Однако сводного списка ошибок SQL Server 2005 там тоже нет: по указанному адресу открывается интерфейс для поиска информации по конкретной ошибке. Поэтому приведем здесь краткую информацию по некоторым наиболее часто перехватываемым ошибкам SQL Server 2005:

- 1204 — появление этой ошибки говорит о том, что SQL Server не хватает специальной области оперативной памяти для того, чтобы наложить новые блокировки в оперативной памяти. Для решения этой проблемы можно увеличить объем оперативной памяти на сервере или в некоторых ситуациях можно выполнить хранимую процедуру `sp_configure` с параметром `locks`;
- 1205 — ошибка, свидетельствующая о *взаимоблокировке* (*deadlock*) на сервере. Взаимоблокировки обычно являются следствием плохо продуманных транзакций. В случае появления таких ошибок администратору рекомендуется предъявлять претензии разработчикам, а разработчикам рекомендуется планировать транзакции так, чтобы взаимоблокировок не происходило, или, по крайней мере, перехватывать эту ошибку на уровне приложения;
- 3041 — сбой резервного копирования (по самым разным причинам);
- 3267 — для начала резервного копирования недостаточно системных ресурсов сервера;
- 6103 — произошло аварийное закрытие пользовательского соединения;
- 9002 — это, наверное, наиболее часто перехватываемая ошибка. Она сообщает о том, что закончилось место в журнале транзакций.

Как правило, встроенные ошибки (кроме ошибки 9002) перехватываются только тогда, когда вы знаете, что на вашей системе могут возникнуть определенные проблемы, и хотите, чтобы вас как можно раньше известили об этих проблемах.

Чаще перехватываются пользовательские ошибки (с номером выше 50000). Классический пример применения этой технологии выглядит следующим образом. Предположим, что в базе данных есть очень важная таблица. О любых изменениях в этой таблице должен сразу же уведомляться системный администратор. Для этого он должен:

- определить пользовательскую ошибку;
- создать для этой таблицы триггеры для операций изменения данных;

- поместить в этот триггер оператор RAISERROR, который генерирует данную ошибку;
- настроить предупреждение SQL Server Agent, которое перехватывает ошибку и реагирует на нее (например, извещает администратора по электронной почте).

Конечно, в этой схеме вполне можно обойтись и без ошибок. Можно, например, просто поместить в триггеры команды на отправку электронной почты. Однако использование ошибок и предупреждений позволяет получить выигрыш в надежности, а в некоторых ситуациях это более удобно: например, предупреждения можно отключать на время, а для операторов, которые будут оповещаться, можно настроить часы работы.

В то же время перехват пользовательских ошибок следует использовать только для каких-то важных ситуаций, которые происходят редко, поскольку эта технология требует большого расхода системных ресурсов.

Работа с пользовательскими ошибками может оказаться удобной также при отладке скриптов Transact-SQL.

Отметим также, что в SQL Server 2005 появилась возможность перехвата ошибок из кода Transact-SQL при помощи синтаксической конструкции TRY ... CATCH.

Теперь остановимся подробнее на том, как настроить перехват пользовательских ошибок на практике:

1. Первое, что нужно сделать, — создать пользовательское сообщение. В SQL Server 2000 это можно было сделать и при помощи графического интерфейса, и из кода Transact-SQL, а в SQL Server 2005 — только средствами кода Transact-SQL при помощи хранимой процедуры sp_addmessage, например:

```
sp_addmessage 50001, 16, 'Пользовательская ошибка';
```

Здесь 50001 — это номер пользовательской ошибки (еще раз напомним, что он обязательно должен быть больше 50000, иначе пользовательская ошибка просто не будет работать), 16 — уровень важности (от 0 до 25), 'Пользовательская ошибка' — текст сообщения ошибки. В этом тексте можно использовать параметры, в которые при возникновении ошибки будут подставляться значения (например, имя базы данных, имя объекта в базе данных и т. п.).

2. Далее нужно настроить триггеры с командой RAISEERROR. Соответствующая команда в теле триггера может выглядеть так:

```
RAISERROR (50001, 16, 1) WITH LOG;
```

Параметр WITH LOG нужно обязательно указывать, т. к. в противном случае предупреждение SQL Server 2005 не сможет "поймать" эту ошибку. Это

связано с тем, что предупреждения SQL Server 2005 используют службу событий Windows, а параметр `WITH LOG` как раз и указывает на то, что информация должна быть записана в журнал событий Windows (ошибка попадет в журнал событий приложений с номером 17063, а ваш номер 50001 будет использован только в тексте сообщения). Параметр 16 — это уровень важности ошибки, а 1 — это код состояния ошибки. Практически во всех ситуациях для кода состояния ошибки нужно использовать значение 1. Исключением является ситуация, когда одна и та же ошибка может быть сгенерирована в разных частях кода. В этом случае выяснить, в каком именно участке кода она возникла, можно, использовав для этого параметра оператора `RAISERROR` значение, отличное от 1 (максимально допустимое значение — 127).

3. Затем нужно настроить предупреждение, которое выполнит необходимые действия.

Можно также настроить перехват всех ошибок с определенным уровнем важности (*severity*). Обычно создают предупреждения для всех ошибок с уровнями важности от 19 до 25. Ошибки этих уровней — это так называемые *критические (fatal)* ошибки, которые сам SQL Server исправить не может. Смысл настройки предупреждений для ошибок определенного уровня важности — быстрое предупреждение администратора о проблемах на важной системе.

Предупреждения типа **SQL Server performance condition alert** настраиваются для того, чтобы отслеживать выход за предельные значения для счетчиков производительности Системного монитора. Для таких оповещений можно настроить:

- **Object** (Объект) — имя объекта в Системном мониторе;
- **Counter** (Счетчик) — счетчик для данного объекта, значение которого будет отслеживаться;
- **Instance** (Экземпляр) — экземпляр объекта, для которого будет производиться мониторинг. Например, экземплярами объекта **Databases** будут базы данных на SQL Server;
- **Alert if counter** (Оповестить, если счетчик) — предупреждение сработает, если станет справедливым одно из трех условий:
 - **rises above** — значение счетчика превысит значение, указанное в поле **Value**;
 - **falls below** — упадет ниже этого значения;
 - **becomes equal to** — станет равным указанному значению.

Например, если вы настроили параметры для оповещения так, как показано на рис. 8.6, предупреждение сработает при заполнении журнала транзакций базы данных db1 более чем на 80%.

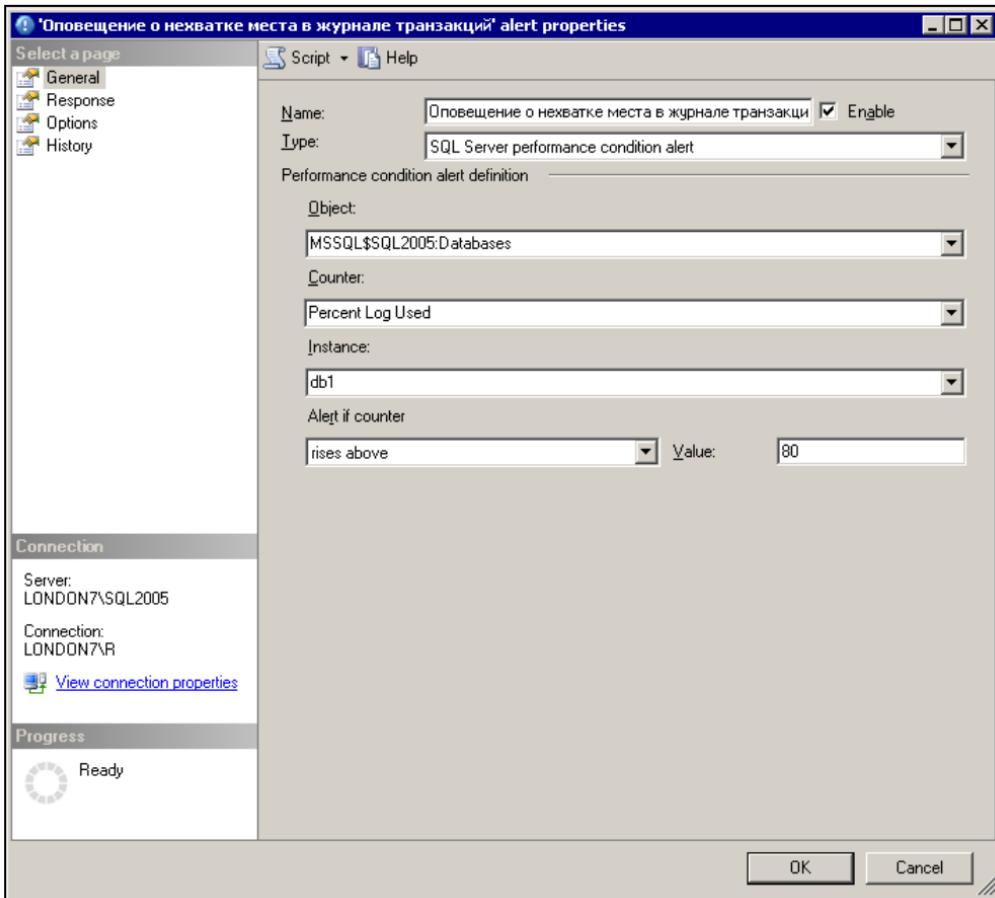


Рис. 8.6. Настройка предупреждения для переполнения журнала транзакций

Отметим, что аналогичные оповещения можно создавать и средствами Windows. Для этого используется консоль **Производительность** в системном меню **Пуск | Программы | Администрирование**. Применение этой консоли по сравнению с SQL Server Management Studio в некоторых случаях может быть даже удобнее, поскольку:

- в одно предупреждение можно добавить сразу несколько счетчиков (в предупреждениях SQL Server Agent допускается проверка только одного значения счетчика);

- можно использовать не только счетчики, относящиеся к данному экземпляру SQL Server (как в предупреждениях SQL Server Agent), но также счетчики для других экземпляров SQL Server и счетчики операционной системы.

Про работу с WMI будет рассказано в гл. 9. Здесь отметим только, что это самый мощный тип предупреждения. При помощи предупреждений этого типа можно перехватить и события SQL Server, и изменения значений счетчиков производительности, а также изменения в сотнях объектов SQL Server и операционной системы. Например, при применении предупреждений этого типа можно отреагировать на запуск или завершение работы приложения, появление или исчезновение файла из каталога, изменение размера файла и т. п.

На вкладке **Response** (Отклик) свойств предупреждения вы можете настроить реакцию на событие, на которое настроено это предупреждение. В вашем распоряжении есть возможность выполнить задание SQL Server Agent или оповестить оператора.

При помощи вкладки **Options** (Настройки) вы можете указать дополнительные параметры для реакции на события, например, можно определить, будет ли передаваться операторам полная информация о событии при отправке уведомления по электронной почте, пейджеру или сети, указать дополнительный текст для сообщения, которое будет передаваться оператору. При помощи раздела **Delay between responses** (Задержка между откликами) вы можете указать задержку между выполнением действий, определенных как реакция на событие. По умолчанию используется значение 0, т. е. действия будут производиться с минимально возможной задержкой.

Последняя вкладка свойств предупреждения — **History** (История). На ней вы можете просмотреть информацию о том, когда в последний раз сработало предупреждение, когда в последний раз были выполнены действия, определенные в качестве реакции, а также общее количество срабатываний предупреждения. При помощи флажка **Reset Count** (Сброс счетчика) можно сбросить этот счетчик предупреждений.

Историю срабатывания предупреждений проще всего просматривать при помощи журналов событий SQL Server. Обратите внимание, что информация записывается именно в журналы SQL Server, а не SQL Server Agent.

8.1.8. Работа с операторами

Последний элемент автоматизации административных операций средствами SQL Server Agent — это *операторы*. Операторы можно представить просто как записи в адресной книге на SQL Server. На адреса, определенные в этой

адресной книге, можно настраивать отправку информации о выполнении заданий и срабатывании предупреждений.

Создание операторов производится из контекстного меню для контейнера **SQL Server Agent | Operators**. Для каждого оператора можно определить:

- Name** — максимальная длина имени составляет 128 символов;
- Enabled** — так же, как для заданий и предупреждений, этот флагок используется для временного отключения оператора в случае необходимости;
- E-mail name** (Адрес электронной почты) — иногда вместо адреса конкретного пользователя удобно использовать список рассылки или совместно используемый почтовый ящик;
- Net send address** (Адрес для отправки сетевых сообщений) — эти сообщения передаются при помощи службы Messenger. Здесь можно использовать имя учетной записи Windows, или имя/IP-адрес рабочей станции администратора;
- Pager e-mail name** (Адрес электронной почты для пейджера) — для использования пейджера вам потребуется настроить соответствующий шлюз на почтовом сервере.

В нижней части вкладки **General** свойств оператора вы можете настроить расписание работы этого оператора. Но оно будет применяться только для сообщений, отправляемых на пейджер.

На вкладке **Notifications** свойств оператора можно выбрать предупреждения и задания. При срабатывании этих предупреждений или выполнении заданий администратору будут отправляться сообщения. Назначить предупреждения и задания оператору можно также из свойств предупреждений и заданий.

8.2. Настройка электронной почты в SQL Server 2005

8.2.1. Обзор возможностей SQL Server 2005 для работы с электронной почтой

В реальных задачах SQL Server очень часто приходится взаимодействовать с серверами электронной почты. Обычно сервер электронной почты используется для автоматической отправки сообщений сервером SQL Server. Например, в рамках скрипта Transact-SQL можно выполнить запрос и сразу отправить по электронной почте результаты его выполнения. Можно настроить триггер для таблицы базы данных так, чтобы при внесении изменений в эту таблицу информация об изменениях отправлялась по электронной почте. По

желанию разработчика SQL Server 2005 в ходе выполнения скрипта может обратиться к почтовому ящику и воспользоваться информацией из содержащихся в нем сообщений.

Электронная почта очень активно используется и при автоматизации административных операций. Например, отчеты о выполнении заданий, информация о срабатывании предупреждений может отправляться по электронной почте на адреса операторов.

В SQL Server 2005 с электронной почтой можно работать тремя способами:

- **средствами Database Mail** (другое название — SQLMail) — это новая подсистема для работы с электронной почтой, которой не было в предыдущих версиях SQL Server. Она ориентирована на использование протокола SMTP. Это наиболее функциональный и рекомендуемый способ работы с электронной почтой;
- **средствами SQLMail** — та же подсистема, что была и в предыдущих версиях SQL Server. Microsoft обещает исключить эту подсистему в будущих версиях SQL Server, поэтому в новых разработках ориентироваться на нее не стоит. В этой подсистеме используются протокол MAPI и почтовый профиль Outlook;
- **средствами операционной системы и нестандартными возможностями SQL Server**. Например, электронную почту можно отправлять при помощи объекта `CDO.Message` и хранимых процедур `SP_OACreate`, `SP_OASetProperty`, `SP_OAMethod`. Другая возможность — использовать утилиту для отправки электронной почты из командной строки и запускать ее при помощи расширенной хранимой процедуры `xp_cmdshell`. Этот способ сложнее, но в некоторых ситуациях он может оказаться более удобным.

Отметим еще несколько моментов.

Настройку электронной почты для служб SQL Server и SQL Server Agent вам придется производить отдельно. Чаще всего для этих служб используются одни и те же настройки, а также один и тот же почтовый ящик. Однако вы вполне можете использовать разные настройки (и даже разные подсистемы) и разные почтовые ящики. Далее мы рассмотрим настройку каждой подсистемы как для использования SQL Server, так и SQL Server Agent.

Если вы собираетесь производить "массовые рассылки" электронной почты, то, возможно, есть смысл использовать специальный компонент SQL Server 2005, который называется Notification Services. Обычное применение этого компонента — уведомлять подписчиков, если в какой-то таблице произошли изменения. Например, появился новый продукт в продаже, или изменились цены на акции, или в футбольном матче забили гол — в общем, произошло любое событие, о котором вас попросили известить. В SQL Ser-

ver 2000 Notification Services нужно было скачивать с сайта Microsoft и установить отдельно. В SQL Server 2005 он входит в состав компонентов сервера.

8.2.2. Работа с Database Mail

Database Mail (другое название — *SQLMail*) — это новая подсистема для организации взаимодействия SQL Server 2005 и SQL Server Agent с почтовым сервером. Именно эту подсистему рекомендуется использовать с SQL Server 2005. По сравнению с традиционной подсистемой *SQLMail*, которая использовалась в предыдущих версиях и оставлена в SQL Server 2005 для обеспечения обратной совместимости, *Database Mail* обладает рядом преимуществ:

- *Database Mail* использует для взаимодействия с почтовым сервером стандартный протокол SMTP. Это значит, что теперь SQL Server стандартными средствами может взаимодействовать с любым сервером электронной почты, установленным на предприятии, а не только с Exchange Server, как это было с SQL Server 2000;
- *Database Mail* работает в отдельном процессе по отношению к процессу SQL Server 2005. Это значит, что если с этой подсистемой произойдут какие-то неполадки, на обычную работу пользователей с SQL Server они не повлияют;
- *Database Mail* использует асинхронный режим доставки. То есть скрипт Transact-SQL сможет дать команду на отправку электронной почты и, не дожидаясь ее завершения, продолжить работу;
- для *Database Mail* можно настроить и использовать несколько учетных записей электронной почты и несколько почтовых серверов. Это существенно повышает удобство работы и надежность;
- *Database Mail* полностью совместима с кластерами Microsoft. В случае каких-то неполадок с основным сервером работа этой подсистемы продолжится на втором сервере кластера, как и работа самого SQL Server;
- подсистема *Database Mail* предусмотрена не только для 32-разрядных версий SQL Server, но и для 64-разрядных версий;
- средствами *Database Mail* вы можете настроить максимальный размер файлов во вложении или запретить работу с вложениями определенных типов;
- все настройки *Database Mail* хранятся на самом SQL Server. Теперь нет необходимости настраивать внешние почтовые профили в операционной системе;

- протоколы отправки сообщений, а также копии отправляемых сообщений и вложений могут сохраняться в базе данных msdb. Эту информацию очень удобно использовать для протоколирования и диагностики.

Настройка Database Mail обычно начинается с мастера настройки Database Mail Configuration Wizard. Вы можете запустить его при помощи команды **Configure Database Mail** (Настроить Database Mail) из контекстного меню для контейнера **Management | Database Mail** (Управление | Database Mail) в SQL Server Management Studio. Кратко опишем работу этого мастера.

На экране **Select Configuration Task** (Выбрать задачу по настройке) в вашем распоряжении есть три варианта:

- **Set up Database Mail** (Настроить Database Mail);
- **Manage Database Mail accounts and profiles** (Управление учетными записями и профилями Database Mail);
- **View or change system parameters** (Просмотреть или изменить системные параметры).

Для первоначальной настройки электронной почты вам потребуется установить переключатель в верхнее положение — **Set up Database Mail**. По умолчанию подсистема Database Mail на SQL Server 2005 отключена, и поэтому появится окно сообщения с предложением ее включить. Также включить Database Mail можно при помощи утилиты SQL Server Surface Area Configuration.

Затем откроется экран **New Profile** (Новый профиль), в котором вам потребуется настроить параметры почтового профиля для работы с Database Mail. Нужно будет указать имя почтового профиля, его описание, а также настроить учетные записи для работы по протоколу SMTP. Для одного почтового профиля вы вполне можете настроить несколько почтовых профилей SMTP. В параметрах почтового профиля можно определить используемый SQL Server адрес электронной почты, почтовый сервер (и номер порта для подключения по протоколу SMTP), а также режим аутентификации при подключении к почтовому серверу (рис. 8.7).

На следующем экране Database Mail Wizard, который называется **Manage Profile Security** (Управление безопасностью профилей), вы можете сделать созданные вами почтовые профили общими (*public*) или личными (*private*). Общие почтовые профили доступны для всех пользователей базы данных (по умолчанию для всех пользователей базы данных msdb, но подсистему Database Mail можно настроить и для любой пользовательской базы данных — тогда общий профиль станет доступным для всех пользователей этой базы данных). Личные почтовые профили доступны только тем пользователям, которым явно назначены эти профили на вкладке **Private Profiles** (Личные профили) этого экрана мастера.

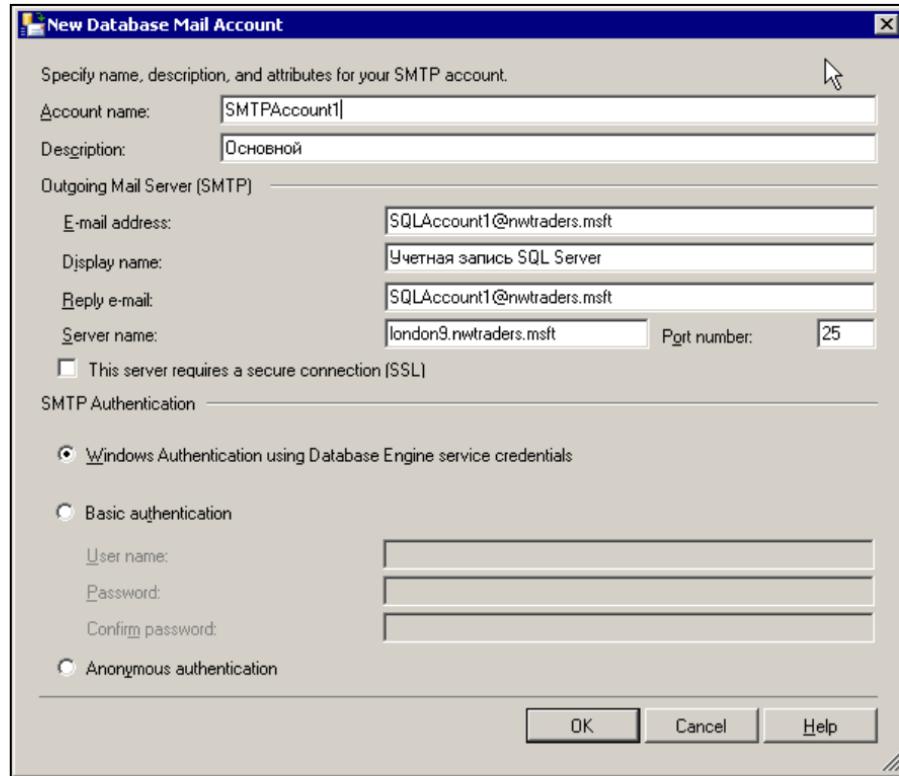


Рис. 8.7. Настройка учетной записи SMTP для Database Mail

Как общий почтовый профиль, так и личный почтовый профиль можно назначить профилем по умолчанию (*default*). Общие почтовые профили станут профилями по умолчанию для всех пользователей, а личные — только для тех пользователей, которым они назначены.

На следующем экране **Configure System Parameters** (Настройка системных параметров) вы можете настроить системные параметры работы Database Mail. Список параметров, которые можно настроить, выглядит так:

- Account Retry Attempts** (Количество повторов для учетной записи) — сколько раз SQL Server будет пытаться повторно обратиться к почтовому серверу, определенному для учетной записи, если он по каким-то причинам оказался недоступным. По умолчанию установлен 1 повтор;
- Account Retry Delay, seconds** (Задержка повторов для учетной записи, в секундах) — время, которое SQL Server будет ждать перед повторным обращением к почтовому серверу. По умолчанию — 60 секунд;
- Maximum File Size** (Максимальный размер файла) — имеется в виду размер всех файлов вложения. По умолчанию установлен примерно 1 Мбайт;

- **Prohibited Attachment File Extensions** (Запрещенные расширения файлов для вложений) — можно запретить пользователям отправлять почтовые сообщения средствами Database Mail с определенными типами вложений. По умолчанию запрещено отправлять файлы exe, dll, vbs, js;
- **Database Mail Executable Minimum Lifetime, seconds** (Время жизни для исполняемого файла Database Mail, в секундах) — сколько времени будет оставаться в памяти процесс Database Mail после завершения отправки всех сообщений в очереди. По умолчанию — 600 секунд;
- **Logging Level** (Уровень протоколирования) — насколько подробно будет протоколироваться работа подсистемы Database Mail. В вашем распоряжении есть три варианта:
 - **Normal** (Обычный) — будут записываться только ошибки;
 - **Extended** (Расширенный) — будут записываться ошибки, предупреждения и информационные сообщения (этот вариант выбирается по умолчанию);
 - **Verbose** (Самый подробный) — наиболее полное протоколирование, будет записываться максимум информации о работе Database Mail.

После этого мастер настроит систему Database Mail на сервере.

Такую же настройку при желании можно выполнить при помощи специальных хранимых процедур базы данных msdb (их названия начинаются с префикса sysmail_...), однако при помощи мастера производить первоначальную настройку и последующее изменение параметров намного проще.

Следующее, что нужно сделать, — предоставить пользователям, которым (или от имени которых) будет отправляться электронная почта средствами Database Mail, необходимые права. Для этого нужно назначить им специальную роль DatabaseMailUserRole в базе данных msdb.

После завершения всех настроек у вас появится возможность использовать хранимые процедуры и представления Database Mail. Хранимые процедуры предназначены для выполнения определенных действий, а представления — для получения информации о работе Database Mail. И хранимые процедуры, и представления находятся в базе данных msdb и принадлежат схеме dbo. Далее представлен перечень хранимых процедур Database Mail с пояснениями:

- sp_send_dbmail — это наиболее часто используемая хранимая процедура Database Mail. Она предназначена для отправки электронной почты. Параметров у этой хранимой процедуры очень много, и разбирать их здесь не будем. Приведем лишь простой пример использования этой хранимой процедуры для отправки электронной почты:

```
EXEC msdb.dbo.sp_send_dbmail @profile_name = 'MailProfile1',
@recipients = 'admininstrator@nwtraders.msft',
@subject = 'Заголовок сообщения', @body = 'Текст сообщения';
```

- `sysmail_delete_mailitems_sp` — эта хранимая процедура позволяет удалить ранее отправленные (успешно или неуспешно) сообщения электронной почты из таблиц Database Mail в базе данных `msdb`. Если вы активно работаете с Database Mail, рекомендуется периодически запускать эту хранимую процедуру, чтобы база данных `msdb` не разрослась слишком сильно;
- `sysmail_delete_log_sp` — эта хранимая процедура также позволяет очистить таблицы базы данных `msdb`. Но при ее использовании удаляются не сами сообщения, а протокол их отправки.

Остальные хранимые процедуры относятся к настройке параметров или к диагностике Database Mail.

Database Mail можно использовать и для получения электронных сообщений службой SQL Server с указанного почтового ящика. Но специальных хранимых процедур для этого не предусмотрено. Для получения электронной почты придется использовать программные средства Service Broker.

Конечно, Database Mail можно использовать и для отправки электронной почты службой SQL Server Agent. Чтобы настроить SQL Server Agent для использования электронной почты, необходимо выполнить следующие действия:

- включить Database Mail и создать необходимый профиль и учетную запись, как это было описано ранее;
- назначить логину, от имени которого SQL Server Agent подключается к SQL Server, роль `DatabaseMailUserRole` в базе данных `msdb`;
- сделать данный профиль профилем по умолчанию для логина SQL Server Agent;
- открыть свойства SQL Server Agent и на вкладке **Alert System** установить флажок **Enable mail profile**. Затем в списке **Mail system** нужно выбрать систему Database Mail, а в списке почтовых профилей — созданный профиль;
- последнее, что осталось сделать, — перезапустить SQL Server Agent, чтобы изменения вступили в силу.

После этого SQL Server Agent сможет средствами Database Mail отправлять информацию о сработавших оповещениях или, например, о результатах выполнения заданий.

Отметим, что в этапах заданий вы можете использовать обходные пути для работы с электронной почтой. Например, из этапа типа Transact-SQL Script

вы можете напрямую использовать хранимую процедуру `sp_send_dbmail`, а из этапа типа ActiveX Script вообще использовать для отправки почты программный объект `CDO.Message`. Конечно, при использовании этих методов специально настраивать SQL Server Agent для использования Database Mail нет необходимости.

Теперь остановимся на том, как можно получить информацию о работе Database Mail. Вне зависимости от того, удалось ли отправить сообщение, хранимая процедура `sp_send_mail` вернет стандартное сообщение "Mail queued" (т. е. письмо поставлено в очередь на отправку). Узнавать о результатах отправки придется другими способами.

Первый способ — просмотреть журналы Database Mail. Для этого можно воспользоваться командой **View Database Mail Log** (Просмотреть журнал Database Mail) контекстного меню контейнера **Management | Database Mail** в Management Studio. Изучив записи в журнале, можно понять, удалось ли отправить сообщение, и если нет, то по какой причине.

Второй способ — воспользоваться специальными представлениями Database Mail в базе данных `msdb`:

- `sysmail_allitems` — главное представление Database Mail. С его помощью можно получить информацию о всех письмах, отправленных средствами Database Mail. Здесь находится информация о получателе, заголовок, начало текста и другие свойства сообщения, время отправки и статус сообщения (отправлено, еще не отправлено, произошел сбой и т. п.). Большинство других представлений показывает только часть записей, которые содержатся в `sysmail_allitems`;
- `sysmail_faileditems` — это представление содержит только те письма, при отправке которых произошел сбой;
- `sysmail_sentitems` — только успешно отправленные письма;
- `sysmail_unsentitems` — только письма, которые еще не были отправлены, но стоят в очереди на отправку;
- `sysmail_mailattachments` — только письма с вложениями;
- `sysmail_event_log` — в этом представлении находится информация из журнала Database Mail (которую можно получить при помощи команды **View Database Mail Log**).

8.2.3. Работа с SQLMail

SQLMail — это подсистема для взаимодействия SQL Server с электронной почтой. Она использовалась в предыдущих версиях SQL Server и оставлена для обеспечения обратной совместимости в SQL Server 2005 (правда, разра-

ботчики предупреждают, что в следующих версиях SQL Server планируется убрать эту подсистему). Тем не менее SQLMail для работы с электронной почтой привычна многим администраторам и разработчикам, и работает вполне надежно на многих предприятиях.

Для тех, кто не сталкивался с этой подсистемой в предыдущих версиях SQL Server, сразу отметим одно важное ограничение: SQLMail может работать только с протоколом MAPI (но не SMTP!). Кроме того, настройку SQLMail нельзя произвести только средствами SQL Server. Для создания почтового профиля MAPI придется использовать внешнюю программу (в большинстве случаев Microsoft Outlook).

Настройка SQL Server для работы с электронной почтой средствами SQLMail состоит из нескольких этапов:

- первое действие, которое нужно сделать — средствами Exchange Server создать почтовый ящик для учетной записи, от имени которой работает SQL Server (если SQL Server Agent работает от имени другой учетной записи, то почтовый ящик потребуется и для него). В Exchange Server 2000 и 2003 эта операция выполняется средствами консоли **Active Directory Users and Computers** (Active Directory — пользователи и компьютеры);
- следующее, что нужно сделать, — установить на компьютер, на котором работает SQL Server, Microsoft Office. Из всего набора программ, которые в него входят, вам нужен только Outlook (любой версии — 97, 2000, XP или 2003). Обратите внимание, что Outlook Express, который есть на любом компьютере Windows, использовать нельзя: эта программа не может работать с протоколом MAPI! Можно использовать как английскую, так и русифицированную версии Outlook;
- следующая операция — настройка почтового профиля MAPI для работы с электронной почтой. Для этого нужно выполнить следующие действия:
 - войти локально на сервер от имени той учетной записи, от имени которой работает SQL Server или SQL Server Agent (в зависимости от того, для кого вы настраиваете почтовый ящик);
 - создать MAPI-совместимый почтовый профиль. Проще всего это сделать при помощи мастера настройки учетных записей Outlook. Если под учетной записью, от имени которой работает SQL Server, Outlook еще не запускался, то при первом запуске Outlook окно мастера запустится автоматически. Если какие-то настройки уже были созданы, то изменить их можно либо из Outlook, либо при помощи консоли **Почта (Mail)** в **Панели управления**.

На первом экране мастера создания учетных записей электронной почты вам потребуется выбрать пункт **Добавить новую учетную запись элект-**

транной почты. На следующем экране, который называется **Тип сервера**, нужно выбрать **Microsoft Exchange Server**. Этот тип учетной записи соответствует протоколу MAPI и только он может использоваться подсистемой SQLMail. Затем в окне настроек Exchange Server вам потребуется ввести имя сервера и имя почтового ящика пользователя. Обратите внимание, что имя почтового ящика пользователя и адрес электронной почты данного пользователя — это не одно и то же. По умолчанию имя почтового ящика на Exchange Server совпадает с именем учетной записи Windows;

- последнее, что осталось сделать, — настроить SQL Server и SQL Server Agent для работы с созданным почтовым профилем.

Для настройки SQL Server нужно воспользоваться свойствами контейнера **Management | Legacy | SQLMail** (Управление | Унаследованное | SQLMail) в SQL Server Management Studio. По умолчанию подсистема SQLMail в SQL Server отключена. При первом обращении к свойствам этого контейнера вам будет предложено ее включить. Единственный параметр, который можно настроить в свойствах контейнера SQLMail, — имя созданного вами почтового профиля. По умолчанию он называется "Outlook".

Чтобы настроить SQL Server Agent для использования SQLMail, в вашем распоряжении — вкладка **Alert System** свойств SQL Server Agent в Management Studio. В списке **Mail system** вам потребуется выбрать **SQLMail**, а в списке **Mail profile** — имя созданного вами профиля (здесь тот же самый профиль выглядит как "Outlook default profile").

После внесения изменений в свойства SQLMail для SQL Server и SQL Server Agent обе эти службы рекомендуется перезапустить.

Для работы с SQLMail используются те же хранимые процедуры, что и в SQL Server 2000:

- `xp_sendmail` — эта хранимая процедура предназначена для отправки сообщений электронной почты, например:

```
EXEC master.dbo.xp_sendmail
@recipients='administrator@nwtraders.msft',
@subject='Заголовок сообщения', @message='Текст сообщения';
```

С ее помощью можно выполнить запрос и сразу отправить результаты выполнения по электронной почте:

```
EXEC master.dbo.xp_sendmail
@recipients='administrator@nwtraders.msft',
@query = 'SELECT * FROM Table1';
```

- `xp_findnextmsg` — возвращает уникальный идентификатор сообщения, которое пришло последним. После этого идентификатор обычно передается хранимым процедурам `xp_readmail` (чтобы прочитать это сообщение) и `xp_deletemail` (чтобы удалить его);

- xp_readmail — считывает сообщение из почтового ящика на Exchange Server. Принимает номер сообщения, который обычно возвращает хранимая процедура xp_findnextmsg;
- xp_deletemail — удаляет сообщение по его номеру;
- sp_proprocessmail — позволяет автоматизировать работу с электронной почтой. Эта хранимая процедура (обычно она запускается по расписанию заданием SQL Server Agent) обращается к почтовому ящику на Exchange Server и пытается выполнить текст каждого непрочитанного сообщения как запрос к SQL Server. Результаты выполнения запроса отправляются в качестве ответа на сообщение (в виде вложенного текстового файла). Эта хранимая процедура справедливо считается небезопасной, но она может быть очень удобной для администрирования SQL Server по электронной почте.

8.2.4. Альтернативные способы работы с электронной почтой SQL Server и SQL Server Agent

Кроме Database Mail и SQLMail, на предприятиях очень часто используются альтернативные средства для работы с электронной почтой служб SQL Server (из кода Transact-SQL) и SQL Server Agent (в этапах заданий). Обычно такие средства основаны на использовании объекта CDO.Message или утилит для отправки электронной почты из командной строки.

Применение объекта CDO.Message очень удобно по следующим причинам:

- библиотека CDO есть на любом компьютере под управлением Windows 2000, XP, 2003;
- не нужно производить никаких настроек для подсистем электронной почты на SQL Server;
- при помощи этого объекта можно подключиться к любому SMTP-совместимому почтовому серверу.

Код для использования этого объекта на языке VBScript очень прост:

```
'Объявляем переменную для нашего сообщения
Dim oMessage
'Создаем объект CDO.Message
Set oMessage = CreateObject ("CDO.Message")
'Настраиваем параметры сообщения
'Кому
oMessage.To = "Administrator@nwtraders.msft"
'От кого
oMessage.From = "Administrator@nwtraders.msft"
```

```
'Заголовок сообщения  
oMessage.Subject = "Проверка"  
'Текст сообщения (может быть текстовым или в формате HTML)  
oMessage.TextBody = "Текст письма"  
'Добавляем вложение  
oMessage.AddAttachment "C:\1.txt"  
'Отправляем сообщение  
oMessage.Send
```

Однако этот вариант с параметрами по умолчанию будет работать только в том случае, если на вашем компьютере установлен Exchange Server 2000 или 2003 или установлен Internet Information Server с настроенной службой SMTP. Причина проста: по умолчанию сообщение просто будет физически помещено в каталог C:\Inetpub\mailroot\Pickup, откуда его и должна забрать служба Exchange Server или IIS. Однако есть более удобный способ отправки сообщения через любой почтовый сервер, который поддерживает протокол SMTP. Для этого перед вызовом метода `Send()` нужно настроить параметры отправки:

```
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/sendusing") = 2
```

В этом параметре по умолчанию используется значение 1, которое означает использование каталога Pickup.

Указать почтовый сервер можно так:

```
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/smtpserver") = _  
"smtp.YourServer.com"
```

Настройка режима аутентификации производится при помощи того же объекта `CDO.Configuration`:

```
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/smtpauthenticate")=1
```

Значение 1 этого параметра означает, что используется базовая аутентификация, значение 0 — без аутентификации (анонимно), значение 2 — аутентификация NTLM.

Имя пользователя и пароль можно передать точно так же:

```
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/sendusername") = _  
"YourLogin@YourDomain.com"
```

```
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/sendpassword") = _  
"Password"
```

Иногда необходимо также определить использование специфического порта (отличного от 25), будет или нет использоваться SSL и время тайм-аута:

```
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/smtpserverport") = 25  
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/smtpusessl") = False  
oMessage.Configuration.Fields.Item  
("http://schemas.microsoft.com/cdo/configuration/smtpconnectiontimeout") = 60
```

После любых изменений, вносимых в конфигурацию, изменения нужно обязательно сохранить:

```
oMessage.Configuration.Fields.Update
```

и только после вызывать метод `Send()`:

```
oMessage.Send
```

Если возникает проблема с кодировками (обычно, если в системе не установлен русский язык), то можно перед отправкой добавить строку вида:

```
oMessage.TextBodyPart.Charset = "windows-1251"
```

Этот код можно напрямую использовать в этапах типа ActiveX Script для заданий SQL Server Agent. При помощи хранимых процедур автоматизации `SP_OACreate`, `SP_OASetProperty`, `SP_OAMethod` его можно использовать и для отправки электронной почты из кода Transact-SQL. Отметим только, что использование хранимых процедур автоматизации в SQL Server 2005 по умолчанию отключено из соображений безопасности. Перед их использованием вам нужно их включить. Проще всего это сделать при помощи утилиты Surface Area Configuration.

Для отправки почтовых сообщений службами SQL Server и SQL Server Agent можно также использовать возможности объектной модели MAPI (основанной на объекте `MAPI.Session`). Она также есть на любом компьютере под управлением Windows 2000, XP и 2003. Эта библиотека позволяет не только отправлять сообщения, но и получать их с сервера, а также выполнять различные операции с сообщениями, которые находятся в почтовом ящике. Однако у этой библиотеки есть серьезное ограничение — как понятно из названия, она может использовать только протокол MAPI. Подробную информацию об использовании этого объекта можно найти в MSDN (www.microsoft.com/msdn).

Другой альтернативный способ работы с электронной почтой связан с использованием утилит для отправки сообщений из командной строки. В Интернете можно найти множество бесплатных утилит для этой цели. Автор обычно использует утилиту `MAIL`. Команда на отправку электронной почты с использованием этой утилиты может выглядеть, например, так:

```
bmail -s smtp.YourServer.com -t komu@nwtraders.msft -f  
otKogo@nwtraders.msft -h -a "Заголовок письма" -b "Текст письма"
```

Чтобы использовать эту утилиту, ее файл нужно поместить в любой каталог, указанный в переменной окружения PATH. После этого соответствующую команду можно использовать в этапе типа CmdExec для задания SQL Server Agent или в коде Transact-SQL при помощи хранимой процедуры xp_cmdshell. Отметим, что эта хранимая процедура также по умолчанию отключена в SQL Server 2005 из соображений безопасности. Включать ее рекомендуется при помощи утилиты Surface Area Configuration.

8.3. Планы обслуживания баз данных

Очень часто для автоматизации операций по обслуживанию SQL Server администраторы на предприятиях используют *планы обслуживания баз данных* (*Database Maintenance Plans*). Планы обслуживания — это специальные объекты, представляющие собой набор самых распространенных операций по обслуживанию, которые можно запланировать для выполнения по расписанию. Например, к этим операциям относятся резервное копирование баз данных и журналов транзакций, проверка целостности баз данных, перестроение индексов, обновление статистики и т. п. Все эти операции можно выполнить и другими способами, например, при помощи заданий SQL Server Agent. Планы обслуживания просто предоставляют удобное средство для управления этими операциями и получения отчетов об их выполнении.

В SQL Server 2005 планы обслуживания изменились очень сильно по сравнению с предыдущими версиями SQL Server. Если в SQL Server 2000 план фактически представлял собой набор заданий SQL Server Agent, то в SQL Server 2005 для каждого плана создается отдельный пакет SSIS (SQL Server Integration Services — подсистема SQL Server, которая раньше называлась DTS). Поэтому для понимания того, как устроены пакеты SSIS, и для освоения работы с ними необходимо знакомство с SSIS. Работа с SSIS подробно будет рассматриваться в гл. 10. Там же будут рассмотрены задачи (*tasks*), которые используются для выполнения операций в планах обслуживания баз данных. А в этом разделе познакомимся только с мастером создания планов обслуживания баз данных и основными возможностями планов.

Работа с планами обслуживания баз данных производится из контейнера **Management | Maintenance Plans** (Управление | Планы обслуживания) в Management Studio. Планы обслуживания можно создавать двумя способами:

- при помощи мастера, который запускается при помощи команды **Maintenance Plan Wizard** (Мастер планов обслуживания) контекстного меню для контейнера **Maintenance Plans**;

- средствами графического интерфейса SSIS. Графический интерфейс для создания планов обслуживания можно запустить из Management Studio (команда **New Maintenance Plan** (Новый план обслуживания) контекстного меню для контейнера **Maintenance Plans**), а можно и обычным для SSIS способом при помощи консоли SQL Server Business Intelligence Development Studio. Графический интерфейс предоставляет большую функциональность по сравнению с мастером, однако использовать его сложнее.

Графический интерфейс для работы с пакетами SSIS будет подробно рассмотрен в гл. 10. Здесь остановимся только на тех возможностях планов обслуживания баз данных, которые доступны при помощи мастера.

Создание планов обслуживания баз данных при помощи мастера выглядит следующим образом:

1. На первом экране мастера, который называется **Select a Target Server** (Выберите сервер-получатель), вам нужно указать имя создаваемого плана обслуживания, имя сервера, на котором будут выполняться операции по обслуживанию, и режим аутентификации при подключении. Обратите внимание, что на одном сервере вы вполне можете создать планы обслуживания, которые будут выполняться на других SQL Server вашего предприятия. Таким образом, можно централизовать выполнение административных операций и сбор отчетов сразу для множества серверов.
2. На втором экране мастера, который называется **Select Maintenance Tasks** (Выберите задачи по обслуживанию), вы можете определить задачи, которые будут выполняться в рамках данного плана. В вашем распоряжении есть следующие задачи (конкретные параметры для них настраиваются на следующих экранах мастера):
 - **Check database integrity** (Проверить целостность базы данных) — выполнить к базе данных команду DBCC CHECKDB;
 - **Shrink database** (Сжать базу данных) — уменьшить размер файлов базы данных при помощи команды DBCC SHRINKDATABASE;
 - **Reorganize index** (Реорганизовать индекс) — реорганизовать индекс (или множество индексов, например, для всех таблиц и представлений) командой ALTER INDEX ... REORGANIZE;
 - **Rebuild index** (Перестроить индекс) — перестроить индекс или индексы командой ALTER INDEX ... REBUILD;
 - **Update statistics** (Обновить статистику) — обновить статистку для всех или только для указанных таблиц и представлений при помощи команды UPDATE STATISTICS;

- **Clean up history** (Очистить историю) — удалить историю резервного копирования, выполнения заданий SQL Server Agent и выполнения планов обслуживания из таблиц базы данных msdb. Удаление будет производиться с помощью хранимых процедур `sp_delete_backuphistory`, `sp_purgejobhistory` и `sp_maintplan_delete_log` соответственно;
 - **Execute SQL Server Agent Job** (Выполнить задание SQL Server Agent) — возможность выполнить указанное вами задание SQL Server Agent. Для этой цели в плане обслуживания будет использоваться хранимая процедура `sp_start_job`;
 - **Backup Database (Full) и Backup Database (Differential)** — выполнить в рамках плана обслуживания соответственно полное или разностное резервное копирование указанных вами баз данных. Для этого, конечно, используется команда `BACKUP DATABASE`;
 - **Backup Database (Transaction Log)** — выполнить резервное копирование журнала транзакций для указанных вами баз данных. Используется команда `BACKUP LOG`.
3. На следующем экране мастера, который называется **Select Maintenance Task Order** (Выберите порядок выполнения задач по обслуживанию), вы можете выбрать последовательность выполнения выбранных вами операций. Для этой цели предназначены кнопки **Move Up** (Перемещение вверх) и **Move Down** (Перемещение вниз) для смещения задачи по списку.
 4. После этого начинаются экраны для настройки конкретных параметров выполнения выбранных вами заданий. Подробно останавливаться на их не будем. Отметим только, что используются те же графические экраны, которые применяются для выполнения аналогичных операций (например, резервного копирования) в Management Studio.
 5. После окончания настройки заданий откроется экран **Select Plan Properties** (Выберите свойства плана). Однако на этом экране вам потребуется выбрать не свойства плана, а расписание его выполнения. По умолчанию задано **Not scheduled (On demand)** (Без расписания (по запросу)). Это значит, что никакого расписания для плана не выбрано, и запускать его на выполнение придется вручную. Если вы настроите расписание, то в списке заданий SQL Server Agent будет создано новое задание с единственным этапом типа SQL Server Integration Services Package. Этот этап будет запускать на выполнение пакет SSIS, представляющий созданный вами план обслуживания.
 6. На предпоследнем экране мастера, который называется **Select Report Options** (Выберите параметры отчета), вы можете настроить параметры отчета, который будет создан при выполнении плана. В вашем распоряже-

ни есть два варианта: создавать отчеты в виде файлов на диске (в этом случае вам потребуется указать каталог) или отправлять созданные отчеты по электронной почте (нужно будет указать оператора SQL Server Agent).

- На последнем экране мастера вы можете просмотреть выбранные вами параметры и нажать кнопку **Finish** (Завершить) для создания плана обслуживания.

Из контекстного меню для созданного вами плана обслуживания в контейнере **Maintenance Plan** (План обслуживания) вы можете:

- View History** (Просмотреть историю) — получить информацию об истории выполнения данного плана (в виде журнала событий SQL Server Agent);
- Modify** (Изменить) — открыть пакет SSIS для плана выполнения, чтобы просмотреть его или внести необходимые изменения;
- Execute** (Выполнить) — запустить план на выполнение;
- Rename** (Переименовать) и **Delete** (Удалить).

Отчет о результатах выполнения планов обслуживания баз данных создается в виде текстового файла (рис. 8.8). Если такой отчет вас не устраивает, то придется создавать его самостоятельно, используя информацию из таблицы sysdbmaintplan_history в базе данных msdb.

```

MaintenancePlan1_20060118165645.txt - Блокнот
Файл Правка Формат Вид Справка
Microsoft(R) server Maintenance utility (unicode) version 9.0.1399
Report was generated on "LONDON7\SQL2005".
Maintenance Plan: MaintenancePlan1
Duration: 00:00:33
Status: Warning: One or more tasks failed.
Details:
Check database integrity (LONDON7\SQL2005)
Check Database integrity on Target server connection
Databases: db1
Include indexes
Task start: 18.01.2006 16:56.
Task end: 18.01.2006 16:56.
SUCCESS
Command:USE [db1]
GO
DBCC CHECKDB WITH NO_INFOMSGS
GO

Shrink Database (LONDON7\SQL2005)
Shrink database on Target server connection I
Databases: db1
Limit: 50 MB
Free space: 10 %
Task start: 18.01.2006 16:56.
Task end: 18.01.2006 16:56.
SUCCESS

Reorganize Index (LONDON7\SQL2005)

```

Рис. 8.8. Пример отчета о выполнении плана обслуживания баз данных

Отметим еще два момента, связанных с планами обслуживания баз данных:

- в SQL Server 2005 вы можете использовать планы обслуживания старых версий SQL Server. Однако они будут доступны только в том случае, если вы обновляли до SQL Server 2005 сервер предыдущей версии, в котором был создан такой план. Найти его можно будет в контейнере **Management | Legacy | Database Maintenance Plans** (Управление | Унаследованное | Планы обслуживания баз данных). Создавать планы старого образца в SQL Server 2005 невозможно;
- вместо планов обслуживания баз данных можно использовать утилиту командной строки `SQLMaint`. Эта утилита позволяет выполнить те же операции по обслуживанию баз данных SQL Server и генерировать отчеты о выполнении как в текстовом формате, так и в формате HTML.

Задание для самостоятельной работы 8.1. Применение заданий, предупреждений и операторов

Задание:

1. Создайте на сервере `имя_вашего_сервера\SQL2005` пользовательское сообщение об ошибке со следующими параметрами:
 - номер — 50005;
 - уровень важности — 16;
 - текст сообщения — "Возникла ошибка";
 - обязательная запись в журнал событий Windows.
2. Создайте на вашем сервере оператора с именем `Operator_имя_вашего_компьютера`. Все сообщения, которые отправляются этому оператору, должны передаваться на консоль вашего компьютера при помощи команды `net send`.

Примечание

Не забывайте, что по умолчанию в Windows Server 2003 служба `Messenger`, которая ответственна за прием таких сетевых сообщений, отключена. Для того чтобы оператор смог получать сообщения по `net send`, эту службу необходимо включить.

3. Настройте на том же сервере предупреждение `MyAlert`. Оно должно перехватывать вашу пользовательскую ошибку и сообщать о ее возникновении оператору.

4. Создайте задание SQL Server Agent, которое будет с интервалом в 5 минут генерировать данную пользовательскую ошибку. Называться это задание должно JobError50005. Убедитесь, что все созданные вами элементы автоматизации работают в соответствии с заданием, а затем отключите задание SQL Server Agent.

Решение:

К пункту 1 задания — создание пользовательской ошибки:

Примечание

Большинство действий, которые описываются далее, можно выполнить как при помощи команд Transact-SQL, так и средствами графического интерфейса SQL Server Management Studio. В этом решении будет приводиться только программный код Transact-SQL. Информацию об использовании графического интерфейса можно найти в разд. 8.1.

1. Команда на создание пользовательской ошибки может выглядеть так:

```
USE master;
GO
sp_addmessage @msgnum = 50005, @severity=16, @msgtext = 'Возникла
ошибка', @with_log = 'TRUE';
```

К пункту 2 задания — создание оператора:

1. Программный код для создания оператора может быть таким (замените LONDON2 на имя вашего компьютера):

```
USE msdb;
GO
EXEC sp_add_operator @name='Operator_LONDON2',
@netsend_address='LONDON2';
```

К пункту 3 задания — создание предупреждения:

1. Соответствующий код может выглядеть так:

```
USE msdb;
GO
EXEC msdb.dbo.sp_add_alert @name='MyAlert', @message_id=50005;
GO
EXEC msdb.dbo.sp_add_notification @alert_name='MyAlert',
@operator_name='Operator_LONDON2', @notification_method = 4;
```

К пункту 4 задания — создание задания:

1. Соответствующий код может быть таким:

```
USE msdb;
GO
```

```
-- Создаем пустое задание
EXEC msdb.dbo.sp_add_job @job_name='JobError50005';
GO
-- Определяем, на каком сервере оно будет выполняться
-- Не забудьте заменить 'LONDON\SQL2005' на имя вашего сервера
EXEC msdb.dbo.sp_add_jobserver @job_name='JobError50005',
@server_name = 'LONDON2\SQL2005';
GO
-- Создаем единственный этап типа TSQL
EXEC msdb.dbo.sp_add_jobstep @job_name='JobError50005',
@step_name='Step1', @subsystem='TSQL', @command=
'RAISERROR (50005, 16, 1)';
GO
-- Создаем "пятиминутное" расписание, которое начнет работать сегодня
DECLARE @thisDay As varchar(8);
SELECT @thisDay = CONVERT(varchar(8),GETDATE(), 112);
EXEC msdb.dbo.sp_add_schedule
@schedule_name='Schedule1',
@freq_type=4,
@freq_interval=1,
@freq_subday_type=4,
@freq_subday_interval=5,
@active_start_date = @thisDay;
GO
-- Назначаем это расписание заданию
EXEC msdb.dbo.sp_attach_schedule @job_name = 'JobError50005',
@schedule_name = 'Schedule1';
```



Рис. 8.9. Пример отчета о выполнении плана обслуживания баз данных

Если все сделано правильно, то каждые 5 минут на вашем компьютере будет появляться окно сообщения, аналогичное представленному на рис. 8.9.

- Чтобы отключить задание, можно выполнить команду:

```
USE msdb;
GO
EXEC msdb.dbo.sp_update_job @job_name='JobError50005', @enabled=0;
```



ГЛАВА 9

Выполнение административных операций при помощи объектных моделей SMO, SQL-DMO и WMI

9.1. Применение скриптов для выполнения административных операций

В предыдущей главе рассматривались стандартные средства для автоматизации административных операций, такие как задания, предупреждения, операторы, планы обслуживания баз данных и т. п. Но часто бывает так, что возможностей этих средств недостаточно. Например, вам нужно с определенной периодичностью синхронизировать логины SQL Server и учетные записи Windows (или список пользователей в каком-то приложении). Другой пример — при выполнении операции вам нужно проверять значения в текстовом файле конфигурации или реестре. Проблемы возникают при выполнении многих нестандартных задач, например, при организации взаимодействия SQL Server с внешними приложениями, другими серверами баз данных и источниками данных, с различными компонентами операционной системы. Такие задачи решить стандартными средствами SQL Server сложно или даже невозможно. На помощь здесь могут прийти административные скрипты.

Есть еще одна ситуация, когда знание скриптов может пригодиться. Существуют очень важные системы, остановка которых даже на некоторое время может привести к серьезным неприятностям (например, биллинг телекоммуникационного оператора, торговая система биржи, операционный день банка и т. п.). Очень часто для таких систем вообще запрещается выполнение каких-либо операций по администрированию на графическом экране. Разрешенная последовательность действий выглядит так:

1. Администратор создает скрипт (либо на Transact-SQL, либо на другом скриптовом языке, например, на VBScript или JavaScript).

2. Этот скрипт проверяется на запасной системе.
3. Он помещается в специальный архив для протоколирования.
4. Скрипт применяется к рабочей системе.

При приеме на работу администраторов таких систем знание скриптов является практически обязательным.

Конечно, при работе с SQL Server традиционно используются скрипты Transact-SQL. Во многих случаях они незаменимы. Однако встречаются такие ситуации, когда возможностей Transact-SQL недостаточно, например:

- вам необходимо интегрировать запуск команд Transact-SQL на SQL Server с выполнением действий или отслеживанием событий в операционной системе. Например, вам нужно сделать что-то сразу при появлении файла в определенном каталоге, произвести резервное копирование базы данных сразу по завершении работы приложения в операционной системе и т. п.;
- вам необходимо выполнять похожие операции одновременно на нескольких серверах SQL Server;
- вам нужно реализовать немедленный отклик на события, которые происходят на SQL Server. В SQL Server для этого используются триггеры (в SQL Server 2005 кроме обычных триггеров можно использовать также триггеры для команд DDL) и предупреждения SQL Server Agent, но возможностей этих средств часто недостаточно;
- не хватает синтаксических и функциональных возможностей Transact-SQL. Например, в Transact-SQL очень сложно реализована работа с классами и объектами, существуют большие ограничения для типов данных (больше всего мешает ограничение в 8000 байт для типа `char` и то, что нельзя объявить переменные типа `text`, `ntext` и `image`), а также нет многих удобных функций, привычных для других языков программирования.

В этих ситуациях многие сложные проблемы могут быть решены очень просто, если выйти за пределы возможностей Management Studio и Transact-SQL. Вместе с SQL Server 2005 поставляются наборы библиотек для специализированных объектных моделей SMO, SQL-DMO и поставщика WMI для SQL Server. С их помощью можно автоматизировать выполнение любых операций на SQL Server 2005. Кратко опишем каждую из этих библиотек:

- **SMO (SQL Management Objects)** — это .NET-совместимая объектная модель, которая обеспечивает полную функциональность при работе с SQL Server. Она включает в себя специальный поднабор объектов RMO (Replication Management Objects). Microsoft рекомендует использовать для работы с SQL Server именно эту библиотеку;
- **SQL-DMO (SQL Database Management Objects)** — объектная модель на основе COM-компонентов. В отличие от SMO, она не требует наличия

.NET Framework и .NET-совместимых программных языков. Эта библиотека использовалась и в предыдущих версиях SQL Server. Она проще, чем SMO, требует меньших системных ресурсов, но функциональность ее меньше (хотя ее вполне хватает для выполнения большинства операций);

- **поставщик WMI (Windows Management Instrumentarium) для SQL Server** — специфическая объектная библиотека, которая ориентирована не на выполнение административных операций, а на получение информации об SQL Server и предупреждений о событиях. Работать с ней сложнее, чем с SMO и SQL-DMO, но во многих ситуациях она может оказаться очень полезной.

Подчеркнем, что эти объектные модели предназначены не для доступа к данным, а для выполнения административных операций (поэтому в названии каждой из них присутствует слово *management* (*управление*)). Для того чтобы работать с данными в базе данных SQL Server, удобнее использовать специально предназначенные для этого объектные модели ADO и ADO.NET.

По опыту автора можно сказать, что время, потраченное на освоение возможностей объектных моделей SMO, SQL-DMO и WMI, для большинства администраторов и разработчиков оккупится очень быстро. Главное преимущество заключается в том, что выполнение многих операций упрощается многократно, а на создание программного кода (в отличие от кода Transact-SQL) требуется намного меньше времени. Кроме того, объекты этих библиотек вполне могут пригодиться и для кода Transact-SQL. Объекты из SQL-DMO и WMI могут использоваться при помощи хранимых процедур автоматизации (`sp_OACreate` и т. п.), а объекты SMO — в подсистеме интеграции CLR (т. е. при обращении к сборкам .NET объектной модели из кода Transact-SQL).

Отметим также, что материалы этой главы предназначены для квалифицированных администраторов и разработчиков. Разд. 9.2, посвященный работе с SMO, рассчитан на то, что читатель знаком с .NET и каким-нибудь .NET-совместимым языком программирования (в примерах будет использоваться самый простой и распространенный — Visual Basic .NET). Разд. 9.3 и 9.4, в которых рассказывается про объектные библиотеки SQL-DMO и WMI, требуют знания любого COM-совместимого языка (VBScript, JavaScript, ActivePerl, Visual Basic, C++, Delphi и т. п.). Для всех примеров этих разделов будет использоваться язык VBScript.

9.2. Объектная модель SMO

9.2.1. Обзор объектной модели SMO

SMO (SQL Management Objects) — это рекомендованная Microsoft объектная модель для программного выполнения административных операций на SQL

Server 2005 (с ее помощью можно также работать с SQL Server 7.0 и 2000). Эта объектная модель основана на технологии .NET. Она требует наличия на компьютере .NET Framework 2.0, а работать с SMO настоятельно рекомендуется только средствами Visual Studio .NET 2005.

Установка SMO производится вместе с SQL Server 2005, а необходимые сборки по умолчанию помещаются в каталог C:\Program Files\Microsoft SQL Server\90\SDK\Assemblies. Для того чтобы были установлены все необходимые компоненты для создания программ, использующих возможности SMO, нужно при установке поставить флажок напротив пункта **SDK** в наборе элементов **Client Components**.

Объектная модель SMO очень похожа на унаследованную объектную модель SQL-DMO, которая применялась в SQL Server 7 и SQL Server 2000 (см. разд. 9.3). Разработчики постарались по возможности использовать те же названия объектов, свойств и методов. Однако в SMO появилось большое количество новых возможностей:

- в SMO используются преимущества .NET с точки зрения повышения производительности, например такие, как частичная загрузка больших объектов в оперативную память. В принципе, SMO намного требовательнее к ресурсам, чем SQL-DMO, что объясняется особенностями работы приложений .NET. Однако некоторые операции в SMO могут выполняться быстрее. Например, если вы проходите циклом по коллекции объектов (баз данных на сервере, таблиц в базе данных и т. п.) в SQL-DMO, то каждый из этих объектов будет создан полностью и помещен в оперативную память. В SMO они будут помещены в оперативную память только частично, и, возможно, такой цикл будет выполнен быстрее;
- улучшена производительность при выполнении команд Transact-SQL на сервере. Теперь среда выполнения .NET/SMO автоматически отслеживает ситуации, когда на сервер передается набор команд Transact-SQL, и посыпает эти команды единым пакетом;
- улучшена событийная модель — как за счет интеграции с возможностями объектной библиотеки WMI, так и за счет собственных средств SMO;
- появилась возможность указания объектов при помощи специальных путей к ним в формате URN (Unique Resource Name — уникальное имя ресурса). Формат URN был разработан на основе формата путей XPath, которые используются для навигации по документам XML. Например, путь в формате URN к базе данных db1 может выглядеть так:

```
/Server/Database[@Name='db1']
```
- появилось большое количество новых объектов, реализующих новые возможности SQL Server 2005. Например, в SMO предусмотрены объекты для

точек подключения по HTTP, схем XML, моментальных снимков баз данных, Database Mail, сертификатов и асимметричных ключей и т. п.

Для использования объектов SMO в приложении вам потребуется добавить в проект ссылки на необходимые библиотеки. Кроме того, чтобы можно было использовать короткие имена объектов, рекомендуется импортировать требуемые пространства имен. Стандартная последовательность действий в Visual Studio .NET 2005 перед началом работы с SMO выглядит так:

1. Создайте нужный проект. В примерах этого раздела будут использоваться проекты Visual Basic .NET 2005 типа Windows Application или Console Application.
2. Добавьте в проект ссылки на следующие библиотеки (сборки .NET):

- Microsoft.SqlServer.ConnectionInfo.dll;
- Microsoft.SqlServer.Smo.dll;
- Microsoft.SqlServer.SqlEnum.dll;
- Microsoft.SqlServer.SmoEnum.dll.

Добавление ссылок в проект можно выполнить при помощи команды **Add Reference** (Добавить ссылку) контекстного меню проекта в окне **Solution Explorer**.

3. Добавьте в самое начало раздела **Declarations** (Объявления) для вашей формы или программного модуля команды:

```
Imports Microsoft.SqlServer.Management.Smo  
Imports Microsoft.SqlServer.Management.Common
```

4. Затем создайте необходимые процедуры и поместите в них программный код для работы с объектами SMO. Основные объекты SMO будут рассмотрены в следующих разделах.

Для всех примеров этого раздела будем считать, что эти операции уже выполнены.

Документацию по работе с SMO можно найти в Books Online в разделе **SQL Server Books Online | SQL Server Programming Reference | Database Engine Administration Programming | SQL Management Objects (SMO)** (SQL Server Books Online | Справка по программированию SQL Server | Программирование операций по администрированию ядра баз данных | SQL Management Objects (SMO)) или в MSDN для Visual Studio .NET 2005. Информация в обоих источниках совершенно одинакова.

К сожалению, справка по объектам SMO отличается очень большой краткостью. И очень часто понять правильную последовательность действий достаточно сложно, а примеры для большинства объектов отсутствуют. Поэтому

рекомендуется, кроме описаний свойств и методов объектов, использовать также:

- *примеры работы с SMO в Books Online.* Найти их можно в разделе **SQL Server Books Online | SQL Server Programming Reference | Database Engine Administration Programming | SQL Management Objects (SMO) | Programming Specific Tasks** (Программирование конкретных задач);
- *примеры (Samples), поставляемые с дистрибутивом SQL Server 2005.* Если вы выбрали их как компонент во время установки, то найти их можно в меню **Пуск | Программы | Microsoft SQL Server 2005 | Documents and Tutorials | Samples** (Документация и учебные руководства | Примеры). Нужно развернуть пакет MSI, который называется **Microsoft SQL Server 2005 Samples (English)** (для этого достаточно просто щелкнуть по нему мышью и выбрать каталог), и открыть в каталоге, в который были помещены примеры, папку `Engine\Programmability\SMO`. В этой папке находится около 20 проектов на Visual Basic и C#, которые иллюстрируют выполнение некоторых операций;
- в качестве дополнительного источника информации можно использовать Интернет.

В этой главе не ставится задача досконально разобрать все возможности огромной объектной библиотеки SMO. Главная цель — познакомить администраторов и разработчиков с этой возможностью и с основными приемами работы с этой библиотекой (которая, по опыту преподавания автора, пока остается неизвестной для большинства администраторов SQL Server 2005).

9.2.2. Общие приемы работы с объектами SMO

Прежде чем приступить к рассмотрению конкретных объектов SMO, опишем свойства и методы, которые являются общими для всех объектов SMO. Эти свойства и методы во многих ситуациях могут оказаться очень полезными. Вначале остановимся на свойствах:

- `Properties` — это свойство возвращает коллекцию `Properties` с объектами `Property`, представляющими свойства данного объекта. При помощи объекта `Property` можно получить информацию об имени свойства, его значении, типе, возможности изменения и т. п. Согласно документации, при помощи свойства `Properties` можно программным путем получить информацию о всех свойствах любого объекта SMO. На практике это не так: многих свойств в этой коллекции почему-то обнаружить не удается. Например, из всех многочисленных свойств объекта `SMO.Server` в эту коллекцию попало только свойство `InstanceName`;

- **State** — при помощи этого свойства можно получить информацию о состоянии объекта SMO. Для него предусмотрено пять значений:
 - *Creating* (0) — в данный момент происходит создание объекта;
 - *Dropped* (1) — в данный момент объект удаляется;
 - *Existing* (2) — нормальное состояние объекта. Он существует, и ничего особенного с ним не происходит;
 - *Pending* (3) — для этого объекта в очереди на выполнение стоит какая-то команда;
 - *ToBeDropped* (4) — для этого объекта в очереди на выполнение стоит команда на его удаление;

- **Urn** — важнейшее свойство объектов SMO. Определяет имя объекта и путь к нему в иерархии объектов SQL Server в специальном формате URN. URN для любого объекта уникален. Например, URN для объекта логина может выглядеть так:

```
Server[@Name='LONDON7\SQL2005']/Login[@Name='sa']
```

В SMO очень удобно находить любой объект по его адресу URN. Для получения ссылки на объект по URN можно использовать метод объекта `SMO.Server`, который называется `GetSMOObject()`:

- **UserData** — это свойство можно использовать для размещения какой-либо пользовательской информации об объекте (например, как флаг обработки при выполнении каких-то операций в несколько приемов).

Для объектов SMO предусмотрены также методы, которые фактически являются общими — они есть у большинства объектов:

- `Alter()` — этот метод вызывается после внесения изменений в свойства данного объекта для того, чтобы эти изменения были сохранены;
- `Grant()`, `Deny()`, `Revoke()` — эти методы позволяют соответственно предоставить кому-либо разрешения, наложить явный запрет и отменить ранее предоставленные разрешения и запреты на объект в иерархии SQL Server;
- `Refresh()` — этот метод заново получает информацию об объекте в базе данных (если объект мог измениться в обход вашей программы);
- `Script()` — этот метод используется для генерации в соответствии с указанными параметрами скрипта Transact-SQL, который позволит воссоздать данный объект.

Отметим еще один момент. В SMO многие объекты сгруппированы в коллекции. Например, в объект `SMO.Server` встроена коллекция `DatabaseCollection`, для каждой базы данных предусмотрены коллекции `TableCollection` и

ViewCollection и т. п. У большинства этих коллекций одинаковые свойства и методы:

- Count — это свойство позволяет получить информацию о количестве элементов в коллекции (например, количество таблиц в базе данных);
- Item — ссылка на конкретный элемент в коллекции. Всегда можно получить такую ссылку по номеру элемента, а в большинстве случаев и по его имени;
- Parent — ссылка на родительский объект, которому принадлежит эта коллекция;
- Contains() — этот метод позволяет проверить, находится ли в коллекции объект с указанным именем (иногда можно произвести такую проверку и по другим свойствам элементов коллекции);
- CopyTo() — этот метод копирует все элементы коллекции в одномерный массив. При этом можно определить номер для первого элемента. Этот метод можно использовать, например, когда нужно получить общий список объектов из нескольких коллекций.

Для всех коллекций можно использовать синтаксическую конструкцию `For Each`. Во многих ситуациях она может быть очень удобна. Например, получить информацию о всех логинах можно так:

```
For Each oLogin In oSrv.Logins  
MsgBox (oLogin.Name)  
Next
```

Отметим также один момент, непривычный для администраторов и разработчиков, привыкших работать с SQL-DMO. В SQL-DMO создание новых объектов (логинов, заданий SQL Server Agent и т. п.) производится при помощи метода `Add()` соответствующей коллекции. В SMO такой метод для коллекций не предусмотрен. Создание объектов теперь производится при помощи конструктора, которому передаются соответствующие параметры (объект сервера, базы данных и т. п.).

9.2.3. Объект `SMO.Server`

Главный объект в объектной модели SMO — это, конечно, объект `SMO.Server`. Он представляет экземпляр сервера SQL Server. С его помощью производится подключение средствами SMO к конкретному серверу SQL Server. После того, как подключение выполнено, можно воспользоваться свойствами и методами этого объекта, а можно через него обратиться к другим объектам на сервере. Например, при помощи свойства `Databases` этого объекта можно получить коллекцию баз данных (объект `DatabaseCollection`) на этом сервере,

при помощи объектов `Database` из этой коллекции можно обратиться к таблицам баз данных и т. п. Но в любом случае работа должна начинаться с подключения к серверу, которое выполняется при помощи объекта `SMO.Server`.

Создание этого объекта выглядит очень просто:

```
Dim oSrv As New Server
```

Эта строка кода позволяет подключиться по умолчанию к экземпляру SQL Server на локальном компьютере при помощи аутентификации Windows. Для подключения при помощи аутентификации Windows к серверу SQL Server на другом компьютере или к именованному экземпляру на локальном компьютере нужно использовать другой вариант конструктора, который принимает имя сервера в качестве строкового параметра:

```
Dim oSrv As New Server("Server1\Instance2")
```

Если вам нужно использовать аутентификацию SQL Server, то придется вначале создать и настроить объект `Microsoft.SqlServer.Management.Common.ServerConnection`, а затем передать его в качестве параметра конструктору объекта `SMO.Server`:

```
Dim oConn As New ServerConnection  
oConn.ServerInstance = "Server1\Instance2"  
oConn.LoginSecure = False      'Выбираем аутентификацию SQL Server  
oConn.Login = "sa"  
oConn.Password = "P@ssw0rd"  
  
Dim oSrv As New Server(oConn)  
MsgBox (oSrv.Name)          'Проверяем
```

После того, как объект `SMO.Server` создан, а значит, вы успешно подключились к интересующему вас серверу SQL Server, можно использовать свойства и методы этого объекта, чтобы непосредственно с их помощью выполнять какие-то действия или получать доступ к другим объектам SQL Server (базам данных, логинам и т. п.). Далее представлен список самых важных свойств объекта `SMO.Server` с комментариями:

- `BackupDevices` — возвращает коллекцию `BackupDeviceCollection`, представляющую логические устройства резервного копирования, созданные на данном сервере. Обычно эта коллекция используется для получения информации о логических устройствах резервного копирования на сервере (объектах `BackupDevice`) и для их программного создания;
- `Configuration` — это свойство возвращает объект `Configuration`, который предоставляет доступ к параметрам настройки сервера. Эти параметры обычно настраиваются или из свойств сервера в Management Studio, или

при помощи утилиты SQL Server Surface Area Configuration, или при помощи хранимой процедуры `sp_configure`. Например, узнать, разрешена ли на сервере работа со сборками .NET из кода Transact-SQL, можно так:

```
MsgBox (oSrv.Configuration.IsSqlClrEnabled.RunValue)
```

А включить этот параметр (т. е. разрешить работу со сборками) можно следующим образом:

```
oSrv.Configuration.IsSqlClrEnabled.ConfigValue = 1  
oSrv.Configuration.Alter()
```

Другие наборы настроек для сервера доступны при помощи его свойства `Settings`. Еще одно свойство `Information` отвечает за параметры работы сервера, которые программным образом изменить нельзя (версия, локализация сервера, количество процессоров в системе и т. п.);

- `Databases` — при помощи этого свойства можно получить доступ к коллекции `DatabaseCollection`, в которой находятся объекты баз данных сервера. В объектах баз данных находятся объекты таблиц, представлений, хранимых процедур и т. п. Естественно, это наиболее часто используемая ветвь объектной модели SMO. Про объекты `DatabaseCollection` и `Databases` будет рассказано в следующем разделе;
- `FullTextService` — при помощи этого свойства можно получить доступ к одноименному объекту, представляющему службу полнотекстового поиска на SQL Server 2005, и, например, программным образом изменить параметры работы службы;
- `InstanceName` — возвращает имя экземпляра SQL Server (например, `Server2`). Информацию о полном имени сервера (вида "`LONDON\Server2`") можно получить при помощи свойства `Name`;
- `JobServer` — это свойство представляет еще один очень важный объект SMO — `JobServer`. С его помощью можно получить доступ как к настройкам SQL Server Agent, так и к его объектам: заданиям, предупреждениям, расписаниям и т. д. (с возможностью их программного создания, изменения, запуска, удаления и т. п.). Например, чтобы получить информацию о всех заданиях SQL Server Agent на сервере, можно использовать код вида:

```
For Each oJob In oSrv.JobServer.Jobs  
    MsgBox (oJob.Name)  
Next
```

- `LinkedServers` — это свойство возвращает коллекцию `LinkedServers`, представляющую объекты `LinkedServer` (подключенные серверы, к которым можно выполнять запросы из кода Transact-SQL). В Management Studio

работа с ними производится из контейнера **Server Objects | Linked Servers** (Объекты серверов | Подключенные серверы). Средствами SMO создать объект подключенного сервера можно, например, так (здесь создается объект для другого сервера SQL Server, который называется `LONDON7`):

```
Dim oLinkedServer As New LinkedServer(oSrv, "LONDON7")
oLinkedServer.ProductName = "SQL Server"
oLinkedServer.Create()
```

- **Logins** — это свойство позволяет получить доступ к коллекции `LoginCollection`, в которой находятся объекты `Login`. Объекты `Login` представляют, конечно, объекты логинов на SQL Server. На практике работать с этой коллекцией приходится достаточно часто. Обычно она используется в ситуации, когда вам нужно синхронизировать набор логинов SQL Server с учетными записями Windows или с набором учетных записей какого-то приложения. Создать новый логин типа SQL Server можно, например, так:

```
Dim oLogin As New Login(oSrv, "NewSMOLogin")
oLogin.LoginType = LoginType.SqlLogin
oLogin.Create("P@ssw0rd")
```

В этом коде "`NewSMOLogin`" — это, конечно, название создаваемого логина, `LoginType.SqlLogin` — тип логина, а "`P@ssw0rd`" — пароль для создаваемой учетной записи (другим путем указать его не получится);

- **Mail** — при помощи этого свойства можно получить доступ к объекту `SQLMail`. Несмотря на название, этот объект представляет не подсистему `SQLMail`, а объекты `Database Mail`. Обычно он используется для того, чтобы программным образом создавать или настраивать профили и учетные записи `Database Mail`;
- **NotificationServices** — это свойство позволяет получить доступ к объекту `NotificationServices`, которое представляет службы `Notification Services`, установленные для данного экземпляра SQL Server. В отличие от других аналогичных ветвей, эта ветвь объектов не позволяет получить нормальный доступ к функциональным возможностям `Notification Services`. Причина проста: для работы с `Notification Services` предназначен свой собственный набор программных объектов, который находится в пространстве имен `Microsoft.SqlServer.Management.Nmo`;
- **ReplicationServer** — это свойство позволяет получить доступ к объекту `ReplicationServer`, представляющему подсистему репликации на SQL Server 2005. Для управления репликацией также предусмотрен свой собственный набор объектов `RMO` (`Replication Management Objects`), но корне-

вым для всего многочисленного набора объектов репликации является именно объект `ReplicationServer`, доступ к которому производится при помощи этого свойства;

- `Roles` — это свойство позволяет получить доступ к объекту `ServerRoleCollection`, представляющему все роли сервера (объекты `ServerRole`). Эти объекты обычно используются для назначения или отмены серверных ролей логинам.

А теперь рассмотрим самые важные методы объекта `SMO.Server`:

- `Alter()` — этот метод вызывается после изменения параметров настройки сервера. Чаще всего этот метод используется не для сервера, а для объектов `Configuration` и `Settings`;
- `AttachDatabase()` и `DetachDatabase()` — эти методы позволяют провести соответственно подключение и отключение базы данных к серверу с указанными вами параметрами;
- `Enum...()` — эти многочисленные методы позволяют вернуть (в виде объекта `DataTable` или какой-нибудь коллекции) информацию о соответствующих объектах на SQL Server. Например, метод `EnumLocks()` возвращает информацию о всех блокировках на сервере, `EnumProcesses()` — о всех процессах и т. п. Объект `DataTable` принадлежит объектной модели ADO.NET, и для него предусмотрен очень удобный набор свойств и методов;
- `GetActiveDBConnectionCount()` — этот метод позволяет узнать, сколько пользователей в настоящее время подключено к указанной базе данных, например:

```
MsgBox(oSrv.GetActiveDBConnectionCount("db1"))
```

Обычно применяется для проверок перед выполнением с базой данных каких-либо операций;

- `GetSMOObject()` — очень важный метод, позволяющий получить ссылку на объект в иерархии объектов SQL Server по уникальному пути к нему в формате URN;
- `KillAllProcesses()` — этот метод принудительно отключает всех пользователей от указанной базы данных. Обычно используется перед выполнением каких-либо операций с базой данных;
- `KillDatabase()` — удаление указанной базы данных (вместе с файлами). Все имеющиеся в этот момент подключения пользователей к этой базе данных будут принудительно закрыты;
- `KillProcess()` — этот метод принудительно закрывает пользовательское подключение к SQL Server. Для процесса нужно указать его ID, которое

можно получить при помощи хранимых процедур (например, `sp_who`) или при помощи метода `EnumProcesses()`;

- `PingServerVersion()` — этот метод принимает имя опрашиваемого сервера и возвращает объект `ServerVersion` с информацией о версии SQL Server. Этот метод очень удобно использовать для выявления серверов, на которых не установлены пакеты обновлений и патчи;
- `ReadErrorLog()` — этот метод возвращает (при помощи объекта `DataTable`) информацию из текущего или указанного вами журнала ошибок SQL Server.

9.2.4. Объект *SMO.Database*

В предыдущем разделе вы познакомились с тем, как при помощи объекта `SMO.Server` подключаться к SQL Server и выполнять некоторые операции на уровне всего сервера. Однако чаще всего вам нужно производить выполнение программных операций с базами данных и их объектами — таблицами, представлениями, индексами, хранимыми процедурами, пользователями и ролями баз данных и т. п. Для того чтобы выполнить какие-то операции с базой данных или получить доступ к ее объектам, необходимо получить ссылку на эту базу данных при помощи объекта `SMO.Database`.

Чтобы получить ссылку на объект базы данных (т. е. создать объект `SMO.Database`), в SMO чаще всего применяют два способа:

1. Использовать свойство `Item` коллекции `DatabaseCollection`, доступ к которой производится при помощи свойства `Databases` объекта `SMO.Server`. Например, чтобы обратиться к базе данных `db1` на сервере `LONDON\Server2`, можно использовать код вида:

```
Dim oSrv As New Server("LONDON\Server2")
Dim oDB As Database
oDB = oSrv.Databases.Item("db1")
```

2. Получить ссылку на объект при помощи идентификатора URN:

```
Dim oDB As Database
oDB =
oSrv.GetSmoObject("Server[@Name='LONDON\Server2']/Database[@Name='db1']")
```

Этот способ несколько сложнее, но более универсален.

Синтаксис с использованием идентификатора URN поначалу может показаться сложным, но на практике этот способ позволяет выиграть с точки зрения производительности и простоты программного кода в тех случаях, когда вам нужно обращаться в объектам "в глубине" иерархии объектов SQL Server, например, к столбцам таблицы.

После того, как вы получили ссылку на объект базы данных, можно использовать свойства и методы этого объекта. Значительная часть свойств объекта SMO.Database предназначена для получения доступа к коллекциям объектов нижних уровней (Tables, Views, Triggers, StoredProcedures и т. п.). Другая часть, к которой относятся свойства `DBOLogin` и множество свойств, начинающихся на `Is...`, предназначена для проверки прав текущего пользователя перед выполнением какой-либо операции. Назначение большинства методов также понятно из их названий. Обратим внимание лишь на два метода:

- `ExecuteNonQuery()` — этот метод позволяет выполнить команды в базе данных в тех ситуациях, когда результаты выполнения запроса (в виде набора табличных данных) вас не интересуют, например:

```
oDB.ExecuteNonQuery("exec SP_CHANGEDBOWNER 'user1'")
```

- `ExecuteWithResult()` — этот метод позволяет выполнить запрос к базе данных и вернуть его результаты в виде стандартного объекта `DataSet` из библиотеки ADO.NET. Обычно этот объект используется для привязки и отображения в специальных элементах управления, таких как `DataGridView`. Самый простой способ, чтобы убедиться, что результаты действительно возвращаются, такой:

```
Dim oDataSet As DataSet  
oDataSet = oDB.ExecuteWithResults("SELECT * FROM table1")  
MsgBox(oDataSet.GetXml)
```

9.3. Объектная модель SQL-DMO

9.3.1. Обзор объектной модели SQL-DMO

Как говорилось в предыдущих разделах, SMO — это новая объектная модель для работы с SQL Server 2005, основанная на технологии .NET. В отличие от нее, объектная модель SQL-DMO построена на традиционных COM-технологиях и не требует .NET. Объекты SQL-DMO удобно применять в скриптах. Кроме того, работа с ними обычно требует намного меньше ресурсов, чем работа с объектами SMO.

Объектная модель SQL-DMO использовалась и в предыдущих версиях SQL Server. В SQL Server 2005 она попала практически без изменений. Поэтому в ней отсутствуют объекты для работы с новыми возможностями SQL Server 2005, такими как сертификаты, точки подключения по HTTP, коллекции XML Schema и т. п. Кроме того, Microsoft предупреждает, что в следующих версиях SQL Server эта объектная библиотека может не поддерживаться. Поэтому для создания больших и функциональных приложений, рассчитан-

ных на продолжительный срок жизни, предпочтительнее использовать SMO. Однако для применения в скриптах (в заданиях SQL Server Agent, пакетах SSIS, просто в операционной системе для целей автоматизации каких-то операций) возможностей SQL-DMO вполне достаточно.

Библиотеки объектной модели SQL-DMO устанавливаются автоматически при установке SQL Server 2005 или средств администрирования. Если вы хотите запускать скрипты или программы с использованием SQL-DMO на компьютере, на котором SQL Server не стоит (например, чтобы с рабочей станции администратора выполнять скрипты сразу для нескольких серверов в вашей сети), то вам потребуется установить эту библиотеку. Для этого достаточно скопировать на нужный компьютер файл Sqldmo.dll (по умолчанию на том компьютере, на котором установлен SQL Server 2005, его можно найти в каталоге C:\Program Files\Microsoft SQL Server\90\Tools\Binn) и зарегистрировать его, выполнив в командной строке команду:

```
regsvr32 sqldmo.dll
```

Конечно, эта команда должна быть выполнена из того каталога, в который вы скопировали этот файл.

Основной источник информации по работе с SQL-DMO — Books Online (раздел **SQL Server Books Online | SQL Server Programming Reference | Database Engine Administration Programming | SQL-DMO** (SQL Server Books Online | Справка по программированию SQL Server | Программирование операций по администрированию ядра баз данных | SQL-DMO)). Кроме того, много примеров выполнения конкретных операций можно найти в Интернете.

В следующих разделах для всех примеров по SQL-DMO будет использоваться язык VBScript (как самый простой и часто используемый для автоматизации администрирования). Для работы с административными скриптами VBScript и многими другими скриптовыми языками можно порекомендовать программу Sapien PrimalScript. В ней предусмотрена подсветка синтаксиса, подсказка по свойствам и методам объектов, примеры синтаксических конструкций, ссылки на сайты с примерами программного кода и т. п. Эту программу можно скачать с сайта www.sapien.com (ею можно пользоваться без регистрации в течение 30 дней).

В этой главе не ставится задача охватить все из более 150 объектов SQL-DMO. Будут рассмотрены только самые важные из них и показаны основные возможности этой объектной модели.

Отметим одну особенность объектной модели SQL-DMO. Многие объекты в ней продублированы. Например, вы можете найти объекты с именами типа SQLServer И SQLServer2, Table И Table2, User И User2 и т. п. Все объекты, кото-

рые оканчиваются на 2, обладают дополнительными свойствами и методами, доступными только при подключении к SQL Server 2005 и SQL Server 2000 (обычные объекты — только свойства и методы, доступные и для 7.0, и для более поздних версий). Далее будут рассматриваться только объекты, оканчивающиеся на 2.

9.3.2. Объект *SQLDMO.Application*

SQLDMO.Application — это объект самого верхнего уровня в иерархии SQL-DMO (в SMO, как вы помните, его уже нет). Он предоставляет доступ к объектам SQL-DMO нижних уровней и к локальной библиотеке DMO.

Обычно этот объект используется для двух целей:

- для получения справки по версиям, наименованию, полному пути к компонентам SQL-DMO;
- для получения списка серверов SQL Server, видимых по сети.

Например, для получения списка всех серверов SQL Server в сети можно использовать скрипт:

```
Dim oApp, oNameList, nCount
Set oApp = CreateObject("SQLDMO.Application")
Set oNameList = oApp.ListAvailableSQLServers()
nCount = oNameList.Count
For i = 0 To nCount
    WScript.Echo (oNameList.Item(nCount))
Next
```

9.3.3. Объект *SQLDMO.SQLServer2*

Как и в объектной модели SMO, главный объект SQL-DMO — объект, представляющий сервер SQL Server. В SQL-DMO он называется *SQLDMO.SQLServer2*. Чаще всего логика скрипта SQL-DMO выглядит так: создаем объект *SQLDMO.SQLServer2*, используем его для подключения к серверу, затем через объект *SQLDMO.SQLServer2* получаем доступ к подчиненным объектам (базам данных, логинам, заданиям и т. п.) и выполняем с ними определенные операции, в конце снова через *SQLDMO.SQLServer2* разрываем соединение. Например, для получения информации о размере свободного пространства во всех базах данных можно использовать скрипт вида:

```
Set oServer = CreateObject("SQLDMO.SQLServer2")
'Используем аутентификацию Windows
oServer.LoginSecure = True
oServer.Connect "Server1\Instance2"
```

```
For Each oDatabase In oServer.Databases
    WScript.Echo oDatabase.Name & " " & oDatabase.SpaceAvailableInMB
Next
oServer.Disconnect
Set oServer = Nothing
```

Свойств и методов у объекта `SQLDMO.SQLServer2` очень много, поэтому далее представлены только самые важные из них:

- `CodePage` — кодировка, установленная для всего SQL Server;
- `LoginSecure` — если для этого свойства используется значение `False` (по умолчанию), то подключение выполняется от имени логина SQL Server, и в методе `Connect()` нужно дополнительно указывать имя логина и его пароль. Если настроить для этого свойства значение `True`, то будет использоваться аутентификация Windows. Обратите внимание, что в отличие от SMO отдельного объекта соединения в SQL-DMO не предусмотрено. Другое важное отличие — то, что по умолчанию используется аутентификация SQL Server, а не Windows;
- `Name` — имя SQL Server. В этом свойстве может быть указан псевдоним;
- `NetName` — реальное сетевое имя SQL Server (обычно применяется для проверок тогда, когда при подключении к серверу использовался псевдоним);
- `RegionalSettings` — региональные установки, настроенные для драйвера ODBC. Эти настройки можно изменять в ходе выполнения скрипта;
- `SaLogin` — это свойство позволяет проверить, обладаете ли вы правами системного администратора на сервере. Обычно используется для проверки перед выполнением какой-либо операции;
- `Status` — текущее состояние SQL Server. Для этого свойства предусмотрено восемь значений:
 - 0 — состояние сервера определить не удалось;
 - 1 — нормальное рабочее состояние;
 - 2 — работа службы приостановлена;
 - 3 — служба SQL Server остановлена;
 - 4 — в данный момент происходит переход из состояния "остановлено" в состояние "запущено";
 - 5 — происходит обратный переход: из "запущено" в "остановлено";
 - 6 — в данный момент сервер запускается после приостановки работы;
 - 7 — обратный переход: работа сервера приостанавливается;

- AddStartParameter() — этот метод позволяет добавить параметр запуска для SQL Server (например, для перевода в однопользовательский или минимальный режим, для запуска с базой данных master на другом диске и т. п.);
- Connect() и Disconnect() — эти методы позволяют соответственно подключиться или отключиться от сервера SQL Server;
- ExecuteImmediate() — этот метод аналогичен методу ExecuteNonQuery() в SMO. Он позволяет запустить на выполнение команду или запрос на SQL Server, когда вас не интересует результат выполнения этой команды или запроса;
- ExecuteWithResults() — этот метод позволяет выполнить команду Transact-SQL на сервере и получить возвращаемый ею табличный набор записей. Прием возвращаемых результатов производится при помощи объекта QueryResults;
- ExecuteWithResultsAndMessages() — этот метод расширяет возможности метода ExecuteWithResults(). Кроме результатов выполнения запроса, он позволяет принять в строковую переменную сообщения SQL Server, которыми сопровождается выполнение запроса;
- IsPackage() — этот метод проверяет редакцию SQL Server. Может возвращать 5 значений:
 - 0 — возникла ошибка, значение получить не удалось;
 - 1 — Desktop Edition;
 - 2 — Standard Edition;
 - 3 — Enterprise или Developer Edition;
 - 4 — Express Edition (бывшая MSDE);
- Start(), Shutdown(), Pause(), Continue() — эти методы соответственно запускают, останавливают, переводят в режим паузы или продолжают работу SQL Server;
- ReadErrorLog() — получает все содержимое журнала SQL Server в виде объекта QueryResult.

Кроме того, при помощи свойств объекта `SQLOLDMO.SQLServer` можно получить доступ к набору объектов нижних уровней:

- BackupDevices — эта коллекция предназначена для работы с логическими устройствами резервного копирования;
- Configuration — это свойство позволяет настроить через скрипт все параметры сервера, доступные средствами хранимой процедуры `sp_configure`;

- Databases — это свойство предоставляет доступ к базам данных и их многочисленным объектам;
- FullTextService — работа из скрипта с полнотекстовыми каталогами;
- IntegratedSecurity — при помощи этого свойства можно получить доступ к одноименному объекту, при помощи которого настраиваются параметры безопасности (в Management Studio они доступны на вкладке **Security** (Безопасность) свойств сервера);
- JobServer — как и в SMO, этот объект предоставляет доступ к элементам автоматизации SQL Server Agent, таким как задания, предупреждения и операторы;
- Languages — информация о языковых модулях, установленных с SQL Server;
- LinkedServers — этот объект предназначен для работы с подключенными серверами (с объектами **LinkedServer**);
- Logins — свойство для доступа к коллекции логинов SQL Server;
- Registry — это свойство позволяет считать или изменять любые параметры реестра, относящиеся к данному экземпляру SQL Server;
- Replication — это свойство возвращает объект **Replication**, который можно использовать для настройки репликации средствами SQL-DMO;
- ServerRoles — этот объект позволяет работать со встроенными серверными ролями.

9.3.4. Объект **SQLDMO.Database2**

Как уже говорилось ранее, чаще всего при работе со скриптами SQL-DMO необходимо вначале подключиться к серверу, а затем перейти к работе с нужной базой данных. В отличие от SMO, путей URN в SQL-DMO нет, поэтому для получения ссылки на объект базы данных вам придется использовать только коллекцию **Databases**:

```
Dim oDB  
Set oDB = oServer.Databases("db1")  
MsgBox oDB.Name
```

Объект **Database2** — один из самых больших по набору свойств и методов в SQL-DMO (как и объекты **SQLServer** и **SQLServer2**). Он позволяет выполнять операции как с самими базами данных, так и с подчиненными объектами — таблицами, представлениями, хранимыми процедурами, пользовательскими типами данных и т. п.

Далее представлены самые важные свойства и методы этого объекта:

- `Size`, `SpaceAvailable`, `DataSpaceUsage` — это, соответственно сколько всего места занимают файлы этой базы данных, сколько места свободно и сколько занято данными в файлах базы;
- `DboLogin` — это свойство позволяет в режиме выполнения проверить, обладает ли вы правами `dbo` для этой базы данных;
- `Status` — информация о состоянии базы данных (доступна только для чтения);
- `Check...` () — эти методы позволяют проверить целостность базы данных или отдельных ее компонентов (аналогично команде `DBCC CHECKDB`);
- `Enum...` () — эти методы возвращают коллекции файлов, файловых групп, блокировок, пользователей и т. п.;
- `Script...` () и `GenerateSQL()` — эти методы позволяют отскриптовать базу данных с подчиненными объектами. Отличие между ними заключается в том, что методы `Script...` () позволяют сразу записать код скрипта в файл, а метод `GenerateSQL()` просто возвращает этот код в виде строкового значения.

Ссылки на коллекции таблиц, представлений, хранимых процедур и других объектов базы данных можно получить при помощи соответствующих свойств (`Tables`, `Views`, `StoredProcedures` и т. п.).

9.4. WMI и SQL Server 2005

9.4.1. Что такое WMI

Формальное определение WMI от Microsoft выглядит так: WMI (Windows Management Instrumentarium) — это реализация Microsoft инициативы WBEM (Web-Based Enterprise Management — управление сетью предприятия, основанной на Web), объявленной консорциумом фирм и направленной на снижение общей стоимости владения сетью. Однако попробуем объяснить, что такое WMI, немного по-другому.

Для работы с сетевыми устройствами уже в течение многих лет используется стандарт SNMP (Simple Network Management Protocol). При помощи этого стандартного протокола SNMP-совместимые средства управления сетью могут получать информацию от устройств самых разных производителей. Такой подход оказался очень удобным, и было принято решение добиться похожей стандартизации и для операционных систем, драйверов оборудования, серверных и обычных программных продуктов. Соответствующая инициатива была названа WBEM, а для принятия решений в области этой инициативы

был сформирован консорциум из ведущих предприятий ИТ-отрасли (в их число вошли Microsoft, IBM, Intel, Hewlett-Packard, Sun и многие другие).

Общая идея WBEM проста — обеспечить всем важнейшим программным продуктам стандартный программный интерфейс, при помощи которого к ним могли бы подключаться WBEM-совместимые средства мониторинга и администрирования. При этом акцент, как и в SNMP, сделан на возможности получения информации о работе продукта и мониторинге событий. Однако при помощи данного программного интерфейса можно также выполнять множество административных операций.

Реализация инициативы WBEM корпорацией Microsoft получила название WMI — Windows Management Instrumentarium. Начиная с конца 1990-х гг. во все важнейшие программные продукты Microsoft включена поддержка интерфейса WMI. При помощи WMI вы можете работать (т. е. получать информацию, отслеживать события, выполнять административные операции):

- с самой операционной системой Windows, включая Active Directory (поддержка WMI встроена в Windows 2000, XP и 2003, на Windows NT и Windows 98 соответствующие программные модули можно установить);
- с Exchange Server 2000 и 2003;
- с Internet Information Server (IIS);
- с Systems Management Server 2003;
- с приложениями Microsoft Office;
- и, конечно, с SQL Server 2000 и 2005.

В вашем распоряжении не так много возможностей для работы с SQL Server 2005 из WMI. Фактически все, что вам доступно, — это настройки сетевой конфигурации средствами WMI Provider for Configuration Management и отслеживание событий на сервере средствами WMI Provider for Server Events. Тем не менее знание WMI необходимо для администраторов. Приемы работы с WMI могут пригодиться в самых разных ситуациях, например, при настройке предупреждений SQL Server Agent типа WMI Event Alert или при работе с соответствующими элементами пакетов SSIS. Поэтому здесь будет подробно рассмотрена работа с этим интерфейсом.

Работать с WMI сложнее, чем с объектными моделями SMO и SQL-DMO, а возможностей для управления SQL Server 2005 в ней намного меньше. Но у этой объектной модели есть и важные преимущества:

- WMI — стандартная объектная модель. Одн раз научившись приемам работы с ней, вы сможете использовать их для работы со многими серверными программными продуктами;
- в WMI реализованы замечательные средства для работы с событиями. При помощи событийных запросов вы можете отслеживать создание, удаление

или изменение любого объекта WMI (включая изменение любого его свойства). Часто оказывается, что другого средства мониторинга и немедленного реагирования на события (например, появился файл на диске, на удаленном компьютере завершился какой-то процесс и т. п.) не существует;

- для тех, кто работает с базами данных, может оказаться удобным то, что в WMI существует свой собственный язык запросов WQL (Windows Query Language), который создан на основе языка SQL. Кроме того, для объектов WMI предусмотрен специальный ODBC-драйвер, при помощи которого вы сможете обращаться к объектам WMI как к таблицам на источнике данных (при этом свойства объектов будут соответствовать столбцам в таблице).

Архитектура WMI состоит из 4 главных компонентов:

- **управляющие приложения** — это, как правило, приложения или службы Windows, которые получают данные от WMI. Например, для SQL Server 2005 роль такого приложения играет SQL Server Configuration Manager (его работа полностью основана на использовании программного интерфейса WMI). WMI используется и для некоторых операций SQL Server Management Studio. По WMI обращаются к SQL Server некоторые другие серверные приложения Microsoft, например, Systems Management Server. В примерах следующих разделов роль управляющих приложений будут играть скрипты;
- **управляемые объекты** — это те объекты, доступ к которым можно получить при помощи WMI. На SQL Server такими объектами могут быть, например, клиентские сетевые библиотеки;
- **поставщики WMI** — это драйверы WMI, которые позволяют получить доступ к какому-либо классу объектов. Для доступа к SQL Server 2005 можно использовать два специализированных поставщика, о которых будет рассказано в следующем разделе. Однако очень часто при работе с SQL Server используются дополнительные поставщики, например, поставщики для работы с объектами операционной системы;
- **программное обеспечение WMI** (*WMI software*, в Windows представлено службой Windows Management Instrumentarium) и **репозиторий CIM** (*Common Information Model*) — оба этих компонента отвечают за предоставление запрашиваемой средствами WMI информации и реакцию на вызов методов, однако у них существует разделение обязанностей: программное обеспечение WMI отвечает за динамическую информацию WMI (данные, поступающие непосредственно от операционной системы, оборудования и т. п.), а репозиторий CIM — за статическую информацию (т. е. настройки WMI на данном компьютере). Физически этот репози-

торий расположен в файле CIM.REP в каталоге C:\WINNT\system32\wbem\Repository.

9.4.2. WMI-поставщики для работы с SQL Server

К сожалению, объектная модель WMI реализована для SQL Server 2005 далеко не так полно, как объектная модель SMO. Может показаться странным, но возможности стандартных поставщиков SQL Server 2005 намного беднее, чем возможности поставщика WMI Administration Provider для SQL Server 2000. Когда вопрос, почему так получилось, был задан руководителю группы разработки средств администрирования SQL Server 2005 Euan Garden, она со сласлась на желание клиентов работать с .NET и предложила использовать объектную модель SMO. Тем не менее и в таком виде объектная модель WMI при работе с SQL Server 2005 может оказаться очень полезной. Для работы с SQL Server 2005 можно использовать два поставщика WMI.

Первый поставщик называется *WMI Provider for Configuration Management* (*Поставщик WMI для управления конфигурацией*). Его основное назначение — управление сетевой конфигурацией SQL Server 2005 (серверными и клиентскими сетевыми библиотеками, настройками сетевых протоколов и т. п.). В нем предусмотрены также объекты для управления службами SQL Server 2005. Фактически функциональность этого поставщика полностью совпадает с функциональностью SQL Server Configuration Manager. Он автоматически устанавливается на компьютер вместе с SQL Server 2005. Справка по нему находится в Books Online (раздел **SQL Server Books Online | SQL Server Programming Reference | Database Engine Administration Programming | WMI and SQL Server | WMI Provider for Configuration Management**).

Все объекты этого поставщика продублированы в объектной модели SMO в пространстве имен Microsoft.SqlServer.Management.Smo.Wmi.

Второй поставщик — *WMI Provider for Server Events* (*Поставщик WMI для событий сервера*). Как понятно из названия, главное назначение этого поставщика — мониторинг событий на сервере. В нем предусмотрено два главных типа событий:

- **события DDL** (*Data Definition Language* — язык определения данных) — это события, которые связаны с созданием, удалением или изменением объектов на уровне сервера или на уровне базы данных;
- **события трассировки** — это те события, информацию о которых можно получить при помощи SQL Server Profiler.

Этот поставщик также автоматически устанавливается с SQL Server 2005. Справка по нему находится в том же разделе **WMI and SQL Server** в Books Online, что и для поставщика WMI Provider for Configuration Management.

С SQL Server 2000 поставлялся поставщик *WMI SQL Server Administration Provider* (он не устанавливался по умолчанию вместе с сервером, но его можно было установить самостоятельно с дистрибутива из каталога X86\OTHER\WMI). Объекты этого поставщика приблизительно соответствовали объектам SQL-DMO (сервер, логин, база данных, пользователь базы данных, таблица, представление и т. п.). Он был очень удобен и часто использовался разработчиками (например, именно при помощи этого поставщика WMI сервер SMS 2003 подключался к серверу SQL Server 2000). Однако применять его для работы с SQL Server 2005 невозможно: экземпляры SQL Server 2005 просто не видят этот поставщик. В свою очередь, поставщики WMI для SQL Server 2005 тоже игнорируют экземпляры SQL Server 2000.

В пространстве имен `root\Microsoft\SqlServer`, в котором находятся оба поставщика WMI для SQL Server 2005, есть еще один поставщик, имеющий отношение к SQL Server 2005. Он называется *ReportServer* и предназначен для получения информации о Reporting Services и мониторинга событий этих служб. В этой главе поставщик *ReportServer* рассматриваться не будет.

9.4.3. Программные средства для работы с WMI

Наиболее полный набор программных средств и документации для работы с WMI можно найти в WMI SDK (Software Development Kit). Его можно скачать с **Download Center** на сайте Microsoft (www.microsoft.com/downloads). Кратко перечислим самые важные программные средства, которые могут вам пригодиться:

- **WMI CIM Studio** — основная графическая утилита, предназначенная для просмотра пространств имен WMI, классов, их свойств и методов. Также ее можно применять для написания и отладки WQL-запросов;
- **WMI Event Registration** и **WMI Event Viewer** — эти средства можно использовать для регистрации событий WMI и просмотра произошедших событий (аналогично тому, как это реализовано, например, в Performance Monitor);
- **WMI Object Browser** — выделенная в отдельное средство часть утилиты WMI CIM Studio. Используется для просмотра классов WMI, их свойств и методов. Работать с ней нужно осторожно — если обратиться к большому набору экземпляров какого-либо класса (например, к информации о службах), то такой запрос будет выполняться очень долго, а прервать выполнение в аварийном режиме очень трудно.

Еще одно средство для работы с WMI — утилита WBEMTest (*Тестер инструментария управления Windows*), которая автоматически устанавливается вместе с Windows Server 2003. Эта утилита чаще всего используется для вы-

полнения запросов к объектам WMI на языке WQL. Она может также использоваться и для других целей: для создания классов и экземпляров классов, выполнения методов и т. п. Интерфейс этой утилиты представлен на рис. 9.1. Отметим, что все возможности WBEMTest в более удобном виде представлены в WMI CIM Studio.

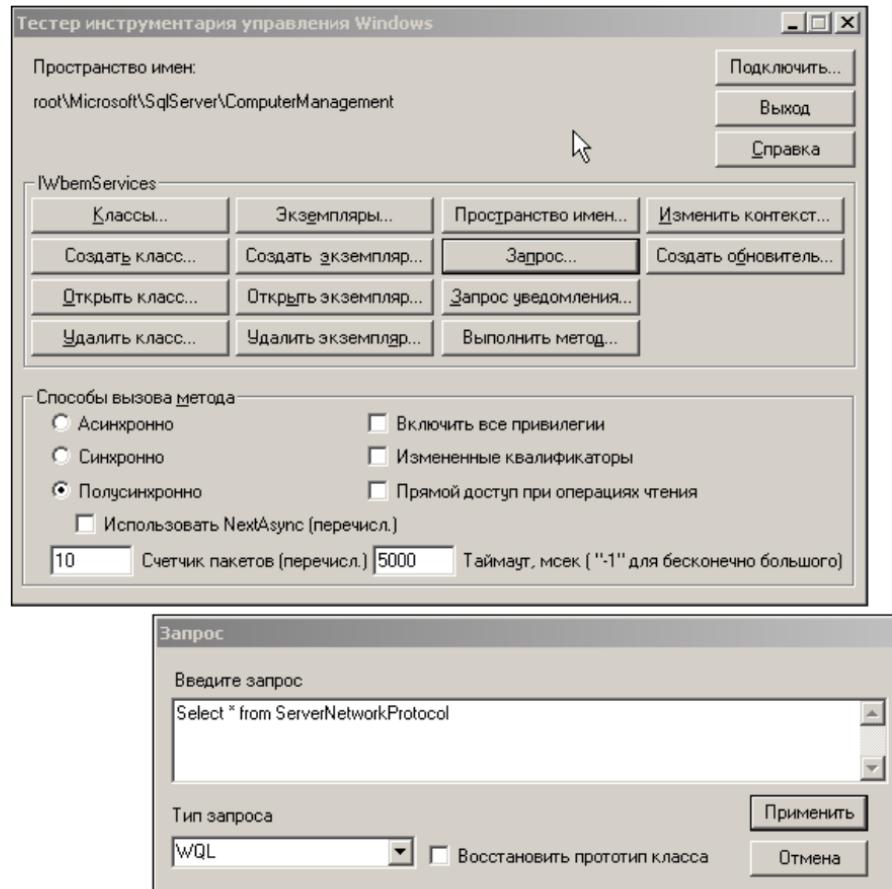


Рис. 9.1. Утилита WBEMTest

Очень удобно установить **ODBC-драйвер для WMI** и получить возможность работать с WMI как с большой базой данных, в которой информация организована в виде привычных таблиц. Драйвер ODBC лежит на компакт-диске дистрибутива Windows 2000 Advanced Server в каталоге D:\VALUEADD\MSFT\MGMT\WBEMODBC. После установки этого драйвера в списке имеющихся на компьютере источников ODBC появляется системный источник данных ODBC (System DSN) под названием **WBEM Source**. Справку по не-

му в виде файла Wbemdr32.chm можно будет найти в файле D:\WINDOWS\Help. При настройке драйвера следует оставить поля для имени пользователя и пароля пустыми. Если вы хотите подключиться при помощи этого источника данных к службе WMI на локальном компьютере, например, из Access, то экран свойств подключения будет выглядеть так, как показано на рис. 9.2.

Затем надо нажать кнопку **Connect** в нижней части экрана (потом она превратится в кнопку **Refresh**) и выбрать соответствующее пространство имен. Например, для поставщика WMI Provider for Configuration Management нужный выбор выглядит так, как представлено на рис. 9.2 (в просмотрщике он называется Computer Management).

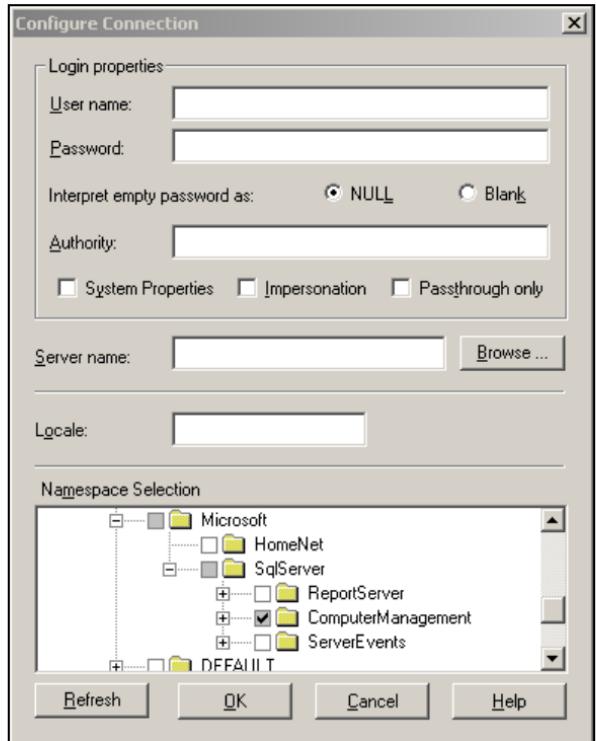


Рис. 9.2. Подключение к объектам WMI при помощи драйвера ODBC

Затем созданное вами подключение можно использовать как обычное подключение к источнику данных по ODBC. Таблицами в нем будут классы данного пространства имен WMI (например, SQLService), свойства, определенные для этого класса, — столбцами, а экземпляры класса (например, логины) — записями в этой таблице. Конечно, все "таблицы", доступные через источник данных ODBC, будут доступны только на чтение.

Работать с WMI можно и при помощи .NET, и средствами обычных СОМ-совместимых программных языков. Далее для всех примеров будет использоваться язык VBScript.

9.4.4. Подключение к службе WMI

Первое, что необходимо сделать в скрипте, — подключиться к службе WMI на локальном или удаленном компьютере. Подключение можно выполнять двумя разными способами: при помощи объекта `Swbemlocator` или при помощи так называемого *моникера* (*moniker* — специальный синтаксис для подключения к объектам СОМ).

Первый способ выглядит так:

1. Вначале получаем объект `SwbemLocator`:

```
Dim oLocator
Set oLocator = CreateObject("wbemScripting.Swbemlocator")
```

У объекта `SwbemLocator` есть всего лишь одно свойство и один метод. Свойство `security_` используется для получения ссылки на одноименный вложенный объект и для настройки безопасности при подключении (если значения параметров безопасности отличаются от значений, предлагаемых по умолчанию), а метод `ConnectServer()` вызывается для подключения к службе WMI на локальном или удаленном компьютере. Если вы подключаетесь к службе WMI на локальном компьютере, то воспользоваться правами другого пользователя (отличного от текущего) вам не удастся — такие ограничения службы DCOM.

2. Далее вызываем метод `ConnectServer()`, который возвращает нам объект `SwbemServices`:

```
Dim oServices
Set oServices = oLocator.ConnectServer("LONDON", _
"root\Microsoft\SqlServer\ComputerManagement")
```

`LONDON` — это, конечно, имя сервера, а `root\Microsoft\SqlServer\ComputerManagement` — пространство имен для поставщика WMI Provider for Configuration Manager. Если вы хотите воспользоваться пространством имен для поставщика WMI Provider for Server Events, последняя строка должна выглядеть так:

```
Set oServices = oLocator.ConnectServer("LONDON", _
"root\Microsoft\SqlServer\ServerEvents")
```

3. Полученный объект `SwbemServices` можно использовать для выполнения запросов WQL, получения ссылок на объекты WMI и т. п.

Второй способ подключения к WMI, который является более стандартным, — использование моникера. *Моникер (moniker)* — это специальная метка, которая применяется для подключения к COM-объектам. При использовании моникера синтаксис получается более коротким и гибким (хотя, возможно, менее понятным):

```
Set oServices =  
GetObject("WinMgmts://LONDON/root/Microsoft/SqlServer/ComputerManagement")
```

А чтобы подключиться с его помощью к объекту, представляющему службы SQL Server, синтаксис может быть таким:

```
Set oServices =  
GetObject("WinMgmts://LONDON/root/Microsoft/SqlServer/ComputerManagement:  
SQLService")
```

Какой же метод выбрать?

Все примеры у Microsoft построены только на использовании моникера, т. к. это проще и гибче. Однако у объекта `Swbemlocator` есть свои преимущества:

- только при использовании этого объекта можно подключаться к компьютеру с правами другого пользователя;
- при использовании этого объекта подсказки по свойствам и методам в средствах работы со скриптами, такими как PrimalScript, остаются, а при использовании моникера сразу пропадают.

Отметим, что подключаться с правами другого пользователя можно только к удаленному компьютеру. Если вы попытаетесь подключиться с новыми правами к службе WMI на локальном компьютере, то возникнет ошибка.

И еще один важный момент. Обратите внимание, что во всех предыдущих примерах использовалось подключение к службе WMI на компьютере в целом, без указания конкретного экземпляра SQL Server. Это характерная особенность поставщиков WMI для работы с SQL Server. Например, при обращении к объектам класса `SQLService`, представляющим службы SQL Server, вы получите объекты для всех экземпляров SQL Server, которые установлены на этом компьютере. Но если вас интересует только конкретный экземпляр сервера, то при обращении к объектам WMI нужно отфильтровывать требуемые экземпляры SQL Server средствами языка запросов WQL.

9.4.5. Язык WQL: подключение к объектам WMI

После того, как вы подключились с службе WMI на нужном компьютере, вам возвращается ссылка на объект `SwbemServices`. В этом несложно убедиться при помощи следующего кода:

```

Dim oLocator
Set oLocator = CreateObject("wbemScripting.Swbemlocator")
Dim oServices
Set oServices = oLocator.ConnectServer("LONDON", _
    "root\Microsoft\SqlServer\ComputerManagement")
MsgBox TypeName(oServices)

```

Объект `SwbemServices` — основная "рабочая лошадка" в объектной модели WMI. Для него предусмотрено множество методов. Однако чаще всего вам нужно выполнить одну задачу — подключиться к конкретному объекту WMI, который, например, представляет службу SQL Server. Это можно сделать разными способами, но самый простой и логичный из них — использовать запрос на языке WQL.

В самом простом варианте применение запроса на языке WQL может выглядеть так:

```

Dim oCollection
Set oCollection = oServices.ExecQuery("SELECT * FROM SQLService")
For Each item In oCollection
    MsgBox item.ServiceName
Next

```

Язык WQL основан на языке SQL и позиционируется Microsoft как ANSI-совместимый, хотя, конечно, отличий от обычного SQL в нем очень много. Например, с его помощью нельзя изменять данные — операторов `INSERT`, `UPDATE` и `DELETE` в нем нет.

В WQL можно выполнять три типа запросов:

- запросы к данным* (они будут рассмотрены далее в этом разделе);
- запросы к событиям* (они будут рассмотрены в разд. 9.4.6);
- запросы к структуре WMI* — позволяют получать информацию о структуре классов WMI. В практической работе они используются редко, поэтому рассматриваться здесь не будут.

Общий вариант синтаксиса запроса WQL выглядит так:

```
SELECT свойства FROM имя_класса WHERE свойство оператор значение
```

Например:

```
SELECT ServiceName, DisplayName FROM SQLService WHERE SQLServiceType='1'
```

В этом примере `ServiceName` и `DisplayName` — это имена свойств, значения которых должны вернуться, `SQLService` — объект WMI, к которому производится запрос, а `WHERE SQLServiceType='1'` — это, конечно, фильтр, который позволяет фильтровать возвращаемые объекты. В данном случае будут воз-

вращены только объекты для службы SQL Server (без служб SQL Server Agent, полнотекстового поиска и т. п.).

Как вы видите, все очень похоже на обычный язык SQL. Классы WMI представляют собой аналоги таблиц, экземпляры этих классов — записи в таблицах, свойства — столбцы в таблице.

В результате выполнения этого запроса вам вернется коллекция `SwbemObjectSet` с объектами `SwbemObject`. В самой коллекции ничего интересного нет, интересны только свойство `Count` (количество элементов в коллекции) и метод `Item()` (который позволяет вернуть нужный элемент). А вот про объекты `SwbemObject`, которые представляют собой конкретные экземпляры объектов WMI, необходимо рассказать подробнее.

Сразу скажем, что `SwbemObject` — это не сам конечный объект WMI (т. е. не служба SQL Server в данном примере), а всего лишь оболочка, в которую "одет" этот объект. Например, у объекта `SwbemObject` для службы SQL Server (представленного конечным объектом `SQLService`) нет свойств `ServiceName`, `DisplayName` и `SQLServiceType`. Но к этим свойствам вполне можно обращаться, как в нашем примере! Этот момент часто запутывает разработчиков: из кода можно вызывать свойства, которые для этого объекта в подсказке редактора кода не видны. Это связано с тем, что обращения ко всем свойствам и методам, которые не определены для объекта `SwbemObject`, автоматически переадресовываются к объекту нижнего уровня. Поэтому, например, такой код будет вполне допустим:

```
Dim oLocator  
Set oLocator = CreateObject("wbemScripting.Swbemlocator")  
  
Dim oServices  
Set oServices = oLocator.ConnectServer("LONDON", _  
    "root\Microsoft\SqlServer\ComputerManagement")  
  
Dim oCollection  
Set oCollection = oServices.ExecQuery("SELECT ServiceName, _  
    DisplayName FROM SQLService WHERE SQLServiceType='1'")  
  
Dim oSWbemObject  
For Each oSWbemObject In oCollection  
    MsgBox oSWbemObject.DisplayName  
Next
```

Отметим еще один важный момент, связанный с объектом `SwbemObject`. При помощи свойства `Properties_` этого объекта можно получить информацию о всех свойствах объекта нижнего уровня, а при помощи свойства `Methods_` —

о всех его методах. Чтобы случайно не перепутать свойства `SwbemObject` и объекта нижнего уровня, названия всех свойств и методов `SwbemObject` заканчиваются символом подчеркивание.

Система WMI изначально построена так, чтобы быть в определенной степени *самодокументируемой*, например, чтобы информацию о всех свойствах и методах объектах WMI можно было получать не при помощи просмотрщика объектов, а просто из кода скрипта. Например, чтобы получить информацию о всех свойствах объекта WMI, представляющего службу SQL Server, можно использовать код вида:

```
For Each item In oSWbemObject.Properties_
    WScript.Echo item.Name & vbTab & item.Value
Next
```

а для получения информации о методах — такую же конструкцию:

```
For Each item In oSWbemObject.Methods_
    WScript.Echo item.Name
Next
```

В первом случае свойство `Properties_` возвращает нам стандартную коллекцию `SwbemPropertySet` с обычными свойствами и методами типа `Add()`, `Remove()`, `Item()`, `Count`, которая состоит из объектов `SwbemProperty`. Самые важные свойства `SwbemProperty` такие:

- `Value` — значение свойства. Это свойство используется по умолчанию;
- `Name` — имя данного свойства;
- `IsArray` — работает это свойство с массивом значений или нет;
- `Qualifiers` — это свойство позволяет спуститься на уровень ниже и получить ссылку на коллекцию `SwbemQualifierSet` (коллекцию допустимых значений для этого свойства), если для него такой набор предусмотрен.

Для методов все устроено точно так же: свойство `Methods_` возвращает коллекцию `SwbemMethodSet`, состоящую из объектов `SwbemMethod`.

9.4.6. Работа с событиями в WMI

Возможно, у вас уже возник вопрос — а зачем нужны такие сложности? Работать с объектами SMO или SQL-DMO намного проще, и функциональных возможностей у них больше. Ответ прост: в WMI реализованы очень мощные средства для работы с событиями.

В WMI предусмотрено понятие *получателя события*. Получатель события — это программа, которой будет отправлена информация от службы WMI о том,

что событие наступило. Получатель события может быть постоянным или временным. Информация о постоянном получателе записывается в репозиторий WMI, и когда наступает событие, для которого зарегистрирован этот получатель, служба WMI проверяет, запущен получатель или нет. Если он работает, то ему просто передается информация о событии, а если нет — служба WMI автоматически запускает получателя (т. е. соответствующую программу).

Временным получателем может быть любая программа, которая подключилась к службе WMI и выполнила специальный событийный запрос при помощи метода `ExecNotificationQuery()` объекта `SwbemServices`. В рассматриваемом примере такими программами будут скрипты. Пример соответствующего скрипта, который будет реагировать на любые изменения для служб SQL Server, может выглядеть так:

```
Dim oLocator  
Set oLocator = CreateObject("wbemScripting.Swbemlocator")  
  
Dim oServices  
Set oServices = oLocator.ConnectServer("LONDON7", _  
    "root\Microsoft\SqlServer\ComputerManagement")  
  
Dim oEventSource  
Set oEventSource = oServices.ExecNotificationQuery _  
    ("SELECT * FROM __InstanceModificationEvent " & _  
    "WITHIN 10 WHERE TargetInstance ISA 'SqlService'")  
  
i = 0  
Do While i = 0  
Set oWbemObject = oEventSource.NextEvent  
MsgBox oWbemObject.TargetInstance.ServiceName & vbTab & _  
    oWbemObject.TargetInstance.State  
Loop
```

Если вы попробуете, например, остановить или запустить службу SQL Server или SQL Server Agent, изменить параметры автозапуска и т. п., то этот скрипт автоматически выведет соответствующее сообщение.

Событийный запрос будет выглядеть так:

```
SELECT * FROM __InstanceModificationEvent WITHIN 10 WHERE TargetInstance  
ISA 'SqlService'
```

В этом запросе `__InstanceModificationEvent` (обратите внимание на два подчеркивания в начале) — одно из девяти событий, на которые можно настроить реакцию:

- `_ClassCreationEvent`, `_ClassDeletionEvent`, `_ClassModificationEvent` — это, соответственно, создание, удаление и изменение класса;
- `_InstanceCreationEvent`, `_InstanceDeletionEvent`, `_InstanceModificationEvent` — создание, удаление и изменение объекта;
- `_NamespaceCreationEvent`, `_NamespaceDeletionEvent`, `_NamespaceModificationEvent` — то же самое для пространства имен (эти события используются редко).

Чаще всего применяются события, связанные с экземпляром класса (*instance*), т. е. с конкретным объектом WMI. Событие `_InstanceModificationEvent` сработает, если изменится любое свойство объекта.

Оператор `WITHIN 10` говорит о том, что опрос службы WMI будет производиться каждые 10 секунд (чтобы не расходовать лишние системные ресурсы). К этому оператору можно добавлять выражение `GROUP BY` (сгруппированные по какому-то признаку события за период будут рассматриваться как одно событие) и `HAVING` (порог на количество событий). Если количество событий меньше этого порога, то событие считается ненаступившим. Полный синтаксис запроса с использованием всех этих операторов может выглядеть, например, так:

```
SELECT * FROM _InstanceModificationEvent
  WITHIN 10 WHERE TargetInstance ISA 'SQLService'
  GROUP WITHIN 30 BY TargetInstance.ServiceName
  HAVING NumberOfEvents > 5
```

Последний оператор событийного запроса `WHERE TargetInstance ISA 'SQLService'` позволяет отфильтровать источник событий. Настоятельно рекомендуется определять его как можно точнее. В данном случае оператор `ISA` используется, чтобы указать, что интересуют все экземпляры класса `SQLService`.

Если нужно включить в запрос дополнительные условия, применяется оператор `AND`:

```
SELECT * FROM _InstanceModificationEvent
  WHERE TargetInstance ISA 'SQLService'
    AND TargetInstance.ServiceName = 'SQLAgent$SQL2005'
```

В результате выполнения метода `ExecQueryNotification()` вам вернется объект `SwbemEventSource`. Обычно для него вызывается метод `NextEvent()`, который ожидает появления события и при срабатывании возвращает стандартный объект `SwbemObject`, представляющий пойманное событие. А затем используется стандартное свойство `TargetInstance` для объекта `SwbemObject`, чтобы обратиться к свойствам и методам конечного объекта (в данном при-

мере — SQLService). Чтобы опрос происходил непрерывно, нужно поместить этот код в бесконечный цикл:

```
i = 0
Do While i = 0
    Set oWbemObject = oEventSource.NextEvent
    MsgBox oWbemObject.TargetInstance.ServiceName & vbTab & _
        oWbemObject.TargetInstance.State
Loop
```

Теперь до тех пор, пока вы не остановите работу скрипта, в окна сообщений будет выводиться информация о любых изменениях свойств служб SQL Server, например, об изменении свойства State (т. е. об изменении состояния службы — остановлена, запущена и т. п.).

Отметим, что этот подход применяется ко всем стандартным поставщикам WMI, в том числе к WMI Provider for Configuration Management. Работа со специальным событийным поставщиком WMI Provider for Server Events выглядит несколько иначе. Она будет рассмотрена в разд. 9.4.8.

9.4.7. Объекты

WMI Provider for Configuration Management

До этого рассматривались общие приемы работы с WMI. Здесь речь пойдет непосредственно о тех объектах, которые можно использовать для работы с SQL Server 2005. В этом разделе будут рассмотрены объекты WMI Provider for Configuration Management.

Еще раз скажем, что функциональность этого поставщика WMI практически полностью повторяет возможности SQL Server Configuration Manager. При помощи объектов этого поставщика вы можете работать с объектами, представляющими службы SQL Server, серверные и клиентские сетевые библиотеки и настройки для протоколов этих библиотек. Приведем список всех объектов данного поставщика:

- SQLService — этот класс представляет службы SQL Server (включая SQL Server Agent, службы полнотекстового поиска, Analysis Services, Reporting Services и т. п.). Он может быть использован:
 - для получения информации о состоянии служб SQL Server (при помощи свойства State);
 - для изменения режима работы служб, например, для перевода в режим автоматического запуска при старте операционной системы (при помощи метода SetStartMode());

- для изменения информации об учетной записи, от имени которой запускаются службы (при помощи методов `SetServiceAccount()` и `SetServicePassword()`);
 - просто для запуска, остановки, перевода в режим паузы, продолжения работы службы (при помощи методов `StartService()`, `StopService()`, `PauseService()`, `ResumeService()`);
- `SQLServiceAdvancedProperty` — класс предназначен для получения информации или изменения параметров дополнительных свойств служб SQL Server. Эти дополнительные свойства средствами графического интерфейса можно просмотреть на вкладке **Advanced** свойств службы в SQL Server Configuration Management. При помощи этих свойств можно настроить, например, разрешение или запрет отправки информации об ошибках в Microsoft, параметры запуска для каждой службы и т. п.;
- `SecurityCertificate` — этот класс предназначен для работы с сертификатом, используемым для шифрования информации средствами SSL при обмене информацией с SQL Server. Позволяет получить информацию о сертификате или настроить его (при помощи метода `SetCurrentCertificate()`);
- `ClientSettings` — этот класс представляет экземпляр клиента SQL Server (т. е. совокупность настроек клиентских сетевых библиотек). Единственное его назначение — возможность вернуть все настройки клиента к исходному состоянию, которое по умолчанию настраивается при установке. Для этой цели используется метод `SetDefaults()`;
- `ClientNetLibInfo` — этот класс позволяет получить информацию о модулях DLL для клиентских сетевых библиотек. В основном используется для проверки их версий;
- `ClientNetworkProtocol` — основной класс для настройки клиентских сетевых библиотек. Позволяет получить информацию о сетевых протоколах, включить или отключить сетевые протоколы и настроить свойства клиентских сетевых библиотек (при помощи класса `ClientNetworkProtocolProperty`), например, IP-адрес для обращения на SQL Server;
- `SQLServerAlias` — класс представляет информацию о псевдонимах SQL Server на клиенте. К сожалению, может использоваться только для получения информации о существующих псевдонимах. Создавать новые псевдонимы или изменять существующие с его помощью нельзя;
- `ServerSettings` — этот класс предназначен для работы с общими настройками серверных сетевых библиотек. Он позволяет вернуть им значения по

умолчанию или настроить используемый сертификат для защиты передаваемой информации по SSL;

- `ServerNetworkProtocol` — этот класс предназначен для настройки сетевых протоколов на SQL Server. Он позволяет получить информацию о настроенных сетевых протоколах, а также включить или отключить сетевые протоколы;
- `ServerNetworkProtocolIPAddress` — этот класс позволяет получить информацию или настроить параметры сетевых протоколов на уровне конкретного IP-адреса. Для внесения изменений в эти параметры может использоваться класс `ServerNetworkProtocolProperty`.

9.4.8. Работа с WMI Provider for Server Events

WMI Provider for Server Events — это специальный поставщик WMI, основная задача которого заключается в предоставлении доступа через стандартный программный интерфейс WMI к уведомлениям о событиях (*event notifications*). Про уведомления о событиях будет рассказываться в разд. 11.2.5. Здесь только отметим, что при помощи уведомлений о событиях можно производить мониторинг событий двух видов: событий DDL и событий трассировки. События DDL связаны с созданием, изменением и удалением объектов базы данных. События трассировки — это те события, которые можно отследить средствами SQL Server Profiler (см. разд. 11.2.3). Событий трассировки очень много. В качестве примеров таких событий можно привести подключение пользователя к серверу, изменение размера базы данных или журнала транзакций, выполнение запроса или запуск на выполнение хранимой процедуры, возникновение взаимоблокировки, пользовательской или системной ошибки и т. п.

WMI Provider for Server Events — это всего лишь внешняя программная оболочка, поэтому при выполнении запроса к объектам этого поставщика создается уведомление о событиях в соответствующей базе данных. При возникновении события сообщение о нем в формате XML помещается в очередь Service Broker. Поэтому для работы с этим поставщиком WMI необходимо обязательно включить Service Broker. Если вы собираетесь отслеживать события для конкретной базы данных, то нужно включить Service Broker для нее, а если вы планируете отслеживать события на уровне всего сервера, то вам потребуется включить Service Broker для базы данных `msdb`. Просмотреть информацию о том, для каких баз данных включен Service Broker, можно при помощи запроса:

```
USE master;
GO
SELECT name, is_broker_enabled FROM sys.databases;
```

Если Service Broker отключен для нужной базы данных, то включить его можно командой ALTER DATABASE, например:

```
ALTER DATABASE AdventureWorks SET ENABLE_BROKER;
```

Работу с WMI Provider for Server Events можно проиллюстрировать на простом примере. Предположим, что вы хотите производить мониторинг всех изменений в структуре любой таблицы базы данных DB1 на сервере LONDON2\SQL2005 и сразу же получать информацию об имени изменившейся таблицы, имени пользователя, который внес изменения, и команде Transact-SQL, которая произвела это изменение. Создать приложение, которое будет реагировать на такое изменение, можно следующим образом.

Первое, что нужно сделать, — создать новый проект .NET. Достаточно выбрать самый простой вариант и воспользоваться шаблоном для языка Visual Basic .NET, который называется Console Application (Консольное приложение).

Далее нужно добавить в проект ссылку на сборку с необходимыми классами для работы с WMI. Для этого в меню **Project** (Проект) в Visual Studio .NET выберите команду **Add Reference**, на вкладке .NET выберите пространство имен System.Management и нажмите кнопку **OK**.

После того как ссылка будет добавлена, можно приступить к созданию программного кода. Весь программный код модуля Module1.vb может выглядеть так:

```
Imports System
Imports System.Management

Module Module1
Sub Main()
    'Создаем объект диапазона и подключаемся к пространству имен
    'WMI Provider for Server Events на локальном сервере
    'для экземпляра SQL2005
    Dim oScope As New
ManagementScope("\\.\root\Microsoft\SqlServer\ServerEvents\SQL2005")
    oScope.Connect()
    'Определяем объект событийного запроса WQL
    Dim oQuery As New WqlEventQuery("SELECT * FROM ALTER_TABLE WHERE
                                         DatabaseName = 'DB1'")
    'Определяем объект наблюдателя за событием и
    'передаем ему запрос
    Dim oWatcher As New ManagementEventWatcher(oQuery)
    'Определяем диапазон наблюдения для объекта наблюдателя
    oWatcher.Scope = oScope
```

```

'Сигнализируем о начале наблюдения
Console.WriteLine("Начинаем мониторинг")
Dim oBaseObject As ManagementBaseObject
'Начинаем перехват событий в бесконечном цикле
Dim i As Integer
i = 0
Do While i = 0
oBaseObject = oWatcher.WaitForNextEvent()
'Выводим нужные свойства объекта, для которого произошло событие
Console.WriteLine(oBaseObject.Properties("ObjectName").Value & _
vbCrLf & oBaseObject.Properties("TSQLCommand").Value)
Loop
End Sub
End Module

```

Задание для самостоятельной работы 9.1. Применение объектной модели SMO

Задание:

Напишите консольную программу на языке Visual Basic .NET с применением объектной модели SMO. Эта программа должна:

- произвести перестроение индексов для всех таблиц базы данных AdventureWorks на сервере *имя_вашего_сервера\SQL2005*;
- вывести на консоль имя каждой таблицы, для которой производится перестроение.

Решение:

1. Откройте Visual Studio .NET 2005 и в меню **File** (Файл) выберите **New | Project**. В окне **New Project** (Новый проект) в списке **Project Types** (Типы проектов) выберите **Visual Basic | Windows** и в списке **Templates** (Шаблоны) справа выберите шаблон **Console Application**. В поле **Name** введите имя для создаваемого проекта (например, *AdventureWorksIndexRebuild*) и нажмите кнопку **OK**. Будет создан новый проект, программный модуль для которого *Module1.vb* откроется в окне редактора кода.
2. В меню **Project** выберите команду **Add Reference** и добавьте в проект ссылки на следующие сборки .NET:

```

Microsoft.SqlServer.ConnectionInfo
Microsoft.SqlServer.SMO
Microsoft.SqlServer.SMOEnum
Microsoft.SqlServer.SQLEnum

```

Добавьте в раздел Declarations программного модуля Module1 следующие команды:

```
Imports Microsoft.SqlServer.Management.Smo
Imports Microsoft.SqlServer.Management.Common
```

3. Добавьте в процедуру Main() программного модуля необходимый программный код. Он для всего модуля может быть таким:

```
Module Module1
    Sub Main()
        'Подключаемся к серверу средствами аутентификации Windows
        Dim oSrv As New Server("LONDON2\SQL2005")
        'Получаем ссылку на объект базы данных AdventureWorks
        Dim oDb As Database
        oDb = oSrv.Databases("AdventureWorks")
        'Проходим циклом по всем таблицам в коллекции Tables
        Dim oTable As Table
        For Each oTable In oDb.Tables
            '(0) – это значение FillFactor, обязательный параметр
            oTable.RebuildIndexes(0)
            Console.WriteLine("Перестроение индексов для таблицы " & _
                oTable.Schema & "." & oTable.Name)
        Next
    End Sub
End Module
```

4. После проверки работоспособности созданной программы можно, например, создать исполняемый файл при помощи меню **Build | Build AdventureWorksIndexRebuild**.

Задание для самостоятельной работы 9.2. Применение объектной модели SQL-DMO

Задание:

Напишите скрипт на языке VBScript с использованием объектной модели SQL-DMO, который:

- проверял бы состояние сервера *имя_вашего_сервера\SQL2005*;
- если сервер остановлен, то автоматически бы его запускал;
- если сервер приостановлен (состояние *Pause*), то переводил бы его в режим *Running*.

Проверьте работу вашего скрипта, запустив его в разных состояниях SQL Server.

Решение:

Соответствующий код скрипта может быть таким:

```
Dim oServer
Set oServer = CreateObject("SQLDMO.SQLServer")
oServer.Name = "LONDON2\SQL2005"
Select Case oServer.Status
    Case 0      'Состояние получить не удалось
        WScript.Echo "Неизвестная ошибка"
    Case 1      'Запущен
        WScript.Echo "Сервер уже работает"
    Case 2      'Приостановлен
        oServer.Continue
    Case 3      'Остановлен
        'FALSE означает: не подключаться к серверу после запуска
        oServer.Start("FALSE")
    Case Else
        WScript.Echo "Сервер в переходном состоянии"
        WScript.Echo "Запустите скрипт повторно через несколько секунд"
End Select
```

Задание для самостоятельной работы 9.3. Работа с WMI Provider for Configuration Management

Задание:

Напишите скрипт VBScript с использованием объектной модели WMI Provider for Configuration Management, который бы с интервалом в 5 секунд опрашивал все установленные на вашем компьютере службы SQL Server (относящиеся к любому экземпляру SQL Server). В случае если служба остановлена или переведена в режим паузы, этот скрипт должен автоматически ее запускать.

Запустите скрипт на выполнение и убедитесь в его работоспособности, останавливая и переводя в режим паузы различные службы SQL Server (в том числе SQL Server Agent) для разных экземпляров SQL Server.

Решение:

Соответствующий скрипт может быть таким:

```
Dim oLocator
Set oLocator = CreateObject("wbemScripting.Swbemlocator")

Set oServices = oLocator.ConnectServer("LONDON2", _
    "root\Microsoft\SqlServer\ComputerManagement")

Dim oEventSource
Set oEventSource = oServices.ExecNotificationQuery _
    ("SELECT * FROM __InstanceModificationEvent " & _
    "WITHIN 5 WHERE TargetInstance ISA 'SqlService'")

i = 0
Do While i = 0
Set oWbemObject = oEventSource.NextEvent
Select Case oWbemObject.TargetInstance.State
    Case 1
        oWbemObject.TargetInstance.StartService()
        WScript.Echo "Служба " & _
            oWbemObject.TargetInstance.ServiceName & _
            " запущена после остановки"
    Case 7
        oWbemObject.TargetInstance.ResumeService()
        WScript.Echo "Служба " & _
            oWbemObject.TargetInstance.ServiceName & _
            " продолжила работу после паузы"
End Select
Loop
```



ГЛАВА 10

Применение SQL Server Integration Services

10.1. Зачем нужны SQL Server Integration Services

SQL Server Integration Services (сокращенно SSIS) формально определяется как набор графических, консольных утилит и программных объектов, которые предназначены для извлечения, преобразования и консолидации данных из разнородных источников в разные "пункты назначения". Можно считать, что SSIS — это что-то вроде мощного насоса, предназначенного для перекачки данных из одного места в другое. Кроме того, службы SSIS в процессе "перекачки" могут преобразовывать и проверять данные.

В предыдущих версиях SQL Server службы-предшественники SSIS, предназначенные для перекачки данных, назывались Data Transformation Services (DTS). Однако изменения в SQL Server 2005 в отношении SSIS/DTS можно назвать скорее революционными, чем эволюционными. Изменилось практически все: средства создания и администрирования пакетов, формат пакетов, среда выполнения, объектные модели, консольные утилиты и т. п. По наблюдениям автора, даже специалисты, которые имеют значительный опыт работы с DTS, осваиваются с SSIS не сразу.

Администраторы и разработчики могут использовать службы SSIS во всех ситуациях, когда нужно производить загрузку, выгрузку, проверку или преобразование данных. При этом совсем не обязательно, чтобы эти данные находились на SQL Server. Вы можете использовать средства SSIS, например, для загрузки файлов DBF на сервер Oracle или для выгрузки информации из Access в файлы XML. Вот несколько обычных ситуаций, когда службы SSIS могут сэкономить вам много времени:

- вам нужно организовать сбор информации из филиалов или подразделений предприятия (например, отчетов о проведенных операциях) и разме-

щение этой информации в базе данных (например, SQL Server или Oracle). Несколько лет назад для этого обычно использовались макетные файлы в формате CSV (Comma-separated Values — значения, разделенные запятыми). В последнее время для передачи данных все чаще используется формат XML. И в том и в другом случае удобнее всего для проверки передаваемых данных и загрузки их на источник использовать средства SSIS;

- в последнее время на многих предприятиях все чаще создают Data Warehouses — хранилища данных. Они представляют собой обычные реляционные базы данных, как правило, большого размера и специальным образом спланированные. В хранилище данных поступает информация из самых разных рабочих источников данных (например, из баз данных SQL Server, Oracle, Access, файлов DBF, Excel и т. п.), с которыми непосредственно работают пользователи. Хранилища данных очень удобно использовать для создания отчетов и анализа сводной информации. Кроме того, за счет перемещения старой информации в хранилища разгружаются рабочие базы данных, с которыми ведется текущая работа (базы данных OLTP), что позволяет сильно повысить их производительность. Поскольку перемещение информации в хранилища производится на регулярной основе, то удобнее всего использовать для этой цели пакеты SSIS;
- многие предприятия не ограничиваются созданием хранилищ данных и делают еще один шаг — создают базы данных OLAP. В этих базах данных вместо обычных двумерных таблиц со столбцами и строками применяются многомерные кубы. Эти кубы очень удобно использовать для проведения анализа, например, в разрезе регионов, типов продуктов, временном разрезе и т. п. Процесс загрузки информации из хранилища данных (или прямо из баз данных OLTP) в кубы OLAP называется *процессингом*. Его тоже проще всего производить при помощи пакетов SSIS.

Существует множество других ситуаций, когда службы SSIS могут оказаться очень полезны. Например, существуют приложения, в которых средствами SSIS в базу данных помещается информация из логов телефонной станции, формируются отчеты и т. п. По опыту автора, время, потраченное на освоение SSIS, окупается очень быстро.

10.2. Средства для работы с SSIS

Главное средство для работы с SSIS — это SSIS Designer, который является составной частью Business Intelligence Development Studio. Чтобы открыть окно SSIS Designer, достаточно запустить Business Intelligence Development Studio из меню **Пуск | Программы | Microsoft SQL Server 2005 | SQL Server Business Intelligence Development Studio** и создать новый проект на основе

шаблона Integration Services Project. Окно SSIS Designer откроется автоматически (рис. 10.1).

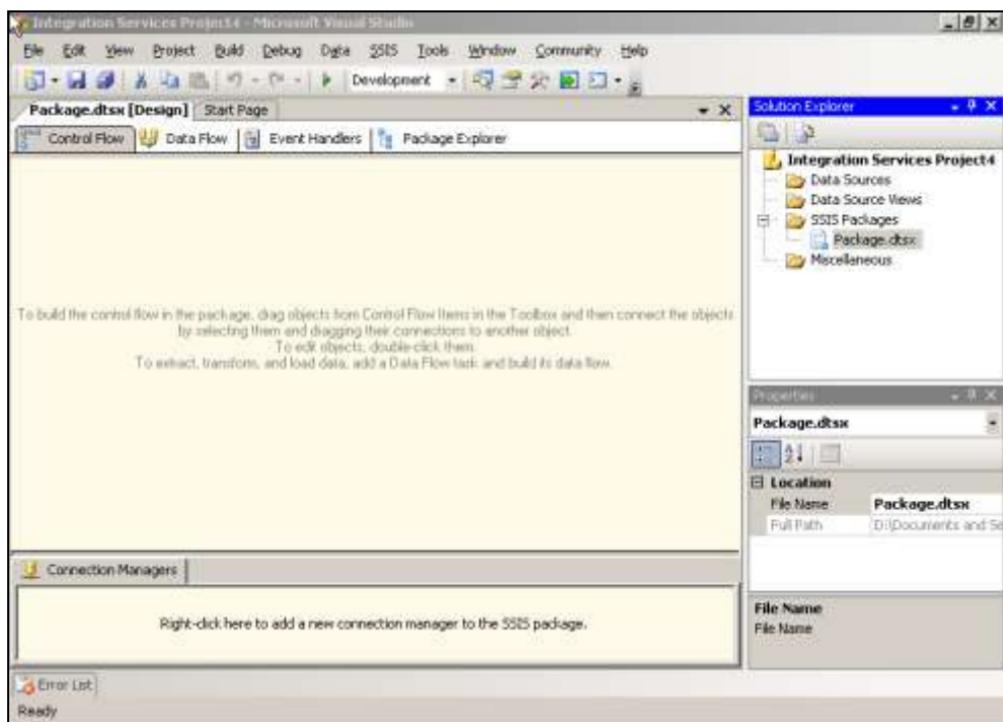


Рис. 10.1. Интерфейс SSIS Designer

При помощи интерфейса SSIS Designer доступны все возможности работы с пакетами — наборами элементов, которые выполняют работу по передаче и преобразованию данных. Вы можете добавлять и настраивать компоненты пакета, сохранять пакет, запускать его на выполнение, производить отладку и т. п.

Отметим принципиальное отличие SSIS Designer от DTS Designer в SQL Server 7.0/2000, который входил в состав Enterprise Manager. В DTS Designer для создания пакета DTS необходимо было обязательно подключиться к SQL Server (даже если в пакете вы к нему не обращались). В SSIS Designer пакет представляет собой фактически специальный проект Visual Studio, и для работы с ним SQL Server не нужен. Это наверняка порадует многих разработчиков.

Если вам нужен самый простой пакет, который будет только перекачивать данные с одного источника на другой, и вы хотите затратить на решение этой

задачи минимум времени, можно воспользоваться мастером SQL Server Import and Export Wizard. Его можно запустить разными способами:

- просто из командной строки операционной системы при помощи команды `DTSWizard`;
- из SSIS Designer при помощи меню **Project | SSIS Import and Export Wizard**;
- из SQL Server Management Studio, если в дереве **Object Explorer** щелкнуть правой кнопкой мыши по объекту базы данных и в контекстном меню выбрать **Tasks | Import Data** (Задачи | Импортировать данные) или **Tasks | Export Data** (Задачи | Экспортировать данные).

Работа с мастером будет рассмотрена в следующем разделе.

Если у вас уже были созданы пакеты DTS в SQL Server 2000, то при переходе на SQL Server 2005 вам может потребоваться перевод их в формат пакетов SSIS. Проще всего это сделать при помощи мастера Package Migration Wizard. Его также можно запустить по-разному:

- из командной строки при помощи команды `DTSMigrationWizard`;
- из SSIS Designer при помощи меню **Project | Migrate DTS 2000 Package** (Проект | Перенести проект DTS2000);
- из SQL Server Management Studio. Для этого нужно раскрыть контейнер **Management | Legacy | Data Transformation Services** (Управление | Унаследованное | Data Transformation Services) и в контекстном меню для этого контейнера выбрать **Migration Wizard**.

Надо сказать, что в принципе можно и не обновлять формат пакетов, созданных на SQL Server 7.0/2000. Эти пакеты вполне можно запускать на выполнение и на SQL Server 2005. Однако редактировать их можно будет только средствами Enterprise Manager из SQL Server 2000.

Часто бывает удобно запускать созданные пакеты из командной строки. Например, это может потребоваться для запуска пакета по расписанию в ночное время (средствами планировщика операционной системы или заданий SQL Server Agent). Запуск пакета из командной строки операционной системы производится при помощи утилиты `dtsexec`. Сгенерировать для нее командную строку (или точно так же запустить пакет) можно при помощи графической утилиты `dtsexecui`.

Из командной строки можно также производить некоторые административные операции с пакетами, например, копирование, перемещение, удаление, шифрование и т. п. Для этого предназначена утилита `dtutil`.

Но основным средством для работы с пакетами, конечно, является SSIS Designer. Но вначале рассмотрим создание простых пакетов средствами SQL Server Import and Export Wizard.

10.3. Применение мастера импорта и экспорта данных

Самый быстрый и простой способ получить доступ к возможностям, предоставляемым SSIS, — воспользоваться мастером SQL Server Import and Export Wizard.

Проиллюстрируем работу мастера на простом примере. Предположим, что нужно выгрузить информацию с SQL Server в файл Excel (такая операция очень часто выполняется на многих предприятиях). В этом примере данные для выгрузки (список сотрудников) содержатся в представлении HumanResources.vEmployee в учебной базе данных AdventureWorks, поставляемой с SQL Server 2005. Сформированный файл должен находиться в корневом каталоге диска D: и называться employees.xls. При этом такую операцию нужно будет повторять и в будущем, поэтому необходимо сохранить настройки импорта и экспорта в виде пакета SSIS.

Способы запуска мастера импорта и экспорта данных были перечислены в предыдущем разделе. Если вы запустите мастер из SQL Server Management Studio, в нем будут автоматически настроены параметры подключения. В этом примере будем считать, что мастер запущен из командной строки при помощи команды `DTSWizard`, и все параметры подключений требуется настроить вручную.

На первом экране мастера **Choose a Data Source** (Выберите источник данных) нужно выбрать то место, откуда будут извлекаться данные (в этом случае — сервер SQL Server 2005, расположенный на вашем компьютере). Отметим несколько моментов, связанных с выбором источника данных:

- в мастере импорта и экспорта данных вам доступны не все источники данных, к которым можно подключиться из SSIS Designer. Например, в вашем распоряжении нет:
 - источника данных Raw File Source. Этот источник представляет собой двоичный файл, который был ранее сформирован средствами SSIS при помощи назначения Raw File Destination. Данные в этот двоичный файл записываются в "родном" формате SQL Server 2005 и при загрузке не требуют дополнительных преобразований, поэтому загружаются очень быстро;
 - источника данных XML (XML Source), который представляет собой файл в формате XML на диске. В списке доступных драйверов присутствуют драйверы **SQLXMLOLEDB**, но они предназначены для других целей, а именно — для получения информации в формате XML с SQL Server;

- кроме перечисленных типов источников данных вам доступны все источники данных ODBC (а драйверы ODBC есть практически для любых баз данных). Источники данных ODBC доступны при помощи драйвера **.NET Framework Data Provider for ODBC**;
- драйвер **Flat File Source** (или, на следующем экране, **Flat File Destination**) позволяет использовать в качестве источника или назначения текстовые файлы на диске (например, с разделителями-запятыми или с полями фиксированной длины);
- для подключения рекомендуется использовать:
 - к SQL Server 2005 — драйвер **SQL Native Client**;
 - к SQL Server 7.0 и 2000 — или драйвер **.NET Framework Data Provider for SqlServer** (он рекомендован Microsoft), или драйвер **Microsoft OLE DB Provider for SQL Server** (он проще в настройке и проверен временем);
 - для подключения к SQL Server более старых версий рекомендуется использовать ODBC.

Конечно, вы можете не следовать этим рекомендациям. Например, к SQL Server 2005 можно подключиться любым из трех способов. Однако при этом будут доступны не все возможности, которые есть, например, у SQL Native Client.

С точки зрения производительности, и SQL Native Client, и .NET Framework Data Provider for SqlServer, и Microsoft OLE DB Provider for SQL Server находятся примерно на одном уровне. Несколько медленнее (за счет необходимости выполнения дополнительных преобразований) работает драйвер ODBC;

- набор драйверов, которые вы видите в списке в мастере, не является фиксированным. Например, после установки Active Directory, Exchange Server или сервера Oracle в этом списке появятся дополнительные драйверы для соответствующих источников данных.

В рассматриваемом примере нужно подключиться к SQL Server 2005, поэтому лучше всего выбрать SQL Native Client.

Набор остальных параметров, которые можно заполнить на первом экране мастера, зависят от выбранного источника данных. Если выбран SQL Native Client, нужно указать имя сервера (если в соответствующем списке его нет, то можно просто ввести его имя), режим аутентификации и базу данных на сервере (в данном случае AdventureWorks).

На следующем экране мастера **Choose a destination** (Выберите назначение) вы точно так же можете выбрать источник данных, в который данные будут перемещены. В рассматриваемом случае нужно выбрать Microsoft Excel и

ввести путь к файлу Excel (D:\Employees.xls). Если файла на диске нет, то он будет создан автоматически. На этом же экране можно выбрать версию Excel и определить, будут ли в первую строку помещены названия столбцов.

На следующем экране **Specify Table Copy or Query** (Укажите копирование таблицы или запрос) вам потребуется выбрать, будут ли выгружаться все данные из таблицы или представления или только те, которые возвращает определенный запрос (его нужно будет указать). В данном случае можно идти любым путем, но чаще требуется выгружать данные, возвращаемые запросом, поэтому переключатель в положение **Write a query to specify the data to transfer** (Написать запрос для выбора передаваемых данных).

На следующем экране **Provide a Source Query** (Обеспечить запрос для источника) нужно ввести текст запроса. В SSIS Designer можно использовать графический построитель запросов. В мастере он не предусмотрен. В данном случае текст запроса будет таким:

```
SELECT * FROM HumanResources.vEmployee;
```

Очень часто на практике требуется использовать в подобных запросах параметры, которые передаются в момент запуска пакета SSIS на выполнение (чтобы выгрузить только данные за определенный период, по определенному региону и т. п.). К сожалению, работа с параметрами возможна только из SSIS Designer. При работе с мастером эта возможность недоступна.

Проверить синтаксическую правильность запроса можно при помощи кнопки **Parse** (Разбор).

На следующем экране **Select Source Tables and Views** (Выберите таблицы и представления источника) можно сопоставить таблицы и представления на источнике с таблицами на получателе данных. При помощи кнопки **Edit** (Изменить) можно настроить точные соответствия между столбцами таблицы назначения и таблицы-получателя, а также выбрать тип данных для каждого столбца. Для рассматриваемой ситуации с файлом Excel это большого значения не имеет, но если вы переносите данные между базами данных, то такая возможность настройки может оказаться очень удобной. Обратите внимание, что для каждого столбца вы можете выбрать не только имя столбца в таблице назначения (существующей или создаваемой), но и специальное значение **Ignore** (Игнорировать). Значения из этого столбца не будут передаваться в таблицу назначения.

На этом же экране **Column Mappings** (Привязки столбцов) можно определить судьбу таблицы назначения. В вашем распоряжении есть несколько вариантов:

- Create destination table** (Создать таблицу назначения);
- Delete rows in destination table** (Удалить записи в таблице назначения) — т. е. предварительно очистить таблицу;

- Append rows to the destination table** (Добавить записи в таблицу назначения);
- Drop and re-create a destination table** (Удалить и создать заново таблицу назначения) — если таблица с таким именем уже есть в базе данных, то все данные теряются.

При помощи кнопки **Edit SQL** (Изменить код SQL) вы можете при желании вмешаться в создание таблицы, явно определив код команды *Create Table*.

В данной ситуации вставка будет производиться на лист Excel, поэтому можно оставить для всех параметров на экране **Column Mappings** значения по умолчанию.

Следующий экран мастера называется **Save and Execute Package** (Сохранить и запустить пакет). С его помощью вы можете выбрать следующие действия:

- Execute immediately** (Выполнить немедленно) — т. е. запустить пакет на выполнение сразу после завершения работы мастера;
- Save SSIS Package** (Сохранить пакет SSIS) — сохранить пакет SSIS в базе данных *msdb* на SQL Server или просто как файл на диске. В любом случае

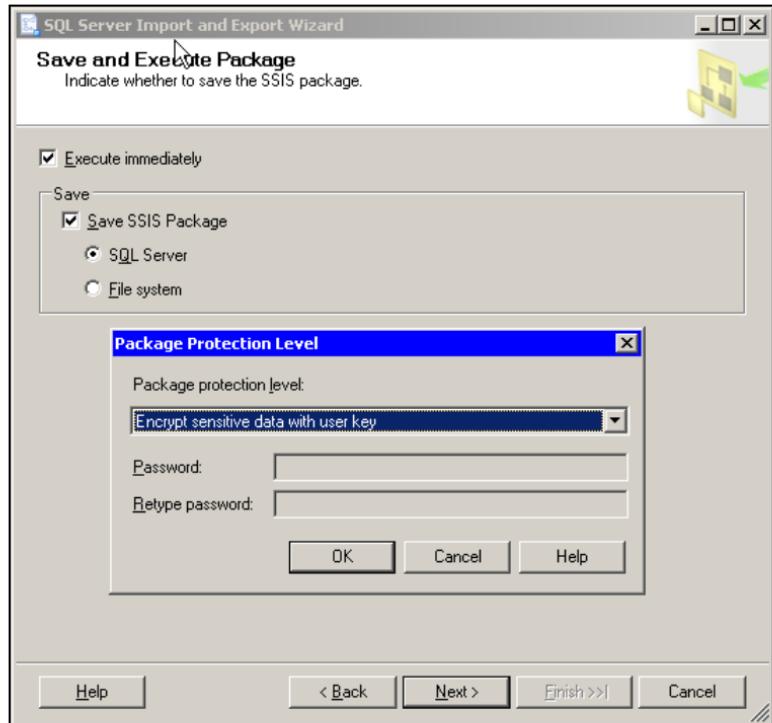


Рис. 10.2. Экран сохранения пакетов SSIS

при нажатии на кнопку **Next** (Дальше) будет открыто окно, в котором вам потребуется определить уровень защиты нашего пакета (рис. 10.2). Про сохранение пакетов и их защиту будет рассказано в разд. 10.27. В рассматриваемом примере вполне можно оставить значения, которые представлены на рис. 10.2.

На следующем экране **Save SSIS Package** необходимо определить параметры сохранения — имя пакета, его описание и сервер, на который он будет помещен (с параметрами аутентификации при подключении к этому серверу). Назовите ваш пакет, например, `WizardPackage1`.

Последнее, что осталось сделать, — запустить пакет на выполнение при помощи кнопки **Finish** (Завершить) и убедиться, что его сохранение и выгрузка данных происходят нормально. По окончании при помощи кнопки **Report** (Отчет) можно сгенерировать отчет о выполнении пакета.

Настроить загрузку и выгрузку данных при помощи мастера можно буквально за минуту. Однако из этого мастера вам доступна очень небольшая часть возможностей SSIS. Все возможности работы с пакетами SSIS доступны при использовании SSIS Designer.

10.4. Пример работы с SSIS Designer

По наблюдениям автора, даже специалисты с опытом работы с DTS часто не могут сразу разобраться с SSIS Designer — настолько многое появилось новых возможностей. Поэтому в этом разделе постараемся просто познакомиться с возможностями SSIS Designer на простом примере. Подробно работа с различными компонентами SSIS Designer рассматривается в следующих разделах.

Задание будет очень простым: предположим, что нужно сделать то же самое, что и в предыдущем разделе (выгрузить данные из представления `AdventureWorks.HumanResources.vEmployee` в файл Excel), но уже средствами SSIS Designer. Экспорт данных должен быть произведен в файл Excel `D:\Employees1.xls`.

SSIS Designer встроен в Business Intelligence Development Studio, и для его запуска вам потребуется запустить это программное средство, основанное на Visual Studio. Если на вашем компьютере установлена Visual Studio 2005, то вы можете работать с пакетами SSIS напрямую из нее.

После запуска Business Intelligence Development Studio вам потребуется создать новый проект (при помощи меню **File | New | Project** (Файл | Новый | Проект)). В окне **New Project** (Новый проект) в списке типов проекта нужно выбрать **Business Intelligence Projects**, а в списке доступных шаблонов в пра-

вой части экрана — Integration Services Project. После того, как вы введете информацию об имени создаваемого проекта и размещении его файлов и нажмете кнопку **OK**, проект будет создан. При этом автоматически откроется окно SSIS Designer с загруженным в него новым пустым пакетом SSIS.

В окне SSIS Designer есть четыре вкладки:

- **Control Flow** (Поток управляющих элементов) — на ней производится управление ходом выполнения пакетов;
- **Data Flow** (Поток данных) — эта вкладка специально предназначена для редактирования элементов Data Flow Task, при помощи которых определяются параметры перемещения данных;
- **Event Handlers** (Обработчики событий) — при помощи этой вкладки регистрируются обработчики событий, которые могут возникнуть при выполнении пакета (например, связанные с ошибками);
- **Package Explorer** (Проводник пакета) — это просмотрщик компонентов объекта, представляющего пакет SSIS.

Работа на этих вкладках будет подробно рассмотрена в следующем разделе. Для того чтобы решить задачу, достаточно будет выполнить лишь несколько простых операций.

Первое, что нужно будет сделать, — разместить на вкладке **Control Flow** единственный элемент Data Flow Task. Для этого нужно сделать видимым окно **Toolbox** (например, при помощи меню **View | Toolbox** (Вид | Элементы управления)) и перетащить из него элемент Data Flow Task на светло-розовое поле SSIS Designer. Выглядеть результат может, например, так, как представлено на рис. 10.3.

Следующее, что вам нужно сделать, — настроить параметры созданной вами задачи Data Flow Task. Для этого лучше всего щелкнуть по ней правой кнопкой мыши и в контекстном меню выбрать **Edit**. Поскольку задача такого типа у вас одна, то можно просто перейти на вкладку **Data Flow** в SSIS Designer. В любом случае откроется вкладка **Data Flow**. Пока она пуста, только надпись в центре этой вкладки советует начать с размещения источника данных. Так и следует поступить, но вначале нужно создать соединение.

Соединение в Data Flow Task создается при помощи объектов Connection Managers (Менеджеры подключений). Область для работы с этими объектами (она так и называется — **Connection Managers**) по умолчанию помещается в нижнюю часть экрана. В этой области при помощи графических средств можно создать соединение с источником данных, которое потом можно использовать в пакете.

Для наших целей потребуются два источника данных: одно для подключения к представлению `HumanResources.vEmployee` в базе данных AdventureWorks на

локальном сервере SQL Server 2005, а второе — для подключения к файлу Excel. В любом случае для создания подключения нужно щелкнуть правой кнопкой мыши по области **Connection Managers** и в контекстном меню выбрать нужный тип подключения.

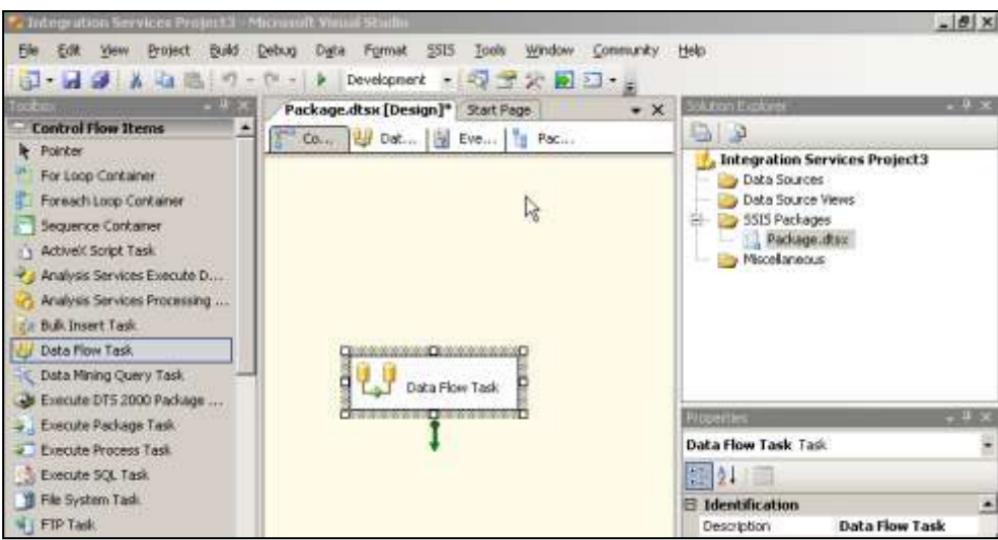


Рис. 10.3. Работа с SSIS Designer

Если вы хотите сделать все так же, как при работе с мастером, то для подключения к SQL Server 2005 нужно использовать SQL Native Client. Это надстройка над OLE DB, и поэтому в контекстном меню при создании нового менеджера подключений нужно выбрать **New OLE DB Connection** (Новое соединение OLE DB). Затем в открывшемся окне **Configure OLE DB Connection Manager** (Настроить менеджер подключений OLE DB) нужно нажать кнопку **New** (Новый). Откроется окно **Connection Manager**, в котором нужное значение **Native OLE DB\SQL Native Client** будет подставлено по умолчанию (рис. 10.4). В этом окне вам потребуется указать только имя базы данных, режим аутентификации и базу данных, а также для проверки нажать кнопку **Test Connection** (Проверить соединение).

Настройка второго подключения (для файла Excel D:\Employees.xls) выглядит точно так же. Нужно щелкнуть правой кнопкой мыши по пустому пространству в области **Connection Managers** и в контекстном меню выбрать **New Connection** (Новое подключение). В открывшемся списке нужно выбрать **EXCEL** и ввести имя файла и версию Excel (вы можете также указать, будут ли в первую строку в файле Excel помещаться имена столбцов).

После того, как оба объекта менеджеров подключений созданы, можно приступить к настройке самой задачи Data Flow Task. Для этого на вкладку **Data**

Flow нужно перетащить источник (*source*) и назначение (*destination*) для передаваемых данных. Наиболее подходящий в данном случае источник — это **OLE DB Source**. Его объект нужно перетащить из **Toolbox** на поле вкладки **Data Flow**. Назначение для данных — это файл Excel, поэтому выбираем **Excel Destination** (назначения находятся в самом низу списка элементов в **Toolbox**).

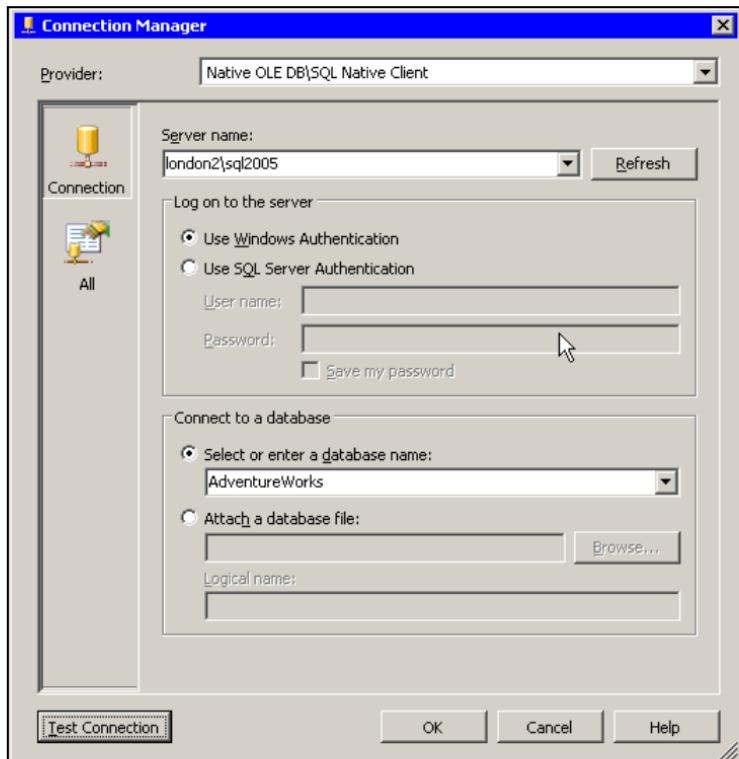


Рис. 10.4. Настройка **Connection Manager**

Пока элементы у вас не настроены, они будут помечены красным кружком с крестиком. Определить их параметры можно, щелкнув по объекту источника или назначения правой кнопкой мыши и в контекстном меню выбрав команду **Edit**. Если вы выполните эту операцию для вашего источника данных, то откроется редактор OLE DB Source Editor с тремя вкладками. На первой вкладке нужно выбрать созданный вами менеджер подключений для базы данных AdventureWorks, определить режим доступа к данным (в рассматриваемом случае это **SQL Command** (Команда SQL)) и ввести текст команды. Обратите внимание, что в отличие от мастера, в вашем распоряжении появился очень удобный построитель запросов, который открывается при нажа-

тии на кнопку **Build Query** (Построить запрос). С его помощью очень удобно создавать сложные запросы с большим количеством соединений. Здесь вы можете определить использование запроса (или имени таблицы/представления), текст которого берется из переменной пакета. В данном случае в поле **Data access mode** (Режим доступа к данным) выберите **SQL Command**, а текст запроса может быть таким же, как и при использовании мастера:

```
SELECT * FROM HumanResources.vEmployee;
```

Если нажать кнопку **Preview** (Предпросмотр), то можно убедиться, что возвращаются требуемые данные.

На вкладке **Columns** (Столбцы) вы можете выбрать столбцы, который будут возвращаться с источника, а на вкладке **Error Output** (Вывод ошибок) — поведение при возникновении ошибок в каждом из столбцов. В рассматриваемом случае значения на этих вкладках можно оставить без изменений.

После настройки источника данных нужно указать, что возвращаемые данные будут передаваться назначению **Excel Destination**. Делается это очень простым, но несколько неожиданным способом. Нужно выделить настроенный вами источник **OLE DB Source**. Тогда от него появятся две стрелки — зеленая и красная, направленные вниз. Нужно зацепить мышью зеленую стрелку (она означает успешное извлечение данных) и перетащить ее на назначение **Excel Destination**. Если все будет сделано правильно, то схема вашей Data Flow Task будет выглядеть так, как представлено на рис. 10.5.

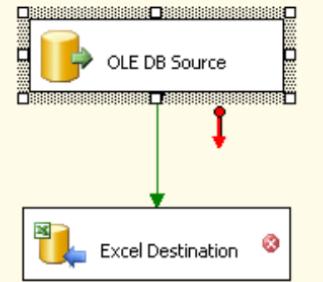


Рис. 10.5. Определение связей между источником и назначением

Следующее, что нужно сделать, — настроить назначение **Excel Destination**. Точно так же откройте его свойства при помощи команды **Edit** в контекстном меню и выберите требуемые параметры на вкладке **Connection Managers** (Менеджеры подключений). В списке **OLE DB Connection Manager** (Менеджер подключений OLE DB) нужно выбрать, конечно, созданный вами **Excel Connection Manager**, а в списке **Data Access Mode** — **Table or View** (Таблица или представление). Если файла Excel еще не существует, то в спи-

ске **Name of the Excel Sheet** (Имя таблицы Excel) появится надпись **No tables or views could be loaded** (Нельзя загрузить ни одну таблицу или представление). В этом случае проще всего воспользоваться кнопкой **New** и, просмотрев команду на создание таблицы, нажать кнопку **OK**. При этом произойдет сразу несколько вещей:

- на диске будет создан файл Excel с указанным вами именем;
- в первой таблице этого файла будет создан именованный диапазон с называнием, указанным в команде `CREATE TABLE` (по умолчанию — `Excel_Destination`);
- если при настройке менеджера подключений был установлен соответствующий флажок, то в первую строку этого именованного диапазона будут помещены названия столбцов, которые возвращают запрос к SQL Server;
- название именованного диапазона (**Excel Destination**, уже без подчеркивания) будет помещено в список **Name of Excel Sheet** (Имя таблицы Excel).

Вам осталось только перейти на вкладку **Mappings** (Привязки), просмотреть информацию для каждого столбца и нажать кнопку **OK**.

Все, ваш пакет создан. Для того чтобы убедиться, что он выполняет все нужные действия, достаточно запустить его на выполнение при помощи меню **Debug | Start Debugging** (Отладка | Запустить с отладкой) или **Debug | Start without debugging** (Отладка | Запустить без отладки). Если вы запустили пакет с отладкой, после окончания его работы завершите отладку при помощи меню **Debug | Stop Debugging** (Отладка | Остановить отладку).

Как вы видите, создание пакета в SSIS Designer несколько сложнее, чем при использовании мастера, и занимает больше времени. Однако возможностей в SSIS Designer намного больше.

При желании вы можете сравнить созданный вами вручную пакет SSIS и пакет, который был создан мастером. Для этого в окне **Solution Explorer** щелкните правой кнопкой мыши по контейнеру **SSIS Packages** (Пакеты SSIS) и в контекстном меню выберите **Add Existing Packages** (Добавить существующие пакеты). Затем в открывшемся окне выберите в поле **Package location** (Местонахождение пакета) значение `SQL Server`, введите имя вашего сервера и нажмите на кнопку справа от поля **Package path** (Путь к пакету). Откроется список пакетов и планов обслуживания баз данных, которые находятся на сервере (они также являются пакетами SSIS). Выберите созданный вами пакет (в примере он назывался `WizardPackage1`) и нажмите кнопку **OK**. Чтобы открыть этот пакет в SSIS Designer, достаточно в его контекстном меню выбрать команду **Open** (Открыть).

Созданный мастером пакет SSIS будет чуть-чуть отличаться от созданного вручную. Создание файла Excel было выполнено в ходе настройки пакета, а в

пакете мастера создание файла Excel производится при помощи специального элемента **Preparation SQL Task** на вкладке **Control Flow**. В остальном оба пакета одинаковы.

10.5. Менеджеры подключений

Как правило, работа с пакетом SSIS начинается с определения соединений (*connections*). Настройка соединений производится при помощи так называемых менеджеров соединений (*Connection Managers*) в одноименном окне, которое находится в нижней части вкладок **Control Flow**, **Data Flow** и **Event Handlers**. Менеджер подключения — это графическая оболочка, при помощи которой очень удобно производить настройку подключения к источнику данных и создавать строки подключения (*connection string*). Стока подключения создается и записывается в пакет автоматически. При желании ее можно будет потом просмотреть или отредактировать из окна редактора кода XML для вашего пакета (редактор кода можно открыть при помощи меню **View | Code** (Вид | Код) в SSIS Designer).

Проще всего создать новый менеджер подключения, щелкнув правой кнопкой мыши по пустому пространству в окне **Connection Managers** и выбрав в контекстном меню нужный тип создаваемого менеджера соединения. Если вы выберете тип **New Connection**, то в вашем распоряжении будут все доступные типы менеджеров соединения. Новый менеджер соединения можно создать также при помощи меню **SSIS | New Connection**.

Рассмотрим каждый тип соединения:

- **ADO** — для подключения к источнику данных будут использоваться средства объекта *Connection* обычного ADO (не ADO.NET). Для подключения вы можете использовать любой драйвер OLE DB, установленный на вашем компьютере. Этот способ очень надежный и нересурсоемкий, но у вас будет меньше возможностей, по сравнению, например, с подключением по ADO.NET;
- **ADO .NET** — для подключения к источнику данных будут использоваться средства объектной библиотеки ADO.NET. Можно использовать три "родных" поставщика .NET: *SqlClient Data Provider*, *OracleClient Data Provider* и *OdBC Data Provider*, а также любой драйвер ODBC. Этот тип менеджера подключений обладает наиболее полными возможностями, и его очень удобно использовать при интеграции пакетов с приложениями на .NET-совместимых языках. Однако при подключении по ADO.NET требуется больше ресурсов, чем при подключении по обычному ADO;
- **EXCEL** — этот тип менеджера подключения используется, конечно, для подключения к Excel. Вы уже использовали его в примерах предыдущих

разделов. При его настройке вам потребуется выбрать версию файла Excel (начиная с версии 3.0) и имя файла, к которому вы подключаетесь. На самом деле этот тип подключения выделен как отдельный только для удобства. Реально для него используется подключение по OLE DB с поставщиком Microsoft.Jet.OLEDB.4.0, в чем легко убедиться, заглянув в код XML пакета;

- **FILE** — это новый тип менеджера подключения, аналогов которому не было в DTS. С его помощью вы можете указать файл или каталог с набором файлов. Этот файл или каталог затем могут использоваться самыми разными задачами. Например, если в указанном файле записан скрипт Transact-SQL, этот скрипт сможет выполнить задачу Execute SQL Task. Кроме имени файла или каталога, при настройке этого менеджера подключения вам потребуется также указать режим подключения: будет использован существующий файл или каталог, или файл будет создан во время выполнения пакета;
- **FLATFILE** — этот тип менеджера подключений предназначен для работы с текстовыми файлами с наборами записей (на отечественных предприятиях такие файлы часто называют "макетными"). Значения в столбцах в таких файлах обычно отделяются разделителями (запятая, точка с запятой или т. п.), а переход на новую строку означает начало новой записи. Для этого менеджера подключения вы можете определить множество параметров (большинство из которых вполне очевидно) — кодировку, разделитель, название каждого столбца, тип данных и т. п.;
- **FTP** — этот экзотический тип менеджера подключений предназначен для настройки соединения с сервером FTP. Обычно он настраивается только для задачи FTP Task (ее можно выбрать на вкладке **Control Flow** в SSIS Designer). Для этого соединения можно указать сервер FTP и используемый им порт, имя пользователя и пароль, активный или пассивный режим работы. Этот менеджер подключений может использоваться для передачи данных в обоих направлениях;
- **HTTP** — еще один тип менеджера подключений, предназначенный для обмена данными по интернет-протоколам. Этот тип предназначен не просто для того, чтобы скачать какой-либо файл, доступный по протоколу HTTP, а для взаимодействия с Web-службами. Обычно он используется задачей, которая называется Web Service Task;
- **MSMQ** — это путь к очереди Microsoft Message Queue Services, которая может быть использована для обмена сообщениями между пакетами SSIS (или приложениями). Применяется этот тип менеджера подключений, конечно, для задачи Message Queue Task. Эта очень популярная и незаменимая во многих ситуациях задача будет рассмотрена в разд. 10.11;

- **MSOLAP90** — этот тип менеджера подключений (другое его название — Analysis Services Connection Manager) предназначен для подключения к базам данных OLAP. При его настройке вы можете указать или компьютер, на котором работает служба Analysis Services, или путь к проекту Analysis Services в текущем решении. Чаще всего этот тип используется вместе с Analysis Services Processing Task для автоматизации процессинга кубов;
- **MULTIFILE** и **MULTIFLATFILE** — эти аналоги подключений FILE и FLATFILE предназначены для работы с несколькими файлами одновременно. Можно указывать как списки файлов (разделяя их имена вертикальной чертой '|'), так и все файлы в определенном каталоге (при помощи символа звездочки '*');
- **ODBC** — этот менеджер подключений, конечно, используется для подключения к источникам данных при помощи драйверов ODBC. Можно использовать существующий источник данных ODBC или создать новый прямо в процессе настройки этого менеджера подключений. ODBC — это унаследованная технология, которая работает медленнее, чем подключения по OLE DB. Однако в некоторых ситуациях, обычно при подключении к не самым распространенным источникам данных (Informix, MySQL и т. п.), без нее не обойтись;
- **OLE DB** — это, видимо, самый распространенный тип менеджера подключений. Он использует драйверы (поставщики) OLE DB. Этот тип менеджера подключений можно использовать для подключения к самым разным источникам данных, например, SQL Server, Oracle, Access и Excel;
- **SMOServer** — этот новый тип менеджера соединений использует средства объектной модели SMO (*см. разд. 9.2*) для подключения к SQL Server 2005. Обычно этот тип используется для задач по переносу объектов SQL Server (Transfer Databases Task, Transfer Error Messages Task, Transfer Jobs Task, Transfer Logins Task и т. п.);
- **SMTP** — этот тип используется для настройки подключения к почтовому серверу, работающему по протоколу SMTP. Можно указать адрес сервера, режим аутентификации и защиты по SSL. Обычно этот тип менеджера подключений используется только для задачи Send Mail Task;
- **SQLMOBILE** — этот тип менеджера подключений, как правило, используется, когда в ходе выполнения пакета SSIS задача Data Flow Task загружает данные в назначение SQL Server Mobile (т. е. в базу данных SQL Server 2005 Mobile Edition). Для этой задачи вы можете настроить имя базы данных SQL Server 2005 Mobile Edition, имя пользователя и пароль;
- **WMI** — при помощи этого типа менеджера подключений вы можете подключиться к определенному пространству имен WMI и указать имя пользо-

зователя и пароль для подключения. Обычно этот тип используется для задач WMI Data Reader Task и WMI Event Watcher Task.

Какой бы тип менеджера подключений вы не выбрали, в любом случае созданные вами объекты появятся в окне **Connection Managers**. При помощи команды **Edit** в контекстном меню для данного менеджера вы можете еще раз открыть графический интерфейс настройки его параметров, а при помощи команды **Properties** (Свойства) — просмотреть список свойств этого менеджера и при необходимости изменить их значения.

Менеджеры также можно создавать и изменять программно в ходе выполнения пакета. Обычно эта операция производится при помощи задачи ActiveX Script Task или средствами внешней программы, которая запускает пакет на выполнение. Однако объектная модель пакетов SSIS — очень большая тема, и рассматриваться в этой книге она не будет.

10.6. Работа с Data Flow Task

10.6.1. Что такое Data Flow Task

Обычно после того, как вы создали объекты менеджеров подключений в окне **Connection Managers**, следующее, что необходимо сделать, — создать нужные задачи на вкладке **Control Flow**.

Самая распространенная задача в пакетах SSIS — Data Flow Task. Ее основное назначение — определять один или несколько потоков данных (*data flow*), т. е. направлений по перекачке данных с одного источника на другой (с возможностью их одновременной проверки и преобразования).

В каждой задаче Data Flow Task можно определить как один поток данных, так и несколько. Выбор в основном зависит от того, допустима ли одновременная параллельная перекачка данных (в этом случае в одной Data Flow Task можно использовать несколько потоков) или данные необходимо перекачивать последовательно (тогда придется создать несколько задач Data Flow Task с единственным потоком в каждой).

По причине большой важности этой задачи в SSIS Designer для нее предусмотрена отдельная вкладка. Однако работать на этой вкладке, выбирая и настраивая элементы Data Flow Task, можно только тогда, когда на первой вкладке SSIS Designer, которая называется **Control Flow**, будет добавлена в проект хотя бы одна задача Data Flow Task. Если после добавления Data Flow Task на вкладке **Control Flow** вы щелкните по этой задаче правой кнопкой мыши и в контекстном меню выберите команду **Edit**, то автоматически откроется вкладка **Data Flow**, в которой можно выбирать и настраивать элементы Data Flow Task.

10.6.2. Элементы Data Flow Task

В Data Flow Task предусмотрено четыре типа элементов:

- **Data Flow Sources** (Источники потока данных) — это источники данных, с которых Data Flow Task будет скачивать данные для помещения в пункт назначения. В вашем распоряжении есть шесть видов источников. Все они доступны в разделе **Data Flow Sources** в **Toolbox**. Вам потребуется перетащить нужный источник на пустое место на вкладке **Data Flow** и определить его свойства (например, для большинства источников вам нужно выбрать менеджер подключения);
- **Data Flow Destinations** (Назначения потока данных) — пункты назначения для данных, куда эта задача будет помещать полученные с источника и преобразованные данные. Все 11 назначений доступны из раздела **Data Flow Destinations** в **Toolbox**. Работа с ними производится точно так же, как и с источниками. Про источники и пути назначения данных подробно рассказывается в *разд. 10.6.3*;
- **Data Flow Transformations** (Преобразования потока данных) — это преобразования, которые будут выполнены в процессе перекачки данных. Множество преобразований доступно из раздела **Data Flow Transformations** в **Toolbox**. Обычно они помещаются между источником и назначением данных (рис. 10.6).

Подробнее про различные типы преобразований будет рассказано в *разд. 10.6.4*.

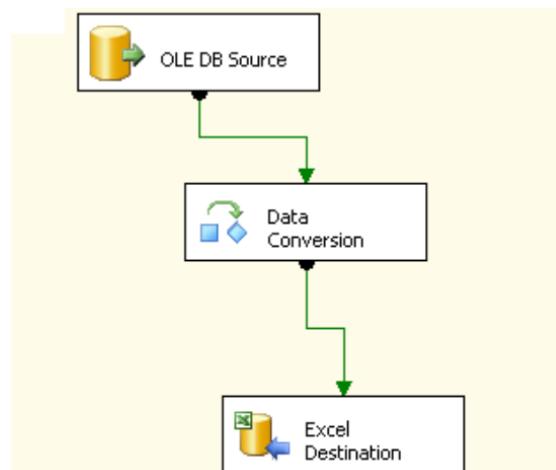


Рис. 10.6. Применение преобразований в SSIS Designer

- **Paths** (Пути) — это последние компоненты Data Flow Task. В отличие от предыдущих компонентов, их не надо перетаскивать с **Toolbox**. Они выглядят как зеленые и красные стрелки, которые используются для соединения элементов Data Flow Task друг с другом. Зеленая стрелка означает нормальное прохождение данных, а красная — направление передачи данных с возникновением ошибок. Эти стрелки становятся видимыми, если выделить соответствующий элемент в SSIS Designer. Пути не только определяют логику выполнения задачи, но и могут также использоваться для отладки. Подробнее про них будет рассказано в разд. 10.6.5.

10.6.3. Источники и назначения Data Flow Task

Как правило, работа с большинством источников и назначений Data Flow Task выглядит одинаково. Вначале назначение нужно перетащить на розовое поле на вкладке **Data Flow** из **Toolbox**, а затем настроить его свойства. Для большей части источников и назначений вам потребуется указать в свойствах созданный вами менеджер подключений подходящего типа (неподходящие будут просто не видны).

Для многих источников и назначений Data Flow Task, таких как EXCEL, OLE DB, FLATFILE, SQL Server, настройка параметров очевидна. Понятно также, с какими менеджерами подключений они работают. Далее представлена информация только о тех источниках и назначениях, при настройке которых могут возникнуть вопросы, и о тех, которые не используют менеджеры подключений:

- **DataReader Source** — этот источник позволяет получить его данные при помощи объекта `DataReader` объектной библиотеки ADO.NET. Требует менеджера подключений типа ADO.NET;
- **DataReader Destination** — в отличие от **DataReader Source**, это назначение предназначено для единственной цели: помещения данных в объект `DataSet` в оперативной памяти. Как и объект `DataReader`, объект `DataSet` определен в объектной библиотеке ADO.NET. Обычно это назначение используется только в том случае, когда пакет запускается из внешнего приложения, которое и будет дальше работать с созданным объектом `DataSet`;
- **Raw File Source** и **Raw File Destination** — источники и назначения этого типа работают с двоичными файлами в родном формате SQL Server 2005. Обычно на одном сервере SQL Server данные экспортируются в файл такого формата, а потом на другом сервере производится загрузка этого файла. Формат файла не документирован. Поскольку никаких преобразований данных при этом не производится, экспорт и импорт данных этим способом производится очень быстро. Эти источник и назначение не ис-

пользуют менеджер подключений. Для них также не предусмотрена красная стрелка для обработки ошибок;

- XML Source — загрузка данных из формата XML. Этот тип источника также не используют менеджеры подключений. Вместо этого у вас есть возможность указать, откуда будут поступать данные в формате XML: из явно указанного файла на диске, из файла, имя которого определяется при помощи переменной, или просто из строковой переменной. Вы можете также указать свой файл схемы XML с описанием структуры этого файла или сгенерировать его автоматически.

Обратите внимание, что назначения типа XML Destination не предусмотрено. Выгружать данные с SQL Server в формат XML вам предлагается при помощи OLE DB Destination. Это назначение должно использовать менеджер подключения OLE DB, в котором выбран поставщик **SQLXMLOLEDB** или **SQLXMLOLEDB4.0**;

- Data Mining Model Training Destination — это очень специальное назначение. Данные, которые получаются с источника, сразу же используются для обучения модели добычи данных. Модель добычи данных (*data mining model*) — это специальный объект Analysis Services, который используется для автоматизированного выявления закономерностей в больших массивах данных и для прогнозирования. Например, на основе накопленной информации о ранее выданных кредитах можно при помощи моделей данных рассчитать вероятность невозврата кредита заемщиком с определенными параметрами. Это назначение требует менеджера подключения типа MSOLAP90 (Analysis Services Connection Manager);
- Dimension Processing Destination — еще одно специальное назначение Analysis Services, которое требует наличия менеджера подключения типа MSOLAP90. Данные, получаемые с источника, сразу используются для процессинга измерения в базе данных OLAP на Analysis Services;
- Partition Processing Destination — это последнее из набора назначений Analysis Services. Работает аналогично Dimension Processing Destination, однако данные используются для процессинга раздела куба, а не измерения. В большинстве кубов используется только один раздел, поэтому это назначение обычно можно использовать для процессинга (загрузки данных) куба целиком.
- Recordset Destination — данными, поступающими с источника, будет автоматически заполняться объект Recordset обычного ADO. Точно так же, как и назначение DataReader Destination, имеет смысл использовать это назначение только тогда, когда пакет вызывается внешним приложением, которое будет дальше работать с созданным объектом Recordset.

10.6.4. Преобразования Data Flow Task

Одна из самых ценных возможностей Data Flow Task — это преобразования (*transformations*). Преобразования — это элементы задачи Data Flow Task, которые предназначены для выполнения каких-либо действий с данными в ходе их перемещения с источника в место назначения. Работа со всеми преобразованиями выглядит одинаково: необходимо перетащить этот элемент из раздела **Transformations** (Преобразования) в **Toolbox** на свободное место на вкладке **Data Flow** в SSIS Designer, а затем соединить преобразование стрелками с источниками, назначениями или другими преобразованиями. После этого нужно будет настроить свойства данного преобразования.

Обратите также внимание, что в преобразованиях нужно указывать не все столбцы источника, а только те, с которыми действительно производятся какие-то преобразования. Значения остальных столбцов просто "пройдут сквозь" преобразование и станут доступны для назначения столбцам источника в исходном виде. Чаще всего преобразования просто добавляют новые столбцы, которые можно использовать в назначении (они появятся на вкладке **Mappings**).

В Data Flow Task предусмотрено много типов преобразований (если быть точным, то 28). Подробное рассмотрение каждого из них потребовало бы слишком много места. Поэтому в этом разделе будет приведена только краткая характеристика каждого преобразования, чтобы можно было составить представление о том, в каких ситуациях оно подойдет наилучшим образом.

- **Aggregate** (Агрегат) — это преобразование предназначено для расчета итоговых значений по столбцам перекачиваемых записей. Можно посчитать сумму, среднее значение, минимум и максимум, общее количество значений и т. п. Это преобразование действует аналогично оператору GROUP BY в запросе SELECT в Transact-SQL. Вы выбираете столбцы на источнике, по которым будет проводиться группировка, и в результате этого преобразования на выходе в ваше распоряжение предоставляются дополнительные столбцы с итогами по группам.

Этим преобразованием нужно пользоваться очень осторожно. Оно не переносит расчет агрегатов на сервер (на SQL Server, например, преобразование отправит запрос вида SELECT * FROM HumanResources.vEmployee), а скачивает все записи в оперативную память и уже по ним считает итоги. Если записей у вас много, то оперативной памяти может просто не хватить. Кроме того, появляется искушение использовать SSIS для генерации отчетов (обычно в отчетах как раз и нужны итоги), что неправильно. Для создания отчетов и работы с ними предназначен другой компонент SQL Server 2005 — Reporting Services.

- **Audit** (Аудит) — это преобразование позволяет использовать в вашем пакете значения системных переменных (например, имя пользователя, который запустил пакет, имя компьютера, имя пакета и т. п.). Фактически у вас появляется новый источник данных, столбцы которого соответствуют системным переменным.
- **Character Map** (Карта символов) — это преобразование позволяет применить к значениям, получаемым с источника, строковые функции, например, перевод в верхний и нижний регистр или перестановка символов в строковом значении в обратном порядке. Остальные преобразования относятся к восточным языкам. Для каждого столбца можно указать в свойствах этого преобразования два назначения: **In-place upgrade** (меняться будут значения того же столбца) и **New column** (результаты преобразования будут помещены в новый столбец, а старый столбец останется в неприкосновенности).
- **Copy Column** (Копирование столбца) — это самый простой тип преобразования. Он используется для того, чтобы "размножить" существующий столбец, сделав из него два (с одинаковыми значениями). Затем такой дополнительный столбец можно использовать для других преобразований.
- **Conditional Split** (Условное разделение) — этот тип преобразования более всего похож на оператор SELECT CASE. Для него можно назначить несколько пунктов назначения. Это преобразование проверяет каждую запись источника на соответствие набору определенных в нем условий (для которых устанавливается очередность проверки), и результат записывается в соответствующее назначение в зависимости от того, какое именно условие вернуло для данной записи значение TRUE. Можно установить также назначение по умолчанию: данные будут записываться в это назначение, если проверка всех условий вернула FALSE.
- **Data Conversion** (Преобразование типов данных) — одно из самых простых преобразований. Позволяет в процессе перекачки данных изменять тип данных. Обратите внимание, что это преобразование просто добавляет новые столбцы, которые можно будет использовать в назначении. Столбцы в исходном виде также остаются в вашем распоряжении.
- **Data Mining Query** (Запрос к модели добычи данных) — это очень экзотический тип преобразования. Он позволяет для каждой записи на источнике выполнить запрос на языке DMX к модели добычи данных на Analysis Services. При этом значения из источника можно использовать как параметры, передаваемые этому запросу. Результаты запроса можно использовать в назначении.
- **Derived Column** (Производные столбцы) — одно из самых популярных преобразований. Позволяет применять встроенные функции SSIS (строко-

вые, математические, даты/времени, преобразования типов данных и т. п.) для значений в столбцах источника. При этом значения в существующем столбце можно заменять новыми значениями, а можно создать новый столбец, который затем можно будет использовать в назначении.

□ **Export Column** (Экспортировать столбец) и **Import Column** (Импортировать столбец) — эти два преобразования предназначены для работы с данными BLOB (большими двоичными типами данных). На SQL Server к ним относятся столбцы `text`, `ntext` и `image`. Для преобразования **Export Column** необходимы два столбца: с данными BLOB и с обычными строковыми значениями. В столбце со строковыми значениями должны находиться имена файлов. Это преобразование запишет двоичные данные для каждой записи в файл на диске с именем, которому будет соответствовать название текстового столбца. Наиболее очевидное использование этого преобразования — экспорт изображений из базы данных в файлы на диске.

Преобразование **Import Column** предназначено для обратной задачи — импорт двоичных данных из файлов на диске в базу данных. В столбце источника должны быть указаны имена файлов с путями. Для каждой записи источника будет открываться соответствующий файл и записываться в столбец назначения.

□ **Fuzzy Grouping** (Нечеткая группировка) — это достаточно сложное преобразование. В нем используются элементы искусственного интеллекта для группировки записей на источнике. В свойствах этого преобразования вам нужно будет выбрать один или несколько столбцов источника, для которых будет проводиться группировка, допустимые строковые преобразования и уровень "похожести" для группировки (от 0,00 до 0,99). Записи на источнике останутся в неприосновенности, но для записи в назначение можно будет использовать в добавление к исходным столбцам источника еще три дополнительных столбца:

- `_key_in` — уникальный идентификатор записи источника;
- `_key_out` — идентификатор группы, к которой преобразование "решило" отнести эту запись;
- `_score` — счет, значение от 0 до 0,99, которое отражает степень похожести данной записи на "идеальную" (*canonical*) для данной группы.

Это преобразование отличается очень большой ресурсоемкостью. Кроме того, оно требует наличия соединения с базой данных SQL Server 2005, в которой это преобразование будет создавать необходимые временные таблицы.

Подобное преобразование может использоваться, например, для обработки значений, полученных путем распознавания отсканированного текста рукописных форм.

- **Lookup** (Обращение к внешнему источнику) — очень удобное во многих ситуациях преобразование. Для каждой записи из источника будет выполнен специальный запрос к внешнему источнику данных (при этом значения из первого источника будут использоваться как параметры внешнего запроса). Затем данные из возвращаемого запроса можно использовать вместе со столбцами источника в назначении. Например, представьте себе такую ситуацию: в таблице на SQL Server у вас находятся номера продуктов, а расшифровка этих номеров находится, например, в файлах FoxPro. При помощи этого преобразования вы можете получить расшифровку для каждого продукта и записать ее вместе с другими столбцами в назначение.
- **Fuzzy Lookup** (Нечеткое обращение к внешнему источнику) — это так называемые нечеткие сравнения. Если в обычном преобразовании **Lookup** просмотр производится в соответствии с четкой логикой запросов SQL, то в **Fuzzy Lookup**, как и в **Fuzzy Grouping**, вы можете задать "уровень похожести" от 0 до 0,99. Соответственно, из внешнего источника данных будут возвращаться не только совпадающие значения, но и "похожие" на значения с источника.
- **Merge** (Слияние) и **Union All** (Полное объединение) — эти преобразования позволяют поместить одинаковые по формату данные из нескольких источников в одно назначение. Например, если филиалы передают вам отчеты о проведенных операциях в виде макетных текстовых файлов (в терминологии SQL Server — *flat files*, плоские файлы), то преобразования **Merge** и **Union All** позволят в ходе одной операции поместить данные из всех текстовых файлов в таблицу SQL Server. Отличие между ними заключается в том, что преобразование типа **Merge** перед загрузкой данных в назначение вначале получает их из всех источников и сортирует их, а **Union All** не выполняет сортировку и загружает данные последовательно в том порядке, как они поступили из источников.
- **Merge Join** (Соединение слиянием) — это преобразование, которое обеспечивает объединение данных из разных таблиц по ключу. Обратите внимание на следующие моменты:
 - если речь идет о соединении данных из двух таблиц на одном источнике данных (например, в базе данных SQL Server), то намного проще будет просто указать в качестве источника запрос, а объединение (т. е. оператор `JOIN`) использовать в самом этом запросе. SQL Server выполнит такое соединение быстрее и эффективнее, чем пакет SSIS. Поэтому такое преобразование имеет смысл использовать только тогда, когда вам нужно объединять данные из таблиц на разных источниках данных;
 - альтернативой этому преобразованию может стать преобразование **Lookup** (точно так же, как запрос с вложенным подзапросом может за-

менить соединения). При использовании преобразования **Merge Join** вы проигрываете в гибкости, но можете выиграть в производительности тогда, когда обе таблицы, информация из которых соединяется, отсортированы по ключу, по которому производится соединение.

В этом преобразовании вы можете использовать два источника данных. Если на одном источнике информация не отсортирована, то рекомендуется перед этим преобразованием настроить преобразование типа **Sort**.

Соединения могут быть такими же, как и в запросах Transact-SQL: правые, левые и внутренние.

□ **Multicast** (Множественная передача) — это преобразование просто распараллеливает поток входящих данных. В нем используется один источник и множество назначений. Во все назначения будет записана одинаковая информация. Можно использовать такое преобразование в качестве элементарной системы репликации, а можно организовывать одинаковые потоки данных из одного источника, чтобы передать их другим преобразованиям и затем опять слить вместе.

□ **OLE DB Command** (Команда OLE DB) — это преобразование позволяет использовать команду SQL для каждой записи, которая передается с источника. При этом в команде SQL информацию из текущей записи можно использовать в качестве значений параметров. Например, преобразование OLE DB Command можно использовать в ситуации, когда каждой записи на источнике соответствует, например, информация о заказе, которую нужно удалить из другой таблицы. В этом случае при помощи этого преобразования можно определять команду `DELETE` и использовать для нее в качестве параметров значения, поступающие с источника.

Это преобразование использует один источник (только типа OLE DB Source) и одно назначение (только типа OLE DB Destination).

□ **Percentage Sampling** (Процентная выборка) — это очень интересное преобразование. Оно принимает данные из одного источника и передает их в два назначения. В одно назначение, которое называется *невыбранный вывод выборки* (*sampling unselected output*), будут передаваться данные в том же виде, в котором они поступили с источника, а в другое, которое называется *выбранный вывод выборки* (*sampling selected output*), поступит случайная выборка значений с источника. При этом вы можете определить процент значений, который попадет в выборку. Но следует учесть, что большой точностью в подсчете процентов это преобразование не отличается — например, если попросить его взять 10% от 290 записей, то получится 35 записей.

Если вас интересует только выборка, то вы можете вообще не определять для этого преобразования *невыбранный вывод выборки*.

Основное назначение этого преобразования — тестирование пакетов, когда тестировать их на всем массиве записей получается очень долго.

- **Pivot** (Смена столбцов и строк) — очень специфическое преобразование, основное назначение которого — денормализация. При использовании этого преобразования увеличивается количество столбцов, зато снижается количество записей. Смысл этого преобразования проще всего показать на простом примере. Предположим, что у вас есть таблица с информацией о заказах (табл. 10.1).

Таблица 10.1. Исходная таблица с информацией о заказах

Заказчик	Продукт	Количество
Заказчик 1	Продукт 1	5
Заказчик 1	Продукт 2	10
Заказчик 1	Продукт 3	8
Заказчик 2	Продукт 2	10
Заказчик 2	Продукт 4	5
Заказчик 2	Продукт 3	6

После применения этого преобразования те же данные могут выглядеть, например, так, как приведено в табл. 10.2.

Таблица 10.2. Таблица после применения преобразования Pivot

Заказчик	Продукт 1	Продукт 2	Продукт 3	Продукт 4
Заказчик 1	5	10	8	
Заказчик 2		10	6	5

Настройка этого преобразования не очевидна. Вначале на вкладке **Input Columns** (Столбцы ввода) свойств преобразования вам потребуется выбрать нужные столбцы источника, а затем перейти на вкладку **Input and Output Properties** (Свойства ввода и вывода) и для столбцов источника настроить значение свойства **PivotUsage**. Для этого свойства используется одно из четырех значений, которое определяет, какая роль в преобразовании предназначается данному столбцу. Затем на вкладке **Output Columns** (Столбцы вывода) нужно создать столбцы, которые будут передаваться назначению, и привязать их к столбцам источника при помощи свойства **PivotKeyValue**.

Существует также преобразование, которое выполняет обратную операцию и производит нормализацию таблицы. Оно называется **Unpivot** (Обратная смена).

- **Row Count** (Счетчик строк) — это преобразование просто считает количество строк источника, которое прошло через него, и сохраняет это значение в переменной пакета (перед настройкой этого преобразования переменная должна быть уже создана). Это преобразование обычно применяется при создании пользовательского отчета по результатам работы пакета.
- **Row Sampling** (Выборка строк) — это преобразование очень похоже на **Percentage Sampling**. Оно также случайным образом выбирает из всех записей источника некоторое их количество. Однако в отличие от **Percentage Sampling**, в котором выбирался примерный процент для записей, попадающих в выборку, здесь указывается не процент, а точное количество. Поэтому Microsoft рекомендует использовать это преобразование, например, для лотерей. Если вам нужно явно выбрать какие-то значения вместо случайных (например, для тестирования, или вы хотите повлиять на результаты лотереи), то в вашем распоряжении — параметр **Seed**.

Точно так же, как и **Percentage Sampling**, источник у этого преобразования может быть только один, а вот назначений — два (для полного набора записей и для выборки).

- **Script Component** (Компонент скрипта) — это, безусловно, самое мощное преобразование. Его можно использовать не только как преобразование, но и как источник или назначение данных. Фактически этим преобразованием можно заменить все остальные преобразования. Если вам не хватает функциональности других преобразований, то опять-таки можно использовать преобразование **Script Component**. В этом преобразовании вы можете использовать свой собственный программный код для выполнения каких-то операций с данными, которые передаются с источника. Несмотря на слово **Script** в названии преобразования, создавать программный код можно только на Visual Basic .NET. Но при этом вам будут доступны все функции Visual Basic .NET, все сборки и пространства имен .NET, а при помощи стандартных средств Visual Studio и обычные СОМ-компоненты.

Расплата за функциональность — резкое падение производительности: программный код выполняется для каждой записи, которая приходит с источника.

Настройка этого преобразования не вполне очевидна. При размещении его в пакете нужно выбрать: будет ли **Script Component** выступать в роли источника, назначения или преобразования. Если вы выбрали для него роль

преобразования, то его нужно разместить, как и другие преобразования, между источником и назначением (рис. 10.7).



Рис. 10.7. Настройка преобразования **Script Component**

Предположим, что вы разместили этот компонент в вашем экспериментальном пакете, который занимается передачей данных из SQL Server в файл Excel, как показано на рисунке, и выбрали для него роль преобразования (т. е. в окне **Select Script Component Type** (Выбрать тип компонента скрипта), которое открывается при перетаскивании этого объекта из **Toolbox**, установили переключатель в положение **Transformation** (Преобразование)). Для демонстрации произведем очень простую операцию: выведем в окно сообщения информацию о каждом сотруднике (его имени и фамилии). Для этого вам нужно выполнить следующие действия:

1. Открыть свойства преобразования **Script Component** (это можно сделать, щелкнув по нему правой кнопкой мыши и в контекстном меню выбрав команду **Edit**).
2. На вкладке **Input Columns** свойств преобразования нужно установить флажки напротив столбцов **FirstName** и **LastName**.
3. На вкладке **Inputs and Outputs** (Вводы и выводы) для вас уже будет заготовлены один источник и одно назначение. При желании можно добавить дополнительные назначения (т. е. для этого преобразования можно будет использовать не одно, а несколько исходящих зеленых стрелок). В данном случае оставьте на этой вкладке значения по умолчанию.
4. Основная работа производится в окне редактора кода. Чтобы его открыть, нужно перейти на вкладку **Script** (Скрипт) и нажать кнопку **Design Script** (Спроектировать скрипт). Откроется окно Visual Studio for Application.
5. В этом окне вам будет предложено ввести свой код для класса **ScriptMain** (точнее, для событийной процедуры **Input0_ProcessInputRow()**, определенной в этом классе). Вводить код рекомендуется на месте комментария с пометкой "Add Your Code Here" (Добавьте сюда ваш код). Вариант этой процедуры может быть таким:

```

Public Overrides Sub Input0_ProcessInputRow(ByVal Row As Input0Buffer)
    MsgBox(Row.FirstName.ToString & " " & Row.LastName.ToString)
End Sub
  
```

Далее вам осталось только закрыть все окна с сохранением внесенных изменений и запустить пакет на выполнение. Если вам не хочется закрывать окна сообщений 290 раз, то можно воспользоваться командой **Debug | Stop debugging** (Отладка | Остановить отладку) из меню SSIS Designer.

Для того чтобы использовать более функциональные варианты этого преобразования, необходимо знакомство с объектной моделью пакетов SSIS. К сожалению, справки по ней в документации по SQL Server 2005 нет. Вам придется обращаться к справке в MSDN (поставляемой с Visual Studio 2005 или на сайте www.microsoft.com/msdn).

- **Slowly Changing Dimension Transformation** (Преобразование медленно изменяющегося измерения) — это экзотическое преобразование предназначено для внесения изменений в измерения кубов Analysis Services на основе информации из реляционного источника данных.
- **Sort** (Сортировка) — это очень простое и часто используемое преобразование предназначено для сортировки записей, которые поступают с источника. В его свойствах вы можете выбрать один или несколько столбцов, настроить для них очередность и порядок сортировки (по возрастанию или убыванию). Это преобразование может также удалять повторяющиеся записи.

Вся сортировка производится в оперативной памяти того компьютера, на котором работает пакет, поэтому с этим преобразованием нужно быть осторожным. По возможности сортировку лучше выполнять на сервере баз данных, с которого поступают записи, указав в качестве источника запрос с оператором `ORDER BY`. Это преобразование лучше всего использовать только в тех ситуациях, когда источник не поддерживает сортировку (например, когда в качестве источника выступают текстовые файлы).

- **Term Extraction** (Извлечение термина) и **Term Lookup** (Извлечение термина из внешнего источника) — это сложные преобразования, использующие элементы полнотекстового поиска. Они могут работать только для источников на английском языке. **Term Extraction** извлекает из источника существительные и словосочетания с существительными (в зависимости от выбранного режима) и записывает их в назначение. **Term Lookup** позволяет также обратиться к еще одному источнику (он называется *справочной таблицей (reference table)*), и определить, сколько найдено между двумя источниками совпадений. Затем сумма совпадений вместе с извлеченными существительными и словосочетаниями записывается в назначение.

10.6.5. Пути и логика выполнения Data Flow Task

Последний элемент Data Flow Task — это пути, т. е. зеленые и красные стрелки, которые соединяют назначения и преобразования. Их применение

достаточно очевидно, отметим лишь некоторые моменты, которые могут вызвать затруднения.

Зеленые стрелки означают нормальное прохождение данных, а красные — путь для передачи данных, при обработке которых возникли ошибки. Красные стрелки предусмотрены далеко не для всех преобразований. Кроме того, если сбор информации о проблемных записях вам не нужен, вы можете просто не использовать красную стрелку.

Если в Data Flow Task есть несколько не связанных между собой маршрутов данных (например, один маршрут — для данных, передаваемых с SQL Server в Excel, а второй — для данных, передаваемых с FoxPro на Oracle), то передача данных по обоим маршрутам начнется одновременно и будет производиться параллельно.

Для многих преобразований можно использовать несколько вводов (*Input*) и несколько выводов (*Output*). Определить несколько вводов можно очень просто: достаточно перетащить на это преобразование зеленые стрелки-пути с других источников. А для того чтобы определить нужный вывод, достаточно перетащить на назначение зеленую стрелку. Откроется окно, в котором вы можете выбрать конкретный вывод, обозначаемый этой стрелкой.

Если открыть свойства пути (команда **Edit** в контекстном меню), то можно настроить имя пути, описание и отображаемую надпись для пути (**PathAnnotation**). К сожалению, вы не можете напрямую ввести свое значение для свойства **PathAnnotation**, зато можно для него выбрать значение **PathName** и ввести свое название для пути. В результате для пути на графическом экране SSIS Designer будет отображаться введенное вами название. Такое решение может быть очень удобным для запутанных пакетов с большим количеством элементов и связей между ними. Кроме того, вы можете добавить в любое место пакета поясняющую текстовую надпись. Для этого достаточно щелкнуть правой кнопкой мыши по пустому месту в пакете и в контекстном меню выбрать **Add Annotation** (Добавить аннотацию).

В SSIS Designer предусмотрена еще одна очень удобная возможность. К любому пути можно присоединить просмотрщик данных **Data Viewer**. Это графический интерфейс, который во время работы пакета будет показывать пересылаемые данные (рис. 10.8).

Конечно же, такое средство очень удобно использовать для отладки пакетов — вы сразу видите, как выглядят передаваемые данные.

Чтобы настроить просмотрщик, достаточно щелкнуть правой кнопкой мыши по пути и в контекстном меню выбрать **Data Viewers** (Просмотрщики данных), а затем нажать кнопку **Add** (Добавить). Вам будет предложено выбрать один из четырех типов просмотрщика (самый удобный из них — **Grid** (Сетка) — выбирается по умолчанию). На второй вкладке вы можете настроить

свойства просмотрщика, например, выбрать отображаемые столбцы. После настройки просмотрщика рядом с путем в SSIS Designer появится специальная иконка.

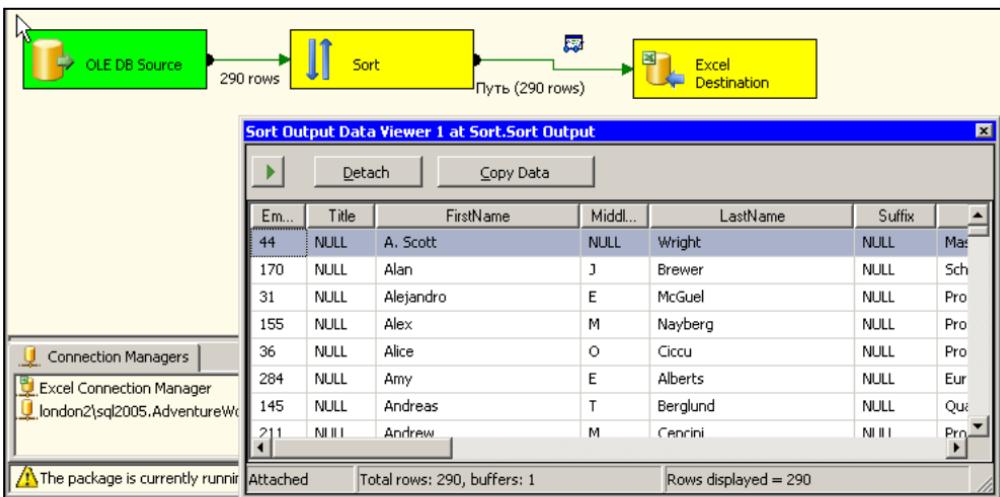


Рис. 10.8. Применение просмотрщика данных

Как только первая группа записей поступит в просмотрщик, выполнение пакета приостановится. Вы можете продолжить выполнение при помощи кнопки с зеленой стрелкой в верхнем левом углу просмотрщика или просто закрыв его.

10.7. Script Task и ActiveX Script Task

В предыдущем разделе была подробно разобрана главная задача Integration Services — Data Flow Task. Однако в SSIS предусмотрено множество других задач, которые могут оказаться очень полезными. Работа с ними выглядит одинаково: вы выбираете задачу в **Toolbox**, перетаскиваете ее на вкладку **Control Flow** и настраиваете свойства. Если вам необходимо обеспечить последовательное выполнение задач, то нужно связать их между собой стрелками (внешне они очень похожи на пути Data Flow Task), которые называются *ограничениями предшественников* (*precedence constraints*). Подробнее про работу с ними будет рассказано в разд. 10.23.

Отметим, что список задач, которые находятся в **Toolbox** на вкладке **Control Flow**, не является исчерпывающим. Вы можете написать свой программный модуль и зарегистрировать его в качестве задачи SSIS. В этом случае он станет доступен как пользовательская задача SSIS. Документация по созданию своих задач находится в MSDN.

Первые дополнительные задачи, которые рассмотрим подробно, — *Script Task* и *ActiveX Script Task*.

Обе эти задачи предназначены для одной и той же цели — выполнение в пакете своего программного кода. Их функциональность очень велика. С их помощью вы можете использовать все объектные модели (или классы пространств имен .NET), которые есть на вашем компьютере. Например, вам необходимо выполнить какие-то операции в файловой системе, или обратиться на контроллер домена, или вызвать на выполнение внешнюю утилиту, или изменить параметры самого пакета SSIS в ходе выполнения (например, присвоить нужные значения переменным) — во всех таких случаях вам пригодятся эти задачи.

Задача *Script Task* использует Visual Studio for Applications и язык Visual Basic.NET. Работа с ней очень похожа на работу с преобразованием **Script Component** в Data Flow Task: вы перетаскиваете эту задачу на вкладку **Control Flow**, открываете ее свойства и на вкладке **Script** нажимаете кнопку **Design Script** (Спроектировать скрипт). Конечно, для комфортной работы с этой задачей необходимо знать объектную модель пакетов SSIS (справка по ней находится в MSDN).

Для программной работы с обычным, не .NET-совместимым программным кодом предназначена задача, которая называется ActiveX Script Task. В ней вы можете использовать код скрипта, который будет выполнен в ходе работы пакета. По умолчанию можно использовать только код на языках VBScript и JavaScript (как обычный, так и защищенный цифровой подписью). Если на компьютере установлен интерпретатор для другого скриптового языка (например, для ActivePerl), то можно использовать его.

Задача ActiveX Script Task используется для тех же целей, что и *Script Task*. При этом при прочих равных условиях *Script Task* обычно работает быстрее, поскольку при сохранении пакета программный код Visual Basic.NET в *Script Task* компилируется и хранится в откомпилированном виде.

Как и в других ситуациях (например, с объектными моделями SMO и SQL-DMO), очевидно, что Microsoft отдает предпочтение .NET-совместимому программному коду перед обычным. Складывается даже ощущение, что удобство работы с ActiveX Script Task специально минимизировано. Фактически вы можете только скопировать в эту задачу текст программного кода. В отличие от ActiveX Script Task в SQL Server 2000 в ActiveX Script Task уже нет списка функций, подсказок по компонентам пакета и т. п. Более того, Microsoft предупреждает, что в следующих версиях SQL Server задачи типа ActiveX Script Task поддерживаться не будут, и не рекомендует использовать их в новых проектах.

10.8. Bulk Insert Task

Bulk Insert Task — это задача, единственное назначение которой заключается в загрузке информации из текстовых файлов с разделителями (*flat files*) на SQL Server. При этом задача Bulk Insert Task не может производить ни преобразований, ни проверок данных.

Может возникнуть вопрос: а зачем нужна такая задача? Ведь то же самое можно сделать при помощи обычной задачи Data Flow Task и с намного большими функциональными возможностями. Ответ прост: главное преимущество Bulk Insert Task — производительность. В этой задаче используются специализированные программные объекты, оптимизированные для достижения максимальной скорости загрузки данных. Фактически эта задача — оболочка SSIS для утилиты командной строки `bcp` (*bulk copy program*), поскольку в ней используются те же объекты. Однако отметим, что `bcp` умеет как загружать данные на SQL Server, так и выгружать их с SQL Server в файл на диске, а Bulk Insert Task работает только в одном направлении и может только загружать данные на SQL Server.

Отметим также, что Bulk Insert Task — достаточно капризная в практическом использовании задача. По опыту работы с ней можно сказать, что при загрузке данных средствами этой задачи периодически возникают ошибки, связанные, например, с тем, что в текстовом файле встретился какой-то нестандартный символ. Поэтому такую задачу имеет смысл использовать только при очень серьезных требованиях к производительности (когда вам нужно, например, загружать миллионы записей в течение короткого времени). Во всех остальных ситуациях предпочтительнее использовать Data Flow Task.

Настройка Bulk Insert Task очень проста. Она требует наличия двух менеджеров подключений (см. разд. 10.5) — **OLE DB Connection Manager** (для SQL Server, на который будут загружаться данные) и **Flat File Connection Manager** (для текстового файла-источника). Кроме того, в свойствах Bulk Insert Task вы можете выбрать таблицу, в которую будет производиться загрузка, кодировку текстового файла-источника, разделители строк и столбцов. Дополнительные параметры загрузки данных (например, назначение столбца в исходном файле столбцу таблицы, в которую производится запись, или игнорирование некоторых столбцов файла-источника) можно определить при помощи файла форматирования (он выбирается при помощи параметра **Format** на вкладке **Connections** (Соединения) свойств Bulk Insert Task). Формат этого файла описан в документации к утилите `bcp` (он одинаков для Bulk Insert Task и для этой утилиты), а сгенерировать его в автоматическом режиме проще всего при помощи Bulk Insert Task средствами DTS Designer в SQL Server 2000.

10.9. Execute SQL Task

Execute SQL Task — это простая и очень удобная во многих ситуациях задача SSIS. Ее основное назначение заключается в выполнении команды SQL или хранимой процедуры на сервере баз данных (с возможной передачей параметров). Эта задача может использоваться для выполнения команд SQL как на SQL Server, так и на других источниках данных (например, Oracle, Access и т. п.). Единственное ограничение — источник данных должен поддерживать интерфейс `ICommand`, т. е. уметь работать с командами SQL. Поэтому, например, использовать эту задачу для обращения к текстовым файлам не получится.

Чаще всего эта задача используется в следующих ситуациях:

- для создания или удаления объектов на SQL Server;
- для очистки промежуточных таблиц, используемых при закачке данных для выполнения преобразований с ними;
- для запуска хранимых процедур;
- для выполнения параметризованных запросов (в качестве параметра можно использовать глобальные переменные, а их, в свою очередь, можно назначить при запуске пакета из командной строки или на графическом экране);
- для сохранения результатов выполнения запросов в глобальной переменной (одно значение или табличный набор записей).

Настройка этой задачи начинается с выбора подключения к источнику данных (свойства **Connection Type** (Тип подключения) и **Connection** (Соединение) на вкладке **General** (Общие)). Затем вы должны выбрать источник, из которого будет браться текст запроса. Для этого предназначено свойство **SQLSourceType** (Тип источника SQL) на той же вкладке. В вашем распоряжении есть три варианта:

- Direct Input** (Прямой ввод) — текст запроса вводится напрямую при помощи свойства **SQLStatement** (Выражение SQL) (или генерируется при помощи кнопки **Build Query**). В тех местах запроса, где должны подставляться параметры, устанавливаются знаки вопроса. Затем на вкладке **Parameter Mapping** (Привязка параметров) этим параметрам можно назначить переменные;
- Variable** (Переменная) — текст запроса будет определяться при помощи единственной строковой переменной (чаще всего значение этой переменной в ходе выполнения пакета конструируется при помощи строковых функций средствами задачи Script Task). Имя переменной определяется при помощи свойства **SourceVariable** (Переменная источника);

- **File connection** (Подключение к файлу) — текст запроса будет поступать из текстового файла на диске. В этом случае вам потребуется создать менеджер подключения типа **File Connection Manager**.

Если вы хотите использовать в вашем пакете результаты выполнения запроса Execute SQL Task, то следующим действием нужно определить формат для этих результатов. В вашем распоряжении значения **None** (означает, что результаты выполнения запроса нас не интересуют, используется по умолчанию), **Single row** (единственная строка или единственное значение), **Full result set** (полный набор записей табличного вида) или **XML**. Если вы выбрали значение **XML**, то возвращаемые данные будут зависеть от типа переменной, в которую будут приниматься результаты. Если выбрана переменная типа *String*, то в нее будет помещен код XML. Если же результат принимается в переменную типа *Object*, то в нее будет помещен объект документа XML объектной модели DOM. Формат документа XML (или структура объекта) будут зависеть от запроса, который был выполнен на источнике данных.

Затем вы должны на вкладке **Parameter Mapping** назначить переменные входящим параметрам вашего запроса, а на вкладке **Result Set** (Набор результатов) определить переменные для значений, которые будет возвращать ваш запрос. Создать, изменить или удалить переменные уровня пакета можно при помощи меню **SSIS | Variables** в SSIS Designer.

10.10. XML Task

XML Task — это задача, которая предназначена для выполнения различных операций с документами в формате XML. Обратите внимание, что эту задачу нельзя использовать для загрузки данных в формате XML на SQL Server или для выгрузки данных с SQL Server в формате XML (что требуется чаще всего). Для загрузки данных удобнее всего использовать Data Flow Task, определив в ней источник XML Source. Для выгрузки данных в формате XML с SQL Server рекомендуется также использовать Data Flow Task с назначением OLE DB Destination, для которого настроен **OLE DB Connection Manager** с поставщиками (драйверами) **SQLXMOLEDB** или **SQLXMOLEDB4.0**.

Задача XML Task обычно используется для подготовки документа XML. В ней всегда используются два источника: **Source** (Источник) на вкладке **General** свойств этой задачи и **Second Operand** (Второй operand) на той же вкладке. Например, при проверке на соответствие структуре (*validation*) первым источником будет проверяемый документ XML, а вторым — файл схемы, которому он должен соответствовать, при выполнении преобразования XSLT вторым документом будет код преобразования и т. п. И первый, и второй документ можно ввести напрямую прямо в свойствах этой задачи (значе-

ние **Direct Input** свойств **SourceType** (Тип источника) и **SecondOperandType** (Тип второго операнда)), получить из файла (значение **File Connection**) или принять из переменной пакета (значение **Variable**). Результаты операции можно сохранить в файл или записать в переменную.

Всего для этой задачи предусмотрено шесть операций. Нужная операция выбирается при помощи свойства **OperationType** (Тип операции) на вкладке **General**:

- Validate** — проверка документа XML на соответствие схеме или DTD;
- XSLT** — проведение преобразования XSL Transformation. Обычно в результате такого преобразования создается документ XML с другим форматом или документ HTML;
- XPATH** — выполнение запроса на языке XPath. Чаще всего такой запрос используется для того, чтобы найти нужный элемент или элементы в документе XML;
- Merge** — слияние двух документов XML в один (например, отчеты за разные периоды или из разных филиалов);
- Diff** — сравнение двух документов XML и генерирование отчета о найденных отличиях в стандартном XML-совместимом формате DiffGram;
- Patch** — использование результатов операции Diff (т. е. данные в формате DiffGram) для создания нового документа XML.

10.11. Message Queue Task

Message Queue Task — это задача, предназначенная для работы с сообщениями в Message Queue Service (служба работы с сообщениями, которая поставляется вместе с Windows 2000 Server и Windows Server 2003). По наблюдениям автора, начинающие разработчики обычно не обращают на эту задачу особого внимания. Зато у опытных разработчиков она относится к числу любимых.

Самые распространенные ситуации, когда обычно используется эта задача, представлены далее:

- вам необходимо повысить надежность передачи данных при не очень надежных или перегруженных сетевых соединениях. Если сетевое соединение, например, с удаленным филиалом недоступно или перегружено, то при передаче данных обычным способом возникнет ошибка. Если же для этой цели вы используете средства работы с очередями, то передаваемые данные будут помещены в очередь и будут доставлены, как только соединение станет доступным;

- для синхронизации работы пакетов SSIS (и просто приложений). Предположим, например, что в течение ночи у вас проводится резервное копирование баз данных, загрузка новых данных, их проверка, процессинг кубов OLAP, в которых используются эти данные, и т. п. При этом время, которое потребуется на выполнение каждой из этих задач, может изменяться. Можно постараться реализовать последовательное выполнение всех этих задач в рамках одного пакета SSIS (или одного задания SQL Server Agent), но это не всегда удобно. Намного удобнее использовать в качестве флага, разрешающего выполнение какой-то операции, сообщение в очереди сообщений Message Queue. Точно так же пакеты или приложения могут обмениваться между собой данными;
- для накопления и последующей пакетной обработки данных. Например, данные, поступающие за день из филиалов, копятся в очередь сообщений, чтобы не перегружать загруженный OLTP-сервер. Ночью, когда нагрузка на OLTP-сервер падает, они из очереди передаются на сервер.

Такие задачи возникают на многих предприятиях, а решить их без помощи Message Queue Task достаточно сложно.

Первое, что нужно сделать при настройке Message Queue Task, — установить службы Message Queue и создать очередь сообщений. Совершенно необязательно устанавливать их на том же компьютере, на котором будет работать пакет SSIS. Наоборот, чаще для всего предприятия используется один компьютер с установленной службой Message Queue. Главные требования — на этом компьютере должна работать серверная версия Windows (2000 или 2003), и этот компьютер должен быть доступен постоянно. Можно, конечно, использовать службы Message Queue и на компьютере, на котором установлен SQL Server 2005 и работают пакеты SSIS, но обычно одного компьютера с установленной службой Message Queue вполне достаточно.

Установка служб Message Queue производится обычными средствами Windows. Нужно открыть **Панель управления**, запустить консоль **Установка и удаление программ** и воспользоваться кнопкой **Установка компонентов Windows**. Затем нужно раскрыть контейнер **Сервер приложений** и установить флажок напротив элемента **Очередь сообщений**. Двух компонентов этого элемента, который устанавливаются по умолчанию (**Общие и Интеграция с Active Directory**) в большинстве случаев вполне достаточно.

Следующая задача, которую нужно сделать после установки служб Message Queue, — создать очередь, которую вы будете использовать. Для этого нужно открыть консоль **Управление компьютером** в меню **Пуск | Программы | Администрирование** и раскрыть в ней контейнер **Службы и приложения | Очередь сообщений**. Обычно для пакетов SSIS используются частные очереди, поэтому лучше всего создать новую частную очередь из контекстного

меню для одноименного контейнера. Называться она будет для целей нашего примера SSIS1 (рис. 10.9).

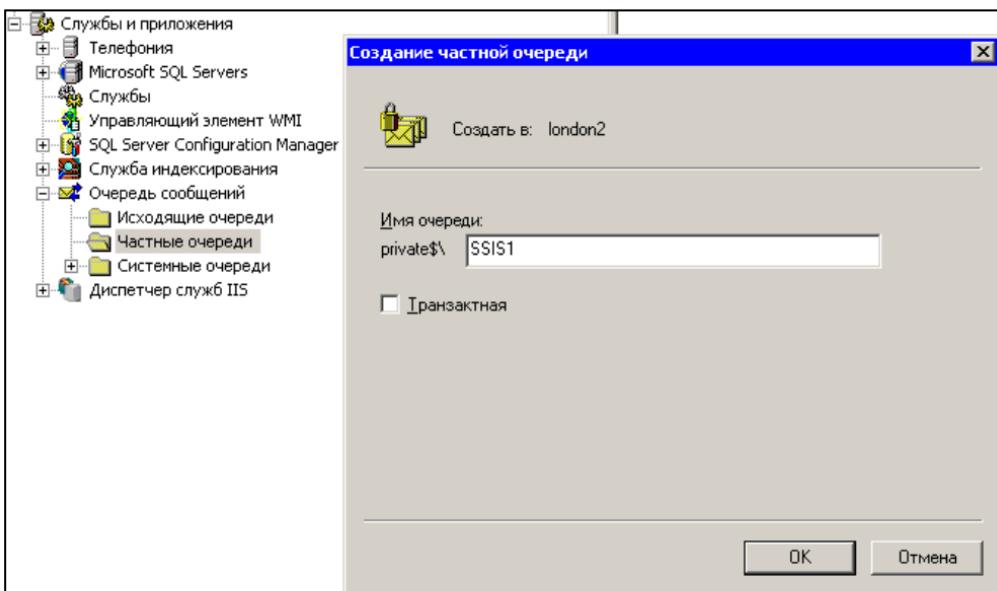


Рис. 10.9. Создание частной очереди

После нажатия на кнопку **OK** частная очередь готова к использованию в пакете.

Следующее, что вам потребуется сделать, — создать в пакете менеджер подключения типа **MSMQ Connection Manager**. Его главное свойство — **Path** (Путь), в котором определяется путь и имя очереди. Значение для этого свойства записывается в формате *имя_компьютера\Private\$\имя_очереди*, например *LONDON2\Private\$\SSIS1*.

Следующая ваша задача — перетащить из **Toolbox** на вкладку **Control Flow** задачу типа **Message Queue Task** и настроить ее свойства. На вкладке **General** свойств **Message Queue Task** можно определить имя и описание задачи, будет ли использоваться формат, совместимый с очередями сообщений в Windows 2000 (по умолчанию используется формат, который совместим с Windows Server 2003), указать менеджер подключения, а также тип задачи с помощью свойства **Message** (Сообщение) (будет ли эта задача отправлять сообщение в очередь (**Send message**) или получать сообщение из очереди (**Receive Message**)). В зависимости от выбранного вами значения вторая вкладка свойств задачи будет называться, соответственно, **Send** (Отправить) или **Receive** (Получить).

На этой второй вкладке вы можете определить параметры отправляемого или принимаемого сообщения. При помощи свойства **MessageType** (Тип сообщения) определяется тип сообщения. Сообщение может быть текстом, или переменной, или файлом данных. Для флага удобнее использовать строковые значения, а для передачи значений обычно используются файлы данных или переменные. В зависимости от выбранного типа можно определить дополнительные параметры сообщения.

После этого осталось включить задачу Message Queue в схему выполнения вашего пакета. Например, если пакет перед выполнением какой-то операции должен получить сигнал (или данные) от другого пакета, задачу Message Queue на получение сообщения можно поставить перед элементом, который выполняет требуемую операцию и настроить логику выполнения пакета при помощи ограничений предшественников (*precedence constraints*) (стрелок, связывающих задачи на вкладке **Control Flow**).

10.12. Execute Package Task и Execute DTS 2000 Package Task

Задачи Execute Package Task и Execute DTS 2000 Package Task относятся к числу самых простых. Единственное назначение этих задач — запускать на выполнение другие пакеты (SSIS и пакеты в формате DTS SQL Server 2000 соответственно). Обычно эти задачи используются для обеспечения модульности и исключения дублирования кода. Если вам нужно выполнять в каких-то пакетах похожий набор действий, то не имеет смысла повторять настройку элементов в каждом пакете. Удобнее поместить повторяющийся набор элементов во внешний пакет (он может находиться и на другом сервере SQL Server или, например, на сетевом ресурсе файл-сервера) и запускать этот внешний пакет при помощи задач Execute Package Task. При запуске такому вложенному пакету можно передавать параметры, которые он сможет использовать при выполнении.

Настройка задачи Execute Package Task очень проста. На вкладке **Package** (Пакет) свойств задачи вы можете выбрать местонахождение пакета (в файловой системе или на SQL Server), выбрать сам пакет (в виде менеджеров подключений **File Connection Manager** или **OLE DB Connection Manager** с указанием имени пакета), а также указать пароль на запуск пакета (если пакет зашифрован) и настроить режим его выполнения (свойство **ExecuteOutOfProcess**). Значение **FALSE** свойства **ExecuteOutOfProcess** (по умолчанию) означает, что вложенный пакет будет выполняться в рамках того же процесса, что и родительский, а значение **TRUE** — что для вложенного пакета будет запущен отдельный процесс.

Единственная проблема, которая может возникнуть при использовании этих задач, — как передавать параметры вложенному пакету.

С задачей Execute DTS 2000 Package Task, предназначеннной для запуска на выполнение старых пакетов DTS, все просто: в свойствах есть вкладки **Inner Variables** (Внутренние переменные) и **Outer Variables** (Внешние переменные). На вкладке **Inner Variables** назначаются значения переменным вложенного пакета, которые будут использованы при его запуске (обычно для этого используются переменные родительского пакета). На вкладке **Outer Variables** обеспечивается прием данных после завершения выполнения вложенного пакета (опять-таки при помощи переменных родительского пакета).

А настройка передачи параметров вложенному пакету в задаче Execute Package Task, предназначенной для запуска пакетов DTS, производится сложнее. Для этого нужно открыть вложенный пакет в SSIS Designer и в меню SSIS выбрать команду **Package Configurations** (Конфигурации пакетов). В открывшемся окне **Package Configurations Organizer** (Организатор конфигураций пакетов) нужно установить флажок **Enable Package Configurations** (Включить конфигурации пакетов) и нажать кнопку **Add**, чтобы создать новую конфигурацию. Запустится мастер конфигураций пакета **Package Configuration Wizard**. На экране **Select Configuration Type** (Выберите тип конфигурации) нужно выбрать тип конфигурации **Parent package variable** (Переменная родительского пакета), ввести имя переменной родительского пакета, а на следующем экране нужно выбрать свойство элемента вложенного пакета (или его переменную), для которого будет использоваться значение переменной родительского пакета. Затем эту же операцию по созданию конфигурации можно повторить еще несколько раз. После этого при запуске вложенного пакета свойства этого пакета будут динамически меняться в зависимости от значений переменных в родительском пакете.

10.13. Transfer Database Task

Это очень простая задача. Она предназначена для копирования или перемещения баз данных с одного сервера SQL Server на другой. Отметим некоторые моменты, которые могут оказаться неочевидными:

- эта задача может работать в двух режимах (они выбираются при помощи свойства **Method** (Метод) на вкладке **Databases** (Базы данных) свойств задачи). Если вы установите для него значение **DatabaseOnline**, то на время работы база данных-источник этой задачи останется доступной для пользователей. Вся работа по перемещению баз данных и их объектов между серверами будет производиться средствами объектной модели SMO. Если же вы выберете автономный режим (значение **DatabaseOffline**, по умол-

чанию), то операция по копированию базы данных будет выполнена по-другому. Исходная база данных будет отсоединенна, ее файлы будут скопированы по сети на сервер назначения и подключены к нему. Будет ли база данных на сервере-источнике заново присоединена после этого, определяется при помощи свойства **ReattachSourceDatabase**:

- эта задача умеет работать не только с экземплярами SQL Server 2005, но и с экземплярами SQL Server 2000. Единственное ограничение — нельзя копировать или перемещать базу данных SQL Server 2005 на SQL Server 2000. Копирование же между экземплярами SQL Server 2000 или с SQL Server 2000 на SQL Server 2005 вполне возможно;
- в качестве источника и назначения можно указывать один и тот же сервер. В этом случае база данных будет скопирована на тот же сервер под другим именем;
- выбирать, какие объекты базы данных будут скопированы или перенесены, нельзя. При помощи этой задачи базу данных можно копировать только целиком.

10.14. Другие задачи копирования объектов SQL Server

К другим задачам копирования объектов SQL Server относятся:

- **Transfer Error Messages Task** — задача для копирования определений пользовательских ошибок между серверами SQL Server 2005;
- **Transfer Jobs Task** — копирование заданий SQL Server Agent;
- **Transfer Logins Task** — копирование логинов SQL Server 2005;
- **Transfer Master Stored Procedures Task** — копирование хранимых процедур, определенных в базе данных `master`;
- **Transfer SQL Server Objects Task** — копирование объектов в базах данных (таблиц, представлений, хранимых процедур, ограничений целостности и т. п.). Можно настроить также копирование логинов SQL Server вместе с другими объектами. Вы можете определить, будут ли таблицы копироваться с данными или без, будут ли копироваться разрешения и т. п.

Чаще всего эти задачи (кроме Transfer SQL Server Objects Task) используются на регулярной основе для дополнения репликации, чтобы кроме реплицируемых данных передавались и объекты SQL Server. Но их можно использовать и для других целей, например, для создания копии вашего рабочего сервера SQL Server для проведения экспериментов.

Все эти задачи могут только копировать объекты SQL Server (но не переносить их). В качестве источника можно использовать и SQL Server 2005, и SQL Server 2000, но для SQL Server 2000 будут доступны не все объекты (например, не будет возможности копировать сборки .NET).

10.15. File System Task и FTP Task

Эти две задачи, как понятно из названия, предназначены для операций с файлами. Задача File System Task выполняет операции обычными средствами работы с локальными и сетевыми каталогами, а FTP Task делает то же средствами протокола FTP. Перечислим некоторые возможности этих двух задач:

- вы можете копировать или перемещать файлы между каталогами. Для тех, кто работал с FTP Task в DTS 2000, отметим, что теперь можно не только получать файлы с FTP-сервера, но и загружать их на сервер;
- вы можете создавать, удалять и перемещать каталоги и файлы;
- File System Task умеет также копировать папки со всем содержимым, очищать папки и изменять атрибуты файлов.

Для каждой из этих задач можно указать весь набор файлов в определенном каталоге (выглядеть соответствующее значение может, например, как "D:\Upload*.csv"). Эти задачи очень удобно использовать, например, для обработки (загрузки) файлов отчетов, которые филиалы передают в определенный каталог.

10.16. Send Mail Task

Эта задача предназначена для отправки сообщений по электронной почте в ходе выполнения пакета. Она требует обязательного наличия менеджера подключений SMTP Connection Manager. В отличие от задачи Send Mail Task в DTS 2000, которая умела работать только по протоколу MAPI, эта задача может работать только по протоколу SMTP.

Чаще всего Send Mail Task используется, конечно, для уведомления администратора о результатах работы пакета.

Настройка этой задачи очень проста. На вкладке **Mail** (Почта) ее свойств вы определяете SMTP Connection Manager и все обычные свойства электронного сообщения: **From** (От кого), **To** (Кому), **Cc** (Копия), **Bcc** (Невидимая копия), **Subject** (Тема). При помощи свойства **MessageSourceType** (Тип источника сообщения) вы можете определить, как именно будет формироваться текст письма. Его можно ввести прямо в окне свойств (**Direct Input**), взять из переменной (**Variable**) или из файла на диске (**File Connection**). Если вы хотите

использовать вложения, то вам потребуется менеджер подключения **File Connection Manager**.

Единственное свойство этой задачи, которое можно определять динамически при помощи переменной, — **MessageSource** (Текст сообщения). Для того чтобы в ходе выполнения пакета менять значения остальных свойств этой задачи, вам потребуется использовать или объектную модель пакетов SSIS средствами Script Task, или использовать конфигурации пакета (меню **SSIS | Package Configurations**).

10.17. Execute Process Task

Задача Execute Process Task предназначена для запуска в ходе выполнения пакетов внешних приложений. Например, если из филиалов пришли текстовые файлы в виде архивов ZIP, то перед их загрузкой в базу данных может потребоваться их разархивировать.

Сама эта задача очень проста. При помощи свойства **Executable** (Исполняемый) указывается исполняемый файл, который будет запускаться на выполнение. Этот файл может представлять приложение с графическим интерфейсом, например, Word или Excel, но обычно для этой задачи используются консольные приложения или пакетные файлы. При запуске приложению можно передать набор параметров при помощи свойства **Arguments** (Параметры) (если параметров несколько, они должны быть разделены пробелами). Можно также передать из переменной пакета значения в стандартную консоль ввода приложения, для этого предназначено свойство **StandardInputVariable** (Переменная стандартного ввода). То, что приложение возвращает в стандартную консоль вывода, можно принять в переменную пакета, определив ее при помощи свойства **StandardOutputVariable** (Переменная стандартного вывода). Затем можно разобрать возвращаемые значения, например, при помощи строковых функций в Script Task. При помощи свойства **StandardErrorVariable** (Переменная стандартной ошибки) можно также определить специальную переменную, в которую будут помещаться сообщения, возвращаемые внешней программой, в случае, если произошла ошибка.

10.18. Web Service Task

Задача типа Web Services Task предназначена для обращения из пакета к Web-службам. Web-служба — это один из самых современных способов организации взаимодействия между процессами. Смысл технологий взаимодей-

ствия между процессами (IPC, InterProcess Communications) заключается в том, что один процесс (например, работающий пакет SSIS) вызывает метод другого процесса и передает ему параметры, а затем принимает от него то, что возвращает этот метод.

У Web-служб есть множество преимуществ по сравнению с другими распределенными методами организации взаимодействия между процессами (такими, как DCOM, RPC, CORBA и т. п.):

- передача данных между процессами производится по стандартному протоколу HTTP. В результате упрощается настройка брандмауэров и появляется возможность надежно защищать передаваемые данные стандартными средствами SSL;
- сами данные передаются в XML-совместимых форматах (чаще всего в формате протокола SOAP). Формат XML — это универсальный формат, для работы с которым предусмотрено множество стандартных средств;
- Web-службы позволяют удобным и стандартным образом обеспечивать взаимодействие между программным кодом, который физически работает на разных операционных системах и платформах. Например, из пакета SSIS, который требует наличия Windows и среды выполнения .NET, вы вполне можете обратиться к Web-службе, которая реализована на Java и работает на сервере под управлением UNIX.

Web-службы очень удобно использовать для предоставления универсальной справочной информации, которая нужна многим приложениям (например, о курсах валют). В качестве примера глобальной Web-службы можно привести, например, Microsoft Passport. Наиболее рекомендованное средство для создания Web-служб на платформе Windows — это ASP.NET в Visual Studio .NET, однако вы можете создать Web-службу и стандартными средствами SQL Server 2005 при помощи объектов HTTP Endpoint.

Web Service Task — очень простая в настройке задача (хотя программный код Web-службы, к которой производится обращение, может быть очень сложным). Эта задача обязательно требует наличия менеджера подключения **HTTP Connection Manager**. В ее свойствах вы определяете имя этого менеджера подключения, имя файла WSDL (Web Service Definition Language — язык описания Web-служб), в котором описывается Web-служба (в частности, содержится информация о ее методах, принимаемых ими параметрах и возвращаемых значениях), а затем определяете вызываемый метод и переменные для параметров, которые будут передаваться этому методу. Принимать возвращаемые Web-службой данные можно как в переменные пакета, так и в текстовый файл (в этом случае потребуется менеджер подключения **File Connection Manager**).

10.19. WMI Data Reader Task и WMI Event Watcher Task

Задачи WMI Data Reader Task и WMI Event Watcher Task предназначены для использования в пакетах SSIS возможностей объектной модели WMI. Подробно про эту объектную модель рассказывалось в разд. 9.4. Обе этих задачи требуют наличия в пакете **WMI Connection Manager**.

WMI Data Reader Task предназначена для выполнения в ходе работы пакета запроса к объектам WMI на локальном или удаленном компьютере. Затем результаты выполнения этого запроса можно использовать для других задач пакета. Например, вы можете выяснить, запущен ли на компьютере тот или иной процесс, есть ли определенный файл или каталог на диске, что в настоящее время показывает определенный счетчик производительности, сколько осталось места на диске и т. п. В самой объектной модели WMI сотни объектов, но задача WMI Data Reader Task относится к категории самых простых. Для нее необходимо указать используемый **WMI Connection Manager**, запрос на языке WQL (и откуда он будет браться — просто из свойств задачи, из файла на диске или из переменной), тип возвращаемых данных (табличный набор записей, пары имя/значение или просто скалярное значение), и где будут сохраняться результаты запроса (в файле на диске или в переменной пакета).

Задача WMI Event Watcher Task также предназначена для выполнения запросов WMI и для работы с полученными результатами. Однако эта задача работает со специальным типом запросов — с событийными запросами WMI (см. разд. 9.4.6). Ее можно использовать, например, когда нужно, чтобы выполнение пакета продолжилось с определенной точки после того, как на диске появится файл с определенным именем, или завершит работу какой-то процесс (который, например, разархивировал файл данных), и т. п. Если WMI Data Reader Task может просто проверять это же самое на определенный момент времени, то WMI Event Watcher Task опрашивает объекты WMI с указанным вами интервалом, и если интересующее вас событие произошло, генерирует событие пакета SSIS. Обработку такого события можно настроить на вкладке **Event Handlers** в SSIS Designer.

Поскольку применение WMI Event Watcher Task может быть очень удобным, а опыта работы с ним у большинства администраторов нет, проиллюстрируем работу с этой задачей на простом примере. Предположим, что вам нужно отследить, когда на вашем компьютере завершит работу процесс с именем WinRAR.exe, и сразу после этого нужно выполнить какие-то действия в пакете SSIS. Для простоты эти действия будет выполнять задача Script Task, которая просто выводит стандартное окно сообщения.

Решение задачи может выглядеть так:

- Создайте новый пустой пакет SSIS.
- Перетащите на вкладку **Control Flow** из **Toolbox** элемент **WMI Event Watcher Task** и откройте его свойства (команда **Edit** контекстного меню), а затем перейдите на вкладку **WMI Options** (Параметры WMI).
- В списке рядом со свойством **WmiConnection** (Соединение WMI) выберите пункт **New WMI Connection** (Новое соединение WMI) и настройте свойства создаваемого менеджера подключений **WMI Connection Manager**. Если вы будете производить мониторинг завершения процесса на локальном компьютере, параметры **WMI Connection Manager** могут выглядеть так, как представлено на рис. 10.10.

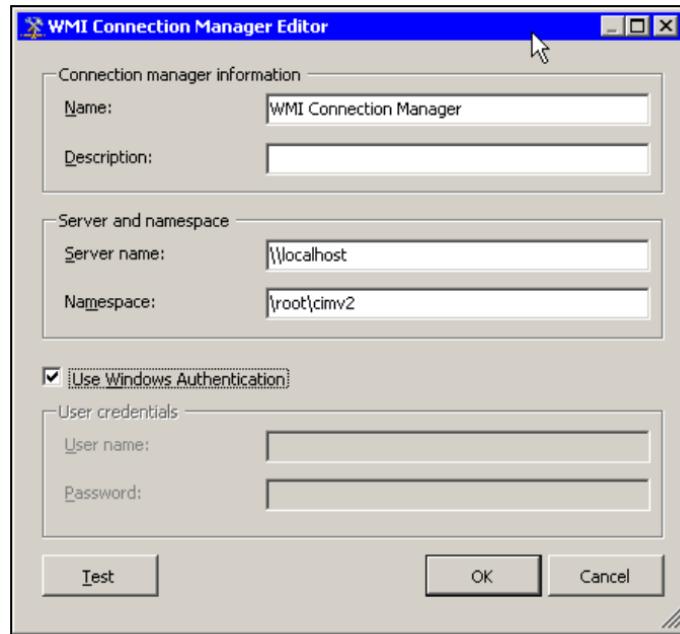


Рис. 10.10. Настройка WMI Connection Manager

- Выберите нужный источник запроса при помощи свойства **WmiQuerySourceType** (Тип источника запроса WMI) (в этом примере будем использовать самый простой вариант — **Direct Input**), а затем введите текст запроса как значение для свойства **WqlQuerySource** (Источник запроса WQL). В данном случае текст запроса может быть таким:

```
SELECT * FROM __instancedeletionevent WITHIN 1 WHERE TargetInstance  
ISA 'Win32_Process' AND TargetInstance.Name = 'WinRAR.exe'
```

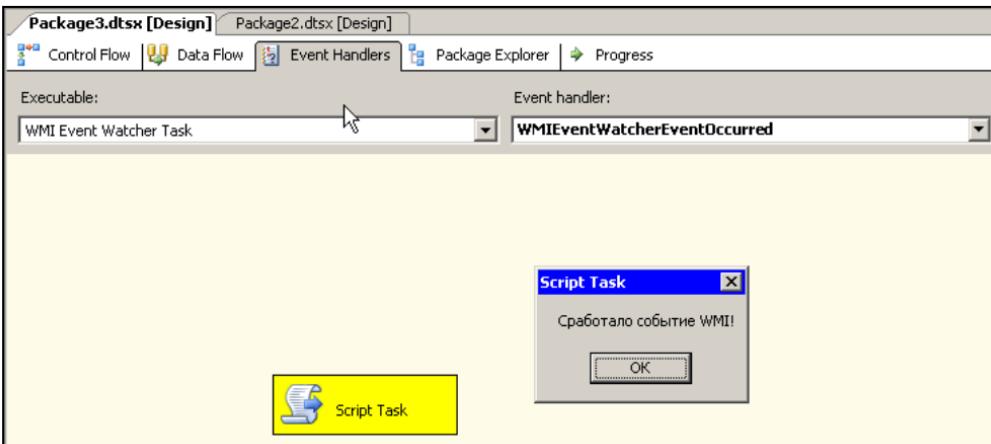


Рис. 10.11. Ваш пакет среагировал на завершение разархивирования

Этот запрос с интервалом в 1 секунду будет опрашивать объект Win32_Process на предмет завершения работы процесса с именем 'WinRAR.exe'. В реальной (не демонстрационной, как здесь) ситуации можно было бы сделать интервал опроса больше (например, 10 секунд), чтобы снизить расход системных ресурсов.

- Для свойства **ActionAtEvent** (Действие при наступлении события) настройте значение **Log the event and fire the SSIS Event** (Запротоколировать событие и запустить событие SSIS), чтобы при возникновении события WMI информация о нем не только записывалась в протокол выполнения пакета, но и срабатывало бы событие пакета. Для остальных свойств задачи WMI Event Watcher Task можно оставить значения по умолчанию.
- Следующее, что вам потребуется сделать, — перейти в WMI Designer на вкладку **Event Handlers**. На ней в списке **Executable** нужно выбрать **WMI Event Watcher Task**, а в списке **Event Handler** (Обработчик события) — событийную процедуру **WMIEventWatcherEventOccured**. Далее щелкните мышью по ссылке **Click here to create a "WMIEventWatcherEventOccured" event handler...** (Щелкните здесь для создания обработчика события "WMIEventWatcherEventOccured"...).
- Последнее, что вам осталось сделать, — настроить реакцию на событие в виде задачи Script Task. Для этого нужно перетащить из **Toolbox** на открывшееся розовое поле на вкладке **Event Handlers** задачу Script Task и настроить ее свойства. На вкладке **Script** свойств этой задачи нужно нажать кнопку **Design Script** и ввести требуемый код для процедуры **Main()**. Он может быть, например, таким:

```
Public Sub Main()
    MsgBox("Сработало событие WMI!")

```

```
Dts.TaskResult = Dts.Results.Success  
End Sub
```

После этого можно запускать пакет на выполнение. Запущенный пакет может работать неопределенное время, расходуя минимальное количество системных ресурсов. Как только вы запустите и потом закроете WinRAR (у вас настроена реакция на закрытие этого приложения), сразу же сработает событийная процедура со Script Task (рис. 10.11).

10.20. Задачи Analysis Services

Последняя группа задач — это задачи, предназначенные для выполнения каких-либо операций в базах данных OLAP на Analysis Services. Поскольку работа с OLAP и Analysis Services в этой книге не рассматривается, далее приводится лишь краткая характеристика каждой из этих задач:

- **Analysis Services Execute DDL Task** — эта задача позволяет создавать, изменять и удалять объекты в базах данных OLAP на Analysis Services (кубы, измерения, модели добычи данных и т. п.);
- **Analysis Services Processing Task** — наиболее часто используемая задача из этой группы. Она позволяет выполнить процессинг куба или измерения, а также тренировку модели добычи данных. Очень часто пакеты с такими задачами используются для автоматизации процессинга кубов OLAP и моделей добычи данных;
- **Data Mining Query Task** — эта задача выполняет запрос к модели добычи данных в базе данных OLAP и использует в пакете возвращаемые этим запросом результаты. Например, можно передать все собранные данные о заемщике модели добычи данных (скоринг-системе) и получить от нее информацию по количеству набранных этим заемщиком баллов, от которых будет зависеть решение о возможности выдачи кредита и процентной ставке по нему.

10.21. Контейнеры SSIS

В SQL Server Integration Services появилась новая возможность — контейнеры. Контейнеры — это специальные задачи Control Flow, которые предназначены для размещения в них других задач Control Flow. В основном контейнеры используются для организации циклов в пакетах. Отметим, что в DTS 2000 цикл в пакете можно было организовать только при помощи программирования (средствами ActiveX Script Task). Такая же возможность (но уже с применением Script Task) осталась и в SSIS, но использование контейнеров проще и удобнее.

В SSIS предусмотрено четыре типа контейнеров:

- For Loop;**
- Foreach Loop;**
- Sequence;**
- Task Host.**

Все контейнеры доступны в **Toolbox** на вкладке **Control Flow**, кроме **Task Host**. Этот контейнер создается автоматически для любой задачи и используется для хранения значений свойств, настроенных на графическом экране SSIS Designer для этой задачи. **Task Host** нельзя увидеть в SSIS Designer — обращение к нему возможно только из программного кода, и поэтому здесь этот контейнер рассматриваться не будет.

Контейнер **For Loop** предназначен для организации цикла. Как и синтаксическая конструкция `For Loop` в программном коде, этот контейнер используется в тех ситуациях, когда вам с самого начала известно, сколько раз должен быть выполнен тот или иной набор задач **Control Flow**. Для этого контейнера предусмотрено три главных свойства:

- InitExpression** — определяет исходное значение счетчика. Например, если в вашем пакете определена целочисленная переменная `Counter`, то значение этого свойства может выглядеть как `@Counter = 0`;
- EvalExpression** — проверяемое выражение. Цикл будет выполняться до тех пор, пока это выражение возвращает истину. Например, если вам нужно выполнить какие-то задачи пять раз, то значение этого свойства может выглядеть как `@Counter < 5`;
- AssignExpression** — выражение, которое будет изменять значение вашего счетчика. Оно может выглядеть, например, как `@Counter = @Counter + 1`.

После настройки свойств контейнера **For Loop** вам останется просто перетащить в него нужную задачу (или набор задач) из **ToolBox** и настроить ее.

Контейнер **Foreach Loop**, более сложный и функциональный, используется чаще. Программная конструкция `Foreach` применяется для того, чтобы пройти по всем элементам какой-то коллекции. Для этой же цели используется и контейнер **Foreach Loop**. В нем можно проходить по элементам следующих коллекций:

- File Enumerator** — эта коллекция позволяет пройти по всем файлам какой-либо папки в файловой системе. Можно настроить фильтр для имен и расширений файлов, а также выбрать, будете ли вы проходить по вложенным папкам. Имена файлов можно передать в переменные пакета (об этом будет рассказываться в *разд. 10.25*) и использовать их для других задач, например, для **Data Flow Task** или **File System Task**;

- **Item Enumerator** — проход по записям таблицы, которую нужно будет создать прямо в окне свойств этого контейнера. Это самый простой, но и самый неудобный в использовании тип коллекции;
- **ADO Enumerator** — проход по всем записям в объекте ADO.Recordset, или по всем записям объекта DataTable объекта DataSet в ADO.NET, или по всем записям во всех таблицах объекта DataSet в ADO.NET, или по объектам таблиц в DataSet. Для этой коллекции вам потребуется указать имя объектной переменной пакета, в которой будет храниться соответствующий объект ADO.Recordset или DataSet;
- **ADO.NET Schema Rowset Enumerator** — коллекция объектов определенного типа на источнике данных OLE DB. Под "*schema*" в данном случае подразумевается тип объекта (таблица, представление, хранимая процедура), а не схема SQL Server 2005. Вы можете выбрать источник данных (только OLE DB), базу данных и тип объекта, а также настроить фильтр для свойств этого объекта, например, фильтр на имя таблицы;
- **From Variable Enumerator** — проход по набору элементов в коллекции, которая находится в переменной пакета. Например, в переменной может находиться набор записей, возвращенных запросом Execute SQL Task;
- **Nodelist Enumerator** — проход по всем узлам документа XML, которые были отфильтрованы при помощи выражения XPath;
- **SMO Enumerator** — коллекция объектов SMO (например, таблиц, представлений, хранимых процедур на SQL Server 2005). Этот вариант требует менеджера подключений **SMO Connection Manager** и строки в формате URN, которая будет отфильтровывать нужные объекты. Ее можно генерировать в автоматическом режиме.

Удобство применения контейнера **Foreach Loop** во многом объясняется тем, что он умеет передавать информацию о текущем обрабатываемом элементе переменным пакета. Затем можно использовать эти значения, например, для того, чтобы изменить свойства элементов пакета, помещенных в этот контейнер для загрузки всех файлов из каталога на SQL Server. Для настройки соответствий переменных возвращаемым значениям используется вкладка **Variable Mappings** (Привязки переменных) свойств контейнера **Foreach Loop**. На ней вы можете указать, в какую переменную какое из набора значений, получаемых для данного элемента, будет передаваться. Например, если вы работаете с коллекцией ADO Enumerator, то индексу 0 будет соответствовать значение из первого столбца таблицы для данной записи, индексу 1 — из второго и т. п. К сожалению, для большинства коллекций такая привязка неочевидна. Например, если вы используете коллекцию File Enumerator, то что будет соответствовать первому столбцу, а что — второму? В документации к SQL Server этой информации автору обнаружить не удалось. Скорее всего,

самый лучший способ в этой ситуации — провести эксперимент и, например, вывести значение каждой переменной при помощи функции `MsgBox` в Script Task.

Последний контейнер Control Flow называется **Sequence**. Это самый простой из контейнеров. Обычно он используется только для одной цели — сгруппировать набор элементов, чтобы их можно было одновременно отключать без удаления. Для этого достаточно просто воспользоваться свойством **Disable** (Отключить) этого контейнера. Кроме того, на уровне контейнера можно объявлять переменные, которые не будут видны в других частях пакета.

10.22. Задачи планов обслуживания

Планы обслуживания (*maintenance plans*) — это наборы операций по обслуживанию базы данных, которые обычно выполняются на регулярной основе. Чаще всего планы обслуживания создаются при помощи мастера из SQL Server Management Studio (см. разд. 8.3). Однако вы можете создать план обслуживания и вручную при помощи соответствующего набора элемента на вкладке **Control Flow**. Все эти элементы сгруппированы в разделе **Maintenance Plan Tasks** (Задачи планов обслуживания) в **Toolbox**. Все они очень просты и подробных комментариев не требуют. Приведем лишь их список с краткими характеристиками:

- **Backup Database Task** — эта задача позволяет провести резервное копирование базы данных или журнала транзакций;
- **Check Database Integrity Task** — проверка целостности базы данных (фактически это соответствует выполнению команды `DBCC CHECKDB`);
- **Execute SQL Server Agent Job Task** — задача, при помощи которой можно выполнить задание SQL Server Agent;
- **Execute T-SQL Statement Task** — задача, позволяющая просто выполнить определенную команду Transact-SQL на SQL Server 2005. В отличие от Execute SQL Task, определять передаваемые и принимаемые параметры для этой задачи нельзя. Кроме того, она работает только с SQL Server 2005, в SQL Server 2000 ее использовать не получится;
- **History Cleanup Task** — очистка истории резервного копирования, или истории выполнения заданий SQL Server Agent, или истории выполнения планов обслуживания. Цель проста — уменьшить размер базы данных `msdb`, удалив из нее ненужную информацию;
- **Maintenance Cleanup Task** — эта задача позволяет удалить старые файлы. Обычно она используется для удаления старых файлов резервных копий, но ничего не мешает вам использовать ее и для удаления любых других файлов;

- **Notify Operator** — эта задача передает сообщение оператору SQL Server Agent. В свойствах задачи вы выбираете оператора и создаете сообщение, которое будет отправлено SQL Server Agent;
- **Rebuild Index Task** — полное перестроение индексов (обычно для целей дефрагментации). При настройке этой задачи необходимо выбрать таблицу или представление или указать все таблицы или представления в определенной базе данных;
- **Reorganize Index Task** — задача для "мягкой" реорганизации индексов. При использовании этой задачи индексы реорганизуются, а не перестраиваются. Степень дефрагментации получается несколько хуже, зато требуется меньше системных ресурсов, и таблицы на время реорганизации остаются доступными для пользователей;
- **Shrink Database Task** — эта задача позволяет уменьшить размер базы данных, высвободив из ее файлов неиспользуемое пространство. Обычно такая задача применяется для баз данных промежуточного хранения с большим количеством временных объектов и данных;
- **Update Statistics Task** — эта задача обновляет статистику оптимизатора запросов (Query Optimizer) для объектов в базах данных SQL Server 2005. Обычно такая задача используется для больших баз данных, в которых автоматическое обновление статистики отключено в целях оптимизации производительности. Подробно про работу со статистикой рассказывается в разд. 11.5.8.

10.23. Ограничения предшественников

Последний тип элементов, который можно использовать на вкладке **Control Flow** в SSIS Designer, называется *ограничениями предшественников* (*precedence constraints*). В **Toolbox** этих элементов вы не найдете. Они представлены зелеными (по умолчанию) стрелками, которые появляются для задачи или контейнера, если эту задачу или контейнер выделить на вкладке **Control Flow** в SSIS Designer.

Главное назначение этих стрелок — определение порядка и логики выполнения элементов в пакете. Если в вашем пакете будет несколько элементов, которые не будут связаны между собой (рис. 10.12), то, как несложно убедиться, выполнение всех этих элементов начнется одновременно. Для такой ситуации существует специальное название — несколько точек входа в пакете.

Если же вы свяжите задачи между собой ограничениями предшественников (рис. 10.13), то эти задачи будут выполняться последовательно (и, возможно, опираясь на результаты выполнения предыдущей задачи).



Рис. 10.12. Несколько точек входа в пакете

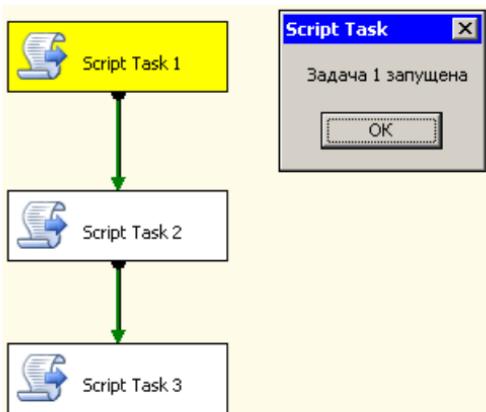


Рис. 10.13. Единственная точка входа в пакете

Работа с ограничениями предшественников выглядит так же, как и работа с потоками ввода/ввода в Data Flow Task: нужно просто взять зеленую стрелку, которая выходит из задачи, и перетащить ее на задачу, которая должна выполняться следующей. Однако свойства ограничений предшественников сильно отличаются от свойств потоков данных в Data Flow Task.

Главное свойство, которое можно настроить для стрелки, — это ее тип. Он выбирается из контекстного меню для стрелки. По умолчанию используется значение **Success** (Успех), т. е. следующая задача, на которую указывает эта стрелка, будет выполнена только в том случае, если предыдущая задача завершится успешно. Этому типу соответствует зеленый цвет стрелки.

Если выбрать для стрелки тип **Failure** (Сбой), то переход к элементу, на который она указывает, будет произведен только тогда, когда в ходе выполнения предыдущей задачи возникнет ошибка. Цвет стрелки этого типа красный.

Последний тип элемента логики выполнения — **Completion** (Завершение). В этом случае следующая задача начнет выполняться в любом случае после завершения предыдущей, независимо от того, успешно она была выполнена или нет. Для таких стрелок предусмотрен синий цвет.

Из одного элемента могут выходить несколько стрелок. Точно так же на один элемент могут указывать несколько стрелок. В последнем случае (рис. 10.14) можно определить тип объединения для условий перехода к данной задаче:

- логическое **OR** (ИЛИ) — переход к следующей задаче произойдет, если любое из условий окажется истинным (например, успешно завершится какая-нибудь из предыдущих задач);
- логическое **AND** (И) — переход к следующей задаче произойдет только в случае, если истинными окажутся все условия (например, успешно завершатся все предыдущие задачи).

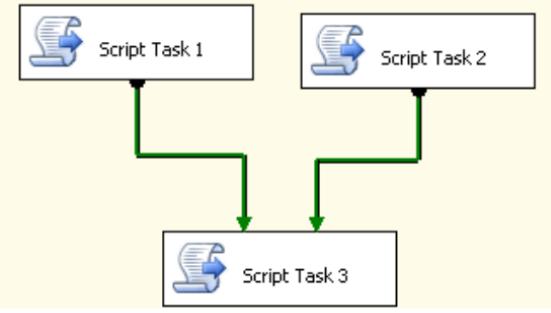


Рис. 10.14. Два ограничения предшественников для задачи Script Task 3

Проверку на успешность можно производить не только при помощи значений констант, которые возвращает задача при выполнении, но и при помощи своих собственных выражений. Их можно настроить в свойствах элемента логики выполнения.

10.24. Протоколирование выполнения пакетов

Очень многие пакеты SSIS изначально предназначаются для выполнения в ночное время, когда нагрузка на сервер баз данных минимальна. Конечно, в этом случае важно иметь возможность получить информацию о том, как

прошло выполнение пакета, и если возникли ошибки, то какие. Этую информацию можно получить из протоколов выполнения пакета.

Настройку параметров протоколирования пакета проще всего произвести при помощи меню **SSIS | Logging** (Протоколирование) в SSIS Designer. Откроется окно **Configure SSIS Logs** (Настроить журналы SSIS) (рис. 10.15).

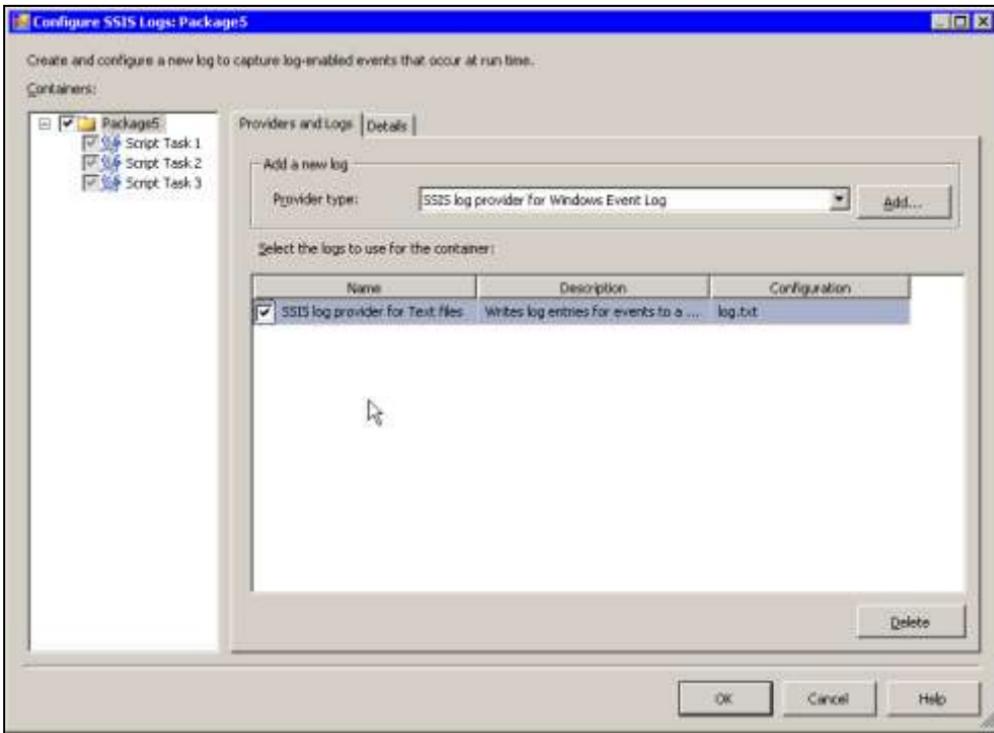


Рис. 10.15. Настройка протоколирования пакета SSIS

Первое, что нужно сделать в этом окне, — установить в дереве **Containers** (Контейнеры) флажки напротив пакета и тех его элементов, для которых необходимо протоколирование. Затем на вкладке **Providers and Logs** (Провайдеры и журналы) необходимо выбрать тип поставщика протоколирования (т. е. формат протоколирования). В вашем распоряжении есть пять вариантов:

- SSIS log provider for text files** — протоколирование будет производиться в текстовый файл. При настройке этого поставщика вам придется указать или создать менеджер подключения **File Connection Manager**;
- SSIS log provider for SQL Profiler** — протоколирование также будет производиться в файл, но уже в формате файлов SQL Server Profiler (с расширением trc). Просматривать этот файл придется в профилировщике. Для

этого поставщика также используется менеджер подключения **File Connection Manager**;

- **SSIS log provider for SQL Server** — протоколирование будет производиться в таблицу sysdtslog90 в выбранной базе данных SQL Server. Для этого поставщика нужен менеджер подключения **OLE DB Connection Manager**;
- **SSIS log provider for Windows Event Log** — протоколирование будет производиться в журнал событий приложений (Application Event Log) журнала событий Windows. Менеджер подключения для этого типа протоколирования не нужен;
- **SSIS Log Provider for XML files** — запись протокола будет выполняться в файлы XML на диске. Для этого подключения также используется менеджер подключения **File Connection Manager**.

Обычно для протоколирования используются текстовые файлы (если протокол собирается просматривать администратор) или таблица sysdtslog90 на SQL Server, если информацию о выполнении пакетов на вашем предприятии нужно хранить в течение длительного времени.

Следующее, что нужно сделать, — перейти на вкладку **Events** (События) и выбрать события для записи в протокол. Обратите внимание, что более точно выбрать события можно при помощи кнопки **Advanced** (Дополнительно) на этой вкладке.

После этого при каждом запуске пакета на выполнение информация будет записываться в указанное вами место назначения.

10.25. Работа с конфигурациями

Конфигурация пакетов (*package configuration*) — это средство, которое позволяет изменять свойства пакетов во время их выполнения, обеспечивая таким образом гибкость при запуске пакета в разных условиях. Например, один и тот же пакет можно использовать для загрузки разных файлов на разные серверы баз данных. Имена и местонахождение файлов, имена серверов баз данных и другие параметры пакета можно будет изменять при помощи информации из конфигурации прямо в процессе работы пакета. Для тех, кто работал с DTS 2000, отметим, что конфигурации SQL Server 2005 предназначены для тех же целей, что и Dynamic Properties Task в предыдущей версии DTS.

Создание конфигурации производится при помощи меню **SSIS | Configurations** в SSIS Designer. В окне **Package Configurations Organizer** нужно установить флажок **Enable package configurations**, нажать кнопку **Add** и прой-

ти по экранам мастера создания конфигураций Package Configuration Wizard. В SSIS предусмотрено пять типов конфигураций:

- **XML Configuration File** (Файл конфигурации XML) — информация для изменения свойств элементов пакетов будет браться из файла в формате XML на диске. Этот тип конфигурации позволяет определять значения сразу для нескольких свойств элементов пакета SSIS;
- **Environment Variable** (Переменная окружения) — значение для свойства элемента пакета будет браться из переменной окружения операционной системы (все имеющиеся на компьютере переменные окружения можно просмотреть, выполнив в командной строке операционной системы команду `set`). Например, можно использовать имя текущего компьютера для настройки свойств подключения в пакете SSIS;
- **Registry Entry** (Запись в реестре) — значение будет браться из выбранного вами параметра реестра на локальном компьютере;
- **Parent Package Variable** (Переменная родительского пакета) — этот случай уже рассматривался в разд. 10.12, посвященном Execute Package Task. Этот тип конфигурации используется в ситуации, когда один пакет (внешний или "родительский") запускает на выполнение другой пакет, изменяя при этом его свойства. Значение будет браться из переменной родительского пакета;
- **SQL Server** — значения для свойств элементов пакета будут браться из таблицы в базе данных SQL Server. Формат этой таблицы строго предопределен (саму таблицу в нужном формате можно сгенерировать при помощи кнопки **New** окна **Select Configuration Type**). Так же, как и при использовании файла конфигурации XML, конфигурация **SQL Server** позволяет хранить сразу несколько новых значений для свойств элементов пакета.

Для всех типов конфигурации, кроме **Environment Variable**, можно указывать опосредованно назначение (например, конкретное имя файла XML) — при помощи выбранной вами переменной окружения операционной системы.

При создании конфигурации в виде файла XML файл с текущим значением свойства будет сгенерирован автоматически. Вам останется только открыть его и изменить нужное значение в текстовом редакторе. Конечно, файл конфигурации можно создать и вручную, но это более трудоемкий способ.

После того, как выбран нужный тип конфигурации, осталось нажать кнопку **Next** и выбрать те свойства элементов пакета (или самого пакета), которые будут изменяться на значения из выбранного вами источника.

Для пакета можно указать несколько конфигураций и определить для них порядок применения. Если одно и то же свойство будет изменяться несколь-

кими конфигурациями, то в итоге будет применено значение из последней конфигурации в списке.

10.26. Хранение пакетов

По умолчанию пакеты SSIS создаются и записываются в папку своего проекта Visual Studio в каталоге **Мои документы** текущего пользователя. Поскольку пакеты представляют собой текстовые файлы в формате XML с расширением `dtsx`, то, в принципе, после создания пакета и сохранения его на диске, можно больше ничего с ним не делать. Сохраненные вами пакеты можно запускать как из SSIS Designer, так и при помощи утилит `dtexec` и `dtexecui`. Однако понятно, что во многих ситуациях папка **Мои документы** — не лучшее место для хранения пакетов. Нужно иметь возможность корректно переносить пакеты с места на место с сохранением всей необходимой служебной информации (например, информации о конфигурациях).

Пакеты SSIS можно хранить только двумя способами: в виде файлов в папках на диске и в таблице `sysdtspackages90` в базе данных `msdb` на SQL Server 2005. В некоторых ситуациях (например, при выборе пакета средствами утилиты `dtsrunui`) в списке для выбора местонахождения пакетов еще одно возможное место хранения, которое называется SSIS Package Store. Пакеты, которые видны в SSIS Package Store, физически находятся или в файлах на диске, или базе данных `msdb` — просто о них "знают" службы Integration Services. Информация о таких пакетах хранится в параметрах конфигурации Integration Services на данном компьютере.

Рекомендованный способ помещения пакета на сервер (в файловую систему или в базу данных `msdb`) — применение мастера развертывания пакетов Package Deployment Wizard. При этом информация о пакете будет помещена на Integration Services данного сервера, и вы сможете выполнять с ним административные операции средствами SQL Server Management Studio. Заметим, что, в принципе, пакеты SSIS можно копировать просто средствами файловой системы и после этого запускать их на выполнение, но в этом случае административные операции с ними производить будет сложнее. Кроме того, вы рискуете случайно забыть сопутствующие пакетам служебные файлы, например, файлы конфигурации.

Работа по помещению пакета на сервер производится со всем проектом Integration Services целиком. В проект могут входить один или несколько пакетов, информация об источниках данных, а также вспомогательные файлы (например, файлы с документацией, которые помещаются в контейнер **Miscellaneous** (Разное)). Помещение пакетов проекта на сервер наиболее recommendedным способом выглядит следующим образом.

1. Первое, что нужно сделать, — создать утилиту развертывания (*deployment utility*). Под утилитой развертывания подразумевается всего лишь набор всех файлов проекта, скопированный в определенную папку (по умолчанию в bin\Deployment в папке проекта), к которым добавляется XML-совместимый файл с расширением SSISDeploymentManifest. Для создания утилиты развертывания нужно выполнить следующие действия:

- открыть свойства проекта (не пакета!) в **Project Explorer**;
- на вкладке **Deployment Utility** (Утилита развертывания) установить для свойства **Create Deployment Utility** (Создать утилиту развертывания) значение **TRUE**. При необходимости можно изменить значения и других свойств проекта, например, настроить другой каталог для размещения файлов Deployment Utility;
- в меню **Build** (Создать) выбрать команду **Build** для вашего проекта.

В результате в выбранный вами каталог будут скопированы все необходимые файлы проекта.

2. Следующее действие — использование мастера развертывания пакетов Package Deployment Wizard для размещения проекта на сервере и помещения информации о всех его пакетах в параметры конфигурации Integration Services. Проще всего запустить мастер развертывания пакетов, щелкнув два раза мышью по сгенерированному файлу с расширением SSISDeploymentManifest в папке утилиты развертывания.

3. В ходе работы мастера вам будет нужно выбрать, где именно будут сохранены проекты пакета. В вашем распоряжении есть два варианта:

- **File system deployment** (Развертывание в файловой системе) — пакеты будут скопированы в папку файловой системы сервера, по умолчанию C:\Program Files\Microsoft SQL Server\90\DTSPackages\имя_проекта;
- **SQL Server Deployment** (Развертывание на SQL Server) — пакеты будут сохранены в базе данных msdb на сервере SQL Server.

Если вы выберете первый вариант, вам потребуется указать каталог для размещения пакетов (или согласиться с предлагаемым по умолчанию). Во втором варианте вам нужно указать сервер SQL Server 2005, параметры аутентификации при подключении к нему, а также каталог на диске, в который при необходимости будут помещены вспомогательные файлы вашего проекта.

Для размещения пакета на сервере можно также использовать утилиту dtexec. Но в этом случае вам придется работать с командной строкой.

После того, как развертывание пакетов завершено, переносить их между серверами и выполнять с ними другие административные операции проще всего

средствами SQL Server Management Studio. Для этого достаточно открыть SQL Server Management Studio, в панели инструментов **Object Explorer** нажать кнопку **Connect** (Подключиться) и в открывшемся списке выбрать Integration Services, а затем указать имя вашего сервера и параметры подключения. В результате все пакеты, про которые знает Integration Services, появятся в дереве SQL Server Management Studio (рис. 10.16). Вы можете экспортировать эти пакеты, импортировать новые пакеты, запускать их на выполнение, управлять безопасностью пакетов и т. п.

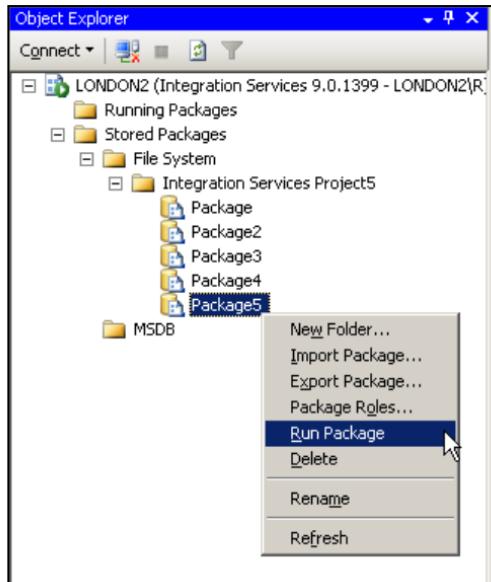


Рис. 10.16. Администрирование пакетов SSIS из SQL Server Management Studio

Резервное копирование пакетов производится в зависимости от того, где именно они были сохранены. Если пакеты SSIS были сохранены в файловой системе, используются обычные средства резервного копирования Windows. Если пакеты были помещены в базу данных msdb, то резервное копирование производится вместе со всей базой данных msdb средствами SQL Server.

10.27. Безопасность пакетов SSIS

Пакеты SSIS являются по своей сущности просто файлами в формате XML, в которых вся информация хранится в открытом виде. В то же время информация в пакетах может представлять ценность для злоумышленников. Например, в пакете может находиться имя пользователя и пароль для подключения к источнику данных.

Другая задача, которая может возникнуть перед администратором на предприятии, — защита пакета от несанкционированного запуска. Учитывая, что большинство пакетов предназначено для загрузки и выгрузки данных, последствия незапланированного запуска пакета могут оказаться очень серьезными.

И, наконец, третья проблема — защита программного кода пакета как от просмотра, так и от несанкционированного изменения со стороны пользователей. Очень часто пакеты являются частью большого приложения с использованием SQL Server 2005, и разработчики не хотят, чтобы код их пакетов был скопирован или изменен.

В этом разделе будут рассмотрены возможности, которые реализованы в Integration Services для защиты пакетов.

Первая возможность — настройка уровня защиты пакета (*protection level*). Уровень защиты пакета проще всего настроить, если щелкнуть правой кнопкой мыши по пустому пространству на вкладке **Control Flow** и в контекстном меню выбрать **Properties** (при открытии свойств пакета из **Solution Explorer** открывается другой набор свойств). В группе **Security** (Безопасность) свойств пакета находится свойство **ProtectionLevel** (Уровень защиты). Для него можно выбрать следующие значения:

- **EncryptSensitiveWithUserKey** (это значение выбирается по умолчанию) — вся секретная информация (например, информация о паролях при подключении к источникам данных) будет шифроваться ключом пользователя (который находится в его профиле). Если этот пакет откроет другой пользователь, то ему придется ввести свои значения вместо зашифрованных;
- **DontSaveSensitive** — вся секретная информация просто не будет сохраняться. Любому пользователю (в том числе и автору пакета) придется при следующем открытии пакета вводить ее повторно;
- **EncryptSensitiveWithPassword** — вся секретная информация будет защищена паролем (т. е. зашифрована по алгоритму Triple DES с ключом 192 бита). Этот пароль нужно будет указывать при открытии пакета и при запуске его на выполнения. Если пароль неизвестен, то всю эту информацию придется вводить заново;
- **EncryptAllWithPassword** — весь пакет (а не только секретная информация) будет зашифрован при помощи пароля. Без знания пароля пакет невозможно будет открыть. Пароль в этом случае (как и в предыдущем) потребуется не только для просмотра и редактирования пакета, но и для его запуска на выполнение;
- **EncryptAllWithUserKey** — весь пакет будет зашифрован при помощи пользовательского ключа, который хранится в профиле пользователя. Никакой

другой пользователь не сможет ни открыть пакет на просмотр или редактирование, ни запустить его на выполнение;

- `ServerStorage` — шифроваться в пакете ничего не будет. Вместо этого для защиты будут использоваться роли базы данных `msdb` (про них речь пойдет далее в этом разделе). Этот вариант можно использовать только тогда, когда пакет хранится в таблице `sysdtspackages90` базы данных `msdb`.

Уровень защиты пакета можно менять также, например, при экспорте и импорте пакетов средствами SQL Server Management Studio.

Отметим два принципиальных момента. Во-первых, вы не можете оставить пакет вообще без защиты, такой возможности просто не предусмотрено. Во-вторых, по умолчанию для шифрования важных данных используется ключ, который хранится в профиле пользователя. По умолчанию пользователь работает со своим локальным профилем, который расположен в каталоге `Documents and Settings` на его локальном компьютере. Как для пользователя, так и для разработчика может оказаться неприятным сюрпризом, что при обращении к своим пакетам с другого компьютера пакеты придется исправлять. Кроме того, если профиль пользователя утрачен (например, операционная система на его рабочей станции была переустановлена), а для пакетов было установлено значение `EncryptAllWithUserKey`, то пакеты придется создавать заново с нуля. Поэтому рекомендуется использовать для защиты пакетов либо пароли (если пакеты сохраняются в файловой системе), либо роли базы данных `msdb`.

Еще одна очень удобная возможность защиты пакетов SSIS — использование ролей базы данных `msdb`. Отметим еще раз, что эта возможность может применяться только к тем пакетам, которые хранятся в базе данных `msdb`. Защиту ролями можно сочетать с защитой при помощи пользовательских ключей и паролей, а можно сделать единственной для пакета (если выбрать уровень защиты пакета `ServerStorage`). Изначально в базе данных `msdb` предусмотрено три специальных роли SSIS:

- `db_dtsadmin` — члены этой роли получают административные права на пакеты DTS. Они могут просматривать, изменять, удалять, запускать на выполнение любые пакеты SSIS в базе данных `msdb`. Права этой роли автоматически получает пользователь с правами роли `sysadmin` для SQL Server;
- `db_dtsuser` — члены этой роли могут создавать пакеты (т. е. импортировать их в базу данных `msdb`), изменять, удалять и запускать на выполнение свои пакеты, т. е. те пакеты, владельцем которых они являются. Прав на пакеты других пользователей они не получают;
- `db_dtsoperator` — это роль для операторов SSIS. Пользователи этой роли могут просматривать любые пакеты, экспортить их, запускать их на

удаление и планировать к исполнению средствами SQL Server Agent. Однако изменять пакеты они не могут.

Кроме этого, можно использовать и другие средства назначения разрешений в базах данных SQL Server, чтобы предоставить какому-либо пользователю или группе пользователей специальный набор прав (например, определенному пользователю нужно разрешить запускать только один пакет SSIS). Для этого достаточно открыть Integration Services в окне **Object Explorer** в SQL Server Management Studio, щелкнуть правой кнопкой мыши по объекту пакета и в контекстном меню выбрать **Package Roles** (Роли пакета).

Если пакеты лежат в папках сервера (или, например, в сетевых каталогах на файл-сервере), то для их защиты рекомендуется использовать обычные разрешения NTFS или общих папок.

В SSIS предусмотрена еще одна возможность защиты пакетов от несанкционированного изменения — применение цифровых подписей для пакетов. Если пакет, защищенный цифровой подписью, будет несанкционированно изменен (и, таким образом, перестанет соответствовать своей подписи), то по умолчанию его будет просто невозможно загрузить или запустить на выполнение. Настройка проверки цифровой подписи производится из SSIS Designer (меню **Tools | Options** (Сервис | Настройки)), затем в открывшемся окне параметров нужно раскрыть узел **Business Intelligence Designers | Integration Services Designers | General** (Дизайнеры Business Intelligence | Дизайнеры Integration Services | Общие)). Нужный параметр называется **Check digital signature when loading a package** (Проверять цифровую подпись при загрузке пакета). Там же можно настроить параметр **Show warning if a package is unsigned** (Показывать предупреждение, если пакет не подписан).

Подписать пакет при помощи цифрового сертификата можно, открыв этот пакет в SSIS Designer и выбрав в меню **SSIS** пункт **Digital signing** (Цифровая подпись). Перед настройкой цифровой подписи нужно создать сертификат средствами Certificate Services и установить его в локальное хранилище сертификатов при помощи Internet Explorer.

10.28. Запуск пакетов SSIS на выполнение

После того, как пакет создан, помещен на SQL Server и для него настроены параметры защиты, осталось сделать последнее — обеспечить его запуск на выполнение. Запустить пакет на выполнение можно несколькими способами.

С первым способом вы уже знакомы. Пакет можно запустить на выполнение из той же среды, в которой он создается, т. е. из SSIS Designer. Этот вариант очень удобен для отладки. Например, вы можете при помощи меню **Debug** установить точки останова внутри элементов пакета и в режиме паузы прове-

рить значения переменных. Очень удобно для отладки использовать также просмотрщика данных (Data Viewer) на вкладке **Data Flow** (см. разд. 10.6.5). Однако для запуска пакетов на регулярной основе использовать SSIS Designer, конечно, неудобно.

Наиболее рекомендованный способ запуска пакетов — применение графической утилиты dtexecui или консольной утилиты dtexec. При этом командную строку для dtexec очень удобно генерировать при помощи графического интерфейса dtexecui. Поэтому параметры командной строки dtexec рассматриваться не будут, а возможности dtexecui рассмотрим подробно, поскольку не все они очевидны.

Утилиту dtexecui можно запустить просто из командной строки операционной системы, а можно воспользоваться командой **Run package** (Запустить пакет) в SQL Server Management Studio. Первое, что необходимо сделать, — выбрать нужный пакет на вкладке **General** графического интерфейса этой утилиты. Как уже говорилось ранее, пакеты SSIS могут храниться или в виде файлов в файловой системе сервера, или в виде записей в таблице sysdtspackages90 базы данных msdb. Пакеты, которые находятся в SSIS Package Store, физически точно так же находятся в файлах на диске или в базе данных msdb, но для этих пакетов предусмотрена дополнительная информация в файлах конфигурации Integration Services.

Следующая вкладка графического интерфейса dtexecui — это вкладка **Configurations** (Конфигурации). На этой вкладке можно добавить конфигурации для пакета (см. разд. 10.25), которые будут изменять его при выполнении. Из пяти возможных типов конфигураций для пакетов здесь вы можете выбрать только один тип — файлы конфигурации в формате XML. Все остальные типы конфигурации вам придется настраивать еще на этапе создания пакетов в SSIS Designer.

На следующей вкладке, которая называется **Command Files** (Командные файлы), вы можете определить один или несколько текстовых файлов с командами. По наблюдениям автора, многие специалисты здесь становятся в тупик: что это за файлы с командами, про которые не найти никакого упоминания при работе с пакетами? Документация по dtexecui также не вносит никакой ясности: "Здесь можно указать командные файлы, которые будет использовать пакет". На самом деле все очень просто: в командном файле должны находиться дополнительные параметры для утилиты dtexec, которые будут передаваться ей напрямую, без возможности обработки и редактирования. Если вы укажете на этой вкладке, например, файл D:\ssis.cmd, то командная строка будет дополнена параметром /COMMANDFILE "D:\ssis.cmd".

На вкладке **Connection Managers** будут перечислены все менеджеры подключений, имеющиеся в пакете. Если вы установите флажок напротив ме-

неджера подключения, у вас появится возможность отредактировать строку подключения для этого менеджера. Исправленная строка подключения будет использована только на время данного запуска пакета. В основном эта возможность предоставлена для заполнения данных, которые зашифрованы пользовательским ключом, и при запуске от имени другого пользователя недоступны.

На вкладке **Execution Options** (Параметры выполнения) можно настроить еще несколько важных параметров:

- Fail the package on validation warnings** (Выходить с ошибкой при проверочных предупреждениях) — если при проверке пакета будет сгенерировано какое-либо предупреждение, эта утилита вообще не будет пытаться запускать пакет, а сразу завершит работу с ошибкой;
- Validate package without executing** (Проверять пакет без выполнения) — если установить этот флагок, то при нажатии на кнопку **Execute** (Выполнить) пакет будет просто проверен на наличие ошибок. Выполниться пакет не будет;
- Maximum concurrent executables** (Максимальное количество исполняемых файлов) — этот параметр определяет, сколько исполняемых файлов среды выполнения SSIS может быть одновременно запущено во время работы этого пакета (запуск дополнительных исполняемых файлов может потребоваться, например, если ваш пакет запускает другие пакеты). По умолчанию это значение равно -1 , т. е. число исполняемых файлов не ограничено;
- Enable package checkpoints** (Включить контрольные точки для пакета) — контрольные точки (информация о них записывается в файл) позволяют продолжить выполнение пакета с определенной задачи (если возникла ошибка). Однако любая задача (или контейнер, например, **ForEach Loop**) считается атомарной единицей пакета. Если при выполнении, например, задачи Data Flow Task возникла ошибка, то выполнение пакета можно продолжить, только начав эту задачу заново — посередине выполнения задачи остановиться нельзя.

Контрольные точки можно настроить как из свойств пакета (группа свойств **Checkpoints** (Контрольные точки)), так и из одноименной вкладки окна утилиты dtexecui (в этом случае настройка контрольных точек будет действительна только для этого запуска). В любом случае вам потребуется указать, будете ли вы использовать контрольные точки, имя текстового файла для хранения контрольных точек и параметры перезапуска пакета, если в файле контрольных точек сохранилась информация о предыдущем запуске с ошибками.

На вкладке **Reporting** (Отчеты) утилиты dtexecui вы можете выбрать набор и параметры информационных сообщений, которые будут выводиться в окно выполнения dtexecui или просто в стандартную консоль dtexec.

На вкладке **Logging** (Протоколирование) можно настроить свои параметры протоколирования работы пакета, если те параметры протоколирования, которые были определены для пакета при его создании, по каким-то причинам вас не устраивают. Эти параметры будут использованы только для текущего запуска пакета.

Вкладка **Set Values** (Настроить значения) предназначена для изменения свойств элементов пакета в ходе его выполнения (фактически это альтернатива конфигурациям). Вам потребуется указать путь для свойства и новое значение. Путь для свойства проще всего взять из автоматически сгенерированного файла конфигурации в формате XML (*см. разд. 10.25*).

Вкладка **Verification** (Проверка) позволяет определить дополнительные проверки, цель которых — обеспечить защиту от неверных версий пакетов. На этой вкладке вы можете определить требование цифровой подписи и проверку версий и идентификаторов пакета.

На последней вкладке, которая называется **Command Line** (Командная строка), находится итог всех настроек, которые были произведены на предыдущих вкладках — сгенерированная команда строка для утилиты dtexec. Вы можете скопировать эту командную строку или произвести ее дополнительное редактирование прямо в этом окне. Конечно, вы можете и сразу запустить пакет на выполнение, нажав кнопку **Execute**.

Очень часто в реальной работе требуется запланировать запуск пакетов SSIS по расписанию, например, в ночное время. Обычно для этого используются два способа. Первый способ — воспользоваться средствами SQL Server Agent. Необходимо будет создать задание (*job*) SQL Server Agent и создать в нем этап (*step*) типа SQL Server Integration Service Package. Другая возможность — сгенерировать командную строку для утилиты dtexec и запланировать ее для выполнения по расписанию стандартными средствами планировщика операционной системы.

Задание для самостоятельной работы 10.1. Создание пакетов SSIS для переноса данных

Задание:

1. Создайте средствами Microsoft Access в корневом каталоге диска C: пустую базу данных Person.

2. Создайте пакет SSIS, который должен копировать все таблицы из схемы Person базы данных AdventureWorks на вашем сервере SQL Server 2005 со всеми данными в базу данных Person. Этот пакет должен быть сохранен в файл C:\CopyPerson.dtsx. Зашифруйте его при помощи пароля P@ssw0rd. Убедитесь, что этот пакет работает.
3. Измените созданный вами пакет таким образом, чтобы при копировании информации в столбце Phone таблицы Person.Contact удалялись все дефисы, пробелы и скобки. Например, адрес вида "1 11 500 555-015" после преобразования должен выглядеть как "111500555015".
4. Создайте на диске С: файл CopyPerson.bat с командной строкой на запуск пакета CopyPerson.dtsx. Подробный протокол выполнения должен записываться в текстовый файл C:\CopyPerson.log. Еще раз удалите все таблицы из файла Person.mdb и убедитесь, что созданный вами пакетный файл работает в соответствии с заданием.

Решение:

К пункту 1 задания — создание пустой базы данных Microsoft Access:

1. Запустите Microsoft Access (меню **Пуск | Программы | Microsoft Office | Microsoft Office Access**) и в меню **Файл** выберите **Создать**.
2. В окне **Создание файла** щелкните по ссылке **Новая база данных**. В открывшемся окне **Файл** новой базы данных в поле **Имя файла** введите C:\Person.mdb. После этого закройте Microsoft Access.

К пункту 2 задания — создание пакета SSIS:

1. Выполните в командной строке команду DTSWizard. Откроется мастер SQL Server Import and Export Wizard. На первом экране мастера нажмите кнопку **Next**.
2. На экране **Choose a Data Source** в списке **Data Source** (Источник данных) выберите **SQL Native Client**. В поле **Server Name** (Имя сервера) выберите сервер **имя_вашего_сервера\SQL2005** и в списке **Database** (База данных) выберите базу данных **AdventureWorks**. Нажмите кнопку **Next**.
3. На экране **Choose a Destination** в списке **Destination** (Назначение) выберите **Microsoft Access**. Затем в списке **File name** (Имя файла) введите или выберите **C:\Person.mdb**. Нажмите кнопку **Advanced**, а затем — **Test Connection**, чтобы убедиться, что соединение установлено успешно. Нажмите кнопку **OK**, а затем — **Next**.
4. На экране **Specify Table Copy or Query** оставьте переключатель в положении **Copy data from one or more tables or views** (Копировать данные из одной или более таблиц или представлений).

5. На следующем экране **Select Source Tables and Views** установите флажки напротив всех таблиц схемы Person (их должно быть 6).
6. На экране **Save and Execute Package** оставьте флажок **Execute Immediately** и установите флажок **Save SSIS Package**. Затем переставьте переключатель в положение **File System** (Файловая система) и нажмите кнопку **Next**.
7. В открывшемся окне **Package Protection Level** (Уровень защиты пакета) выберите тип защиты **Encrypt all data with password** (Зашифровать все данные паролем) и введите пароль P@ssw0rd.
8. На экране **Save SSIS Package** в поле **Name** введите CopyPerson, а в поле **File Name** — C:\CopyPerson.dtsx. Нажмите кнопку **Next**, а затем — **Finish**. Убедитесь на экране протокола выполнения, что все операции выполнены успешно, а затем нажмите кнопку **Close** (Закрыть).
9. Откройте созданный вами файл C:\Person.mdb в Microsoft Access и убедитесь, что данные скопированы.

К пункту 3 задания — применение преобразований Data Flow Task:

1. Запустите Business Intelligence Development Studio (меню **Пуск | Программы | Microsoft SQL Server 2005 | SQL Server Business Intelligence Development Studio**). В меню **File** выберите **Open | File** и в открывшемся окне выберите файл C:\CopyPerson.dtsx. В ответ на приглашение ввести пароль введите P@ssw0rd и нажмите кнопку **OK**. Если возникнет предупреждение "Document contains one or more extremely long lines of text", проигнорируйте его, нажав кнопку **Да**. Пакет будет открыт в SSIS Designer.
2. Перейдите на вкладку **Data Flow** редактора SSIS Designer и найдите преобразование для таблицы Contact. Оно может выглядеть, например, так, как представлено на рис. 10.17.
3. В меню **View** выберите **Toolbox**, чтобы открылось окно **Toolbox**. Затем перетащите в окно SSIS Designer на вкладку **Data Flow** элемент **Script Component** из раздела **Data Flow Transformations**. В открывшемся окне **Select Script Component Type** переставьте переключатель в положение **Transformation** и нажмите кнопку **OK**.
4. Щелкните правой кнопкой мыши по зеленою стрелке, которая идет от преобразования **Data Conversion 3** и в контекстном меню выберите **Delete** (Удалить). Затем щелкните мышью по элементу **Data Conversion 3**, чтобы его выделить. Перетащите зеленую стрелку, которая выходит из этого элемента, на созданный вами элемент **Script Component**. Затем точно так же выделите элемент **Script Component** и перетащите выходящую из него

зеленую стрелку на элемент Destination 2 – Contact. В итоге конфигурация должна выглядеть так, как представлено на рис. 10.18.

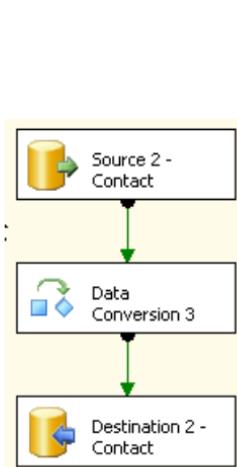


Рис. 10.17. Преобразование для таблицы Contact в окне SSIS Designer

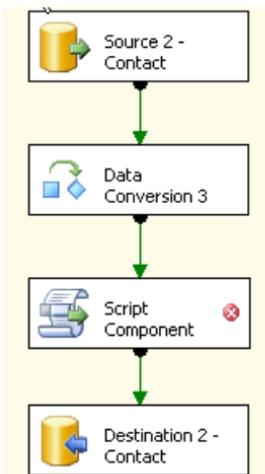


Рис. 10.18. Схема прохождения данных после добавления преобразования Script Component

5. Щелкните правой кнопкой мыши по созданному вами элементу Script Component и в контекстном меню выберите **Edit**.
 6. На вкладке **Input Columns** установите флажок напротив столбца Phone и в списке в нижней части вкладки в столбце **Usage Type** (Тип использования) для этого столбца выберите **ReadWrite**.
 7. Перейдите на вкладку **Script** и нажмите кнопку **Design Script**.
 8. В открывшемся окне редактора кода на месте комментария "Add your code here" введите:
- ```
Row.Phone = Replace(Replace(Row.Phone, "-", ""), " ", "")
```
- Закройте окно редактора кода с сохранением внесенных изменений и нажмите кнопку **OK**, чтобы закрыть свойства преобразования **Script Component**.
9. Сохраните измененный вами пакет CopyPerson.dtsx и закройте окно SSIS Designer.
  10. Для проверки работоспособности созданного вами пакета откройте уже созданный файл Person.mdb в Microsoft Access и удалите в нем все таблицы, а потом закройте его.
  11. Щелкните правой кнопкой мыши по измененному вами пакету CopyPerson.dtsx в Проводнике Windows и в контекстном меню выберите

**Открыть.** Пакет будет открыт при помощи программы Execute Package Utility.

12. Нажмите кнопку **Execute** в окне этой программы и в ответ на приглашение введите пароль P@ssw0rd. Просмотрите информацию о выполнении пакета. После окончания выполнения откройте средствами Microsoft Access файл Person.mdb и просмотрите созданную в нем таблицу Contact. В столбце Phone не должно быть ни пробелов, ни дефисов.

К пункту 4 задания — создание пакетного файла с зашифрованной командной строкой:

1. Команда в пакетном файле C:\CopyPerson.bat может выглядеть так:

```
DTEXEC /F "C:\CopyPerson.dtsx" /De P@ssw0rd /Rep V > C:\CopyPerson.Log
```



## ГЛАВА 11

# Мониторинг и оптимизация производительности SQL Server 2005

### 11.1. Введение в мониторинг работы и оптимизацию производительности SQL Server 2005

Мониторинг работы сервера SQL Server — это одна из ежедневных обязанностей администратора. Администраторы-профессионалы всегда стремятся владеть наиболее полной информацией о всем происходящем на SQL Server, а не дожидаться, пока о возникших проблемах их известят пользователи.

В этой главе будут рассмотрены возможности мониторинга активности пользователей, а также мониторинг событий, происходящих на сервере, и мониторинг производительности работы сервера.

Производительность работы сервера SQL Server — это очень большой вопрос для большинства предприятий. За восемь лет проведения курсов Microsoft по SQL Server разных версий автор не встретил еще ни одного слушателя, который бы работал с большой базой данных SQL Server и при этом был бы доволен производительностью сервера. Чаще всего причина заключается в том, что разработчики приложений на основе SQL Server практически не уделяют внимания вопросам, связанным с производительностью. Это относится к самым разным приложениям. Проблемы с производительностью есть и у созданных "своими силами" приложений для конкретного предприятия, и у распространенных решений (клиент-серверный вариант 1С), у приложений, созданных разработчиком в свободное время как "халтура", и у приложений от крупнейших разработчиков (например, Axapta от Microsoft). Как правило, добиться от разработчиков, чтобы они уделили достаточное внимание повы-

шению производительности, очень сложно. Поэтому оптимизацией задачи часто приходится заниматься тем, кто непосредственно отвечает за ее работу, т. е. администраторам. Надо сказать, что обычно резервы оптимизации имеются у каждого приложения.

В этой главе будут рассмотрены возможности мониторинга производительности приложений, работающих с SQL Server, и некоторые приемы оптимизации производительности.

## 11.2. Мониторинг активности пользователей

### 11.2.1. Применение Activity Monitor

Очень часто возникает необходимость выяснить, какие пользователи в данный момент подключены к SQL Server, с какими объектами они работают и какие операции выполняют. Самый простой способ для этого — воспользоваться консолью Activity Monitor, которая встроена в SQL Server Management Studio. Найти ее можно в контейнере **Management** (Управление) в **Object Explorer**.

В консоли Activity Monitor предусмотрено три вкладки:

- на вкладке **Process Info** (Информация о процессах) вы можете просмотреть информацию об установленных к SQL Server подключениях. Для каждого из процессов можно просмотреть последнюю выполненную команду (пункт **Details** (Подробно) контекстного меню) и принудительно закрыть это подключение (команда `KILL`). Возможности отправить пользователю предупреждающее сообщение, которая была в SQL Server 2000, больше нет.

Обратите внимание, что на самом деле установленных подключений намного больше. Просто все системные подключения по умолчанию скрыты. Чтобы просмотреть все подключения, нужно нажать кнопку **Filter** (Фильтр) и снять флажок **Apply Filter** (Применить фильтр) (при помощи этой кнопки можно также настроить свой фильтр).

Администраторам рекомендуется заглядывать на эту вкладку с определенной периодичностью, например, раз в день. Главное, что вас должно интересовать, — это информация в столбцах **Login Time** (время входа на сервер) и **Last Batch** (последний выполненный пакет). Для удобства можно отсортировать записи по этим столбцам. Если какое-то пользовательское соединение живет уже несколько дней, а последняя команда была выполнена достаточно давно, то, вполне возможно, что это "мертвый процесс". Такие процессы могут оставаться, когда приложения пользователей завершили работу некорректно. Задача администратора — удалить мертвые процессы.

вые процессы, чтобы освободить системные ресурсы (а возможно, и снять блокировки с объектов базы данных);

- на вкладке **Locks by Process** (Блокировки по процессам) показаны все блокировки, которые применены к объектам баз данных определенным процессом. К сожалению, здесь можно просмотреть информацию только для одного процесса, выбрав его (но номеру) в верхней части вкладки. Вкладкой **Locks by Process** приходится пользоваться редко, обычно только для целей диагностики каких-либо проблем;
- на вкладке **Locks by Object** (Блокировки по объектам) вы можете просмотреть блокировки, которые применены к конкретному объекту. Иногда возникает ситуация, когда какой-то пользователь (обычно даже не подозревая об этом) в ходе выполнения какой-либо операции в своем приложении накладывает блокировку на объект базы данных и не снимает ее (например, по причине сбоя приложения). Как правило, о такой ситуации администратора оповещают телефонные звонки других пользователей. Задача администратора — найти на этой вкладке нужный объект, определить номер процесса, который владеет блокировкой, и по этому номеру на вкладке **Process Info** вручную завершить работу процесса.

## 11.2.2. Другие средства мониторинга активности пользователей

Просмотреть, кто сейчас подключен к SQL Server и какие блокировки применены к какому объекту, можно и другими способами.

Первый способ — использование системных хранимых процедур. Получить информацию о том, какие соединения в настоящее время установлены с SQL Server, можно при помощи процедуры `sp_who`. Для этой процедуры предусмотрено два варианта запуска. Если выполнить ее без параметров, то она вернет информацию о всех процессах. Если передать ей идентификатор процесса (`spid`), например, `exec sp_who 55`, то она вернет информацию только по этому процессу.

У этой хранимой процедуры есть недокументированный вариант — `sp_who2`. Эта хранимая процедура запускается на выполнение точно так же, но возвращает более подробную информацию.

Просмотреть информацию о блокировках, которые применены к объектам определенным процессом, можно при помощи хранимой процедуры `sp_lock`. Если запустить ее без параметров, она вернет информацию о блокировках для всех процессов. Если передать ей идентификатор процесса, то будет предоставлена информация о блокировках только для этого процесса. Хранимой процедуре `sp_lock` можно передать сразу несколько идентификаторов, разделив их запятыми.

В случае необходимости можно принудительно закрыть соединение при помощи команды Transact-SQL `KILL`, передав ей идентификатор процесса, например, `KILL 51`.

Новая возможность SQL Server 2005 — динамические представления. Это специальный интерфейс, при помощи которого можно получать информацию о текущей работе сервера или базы данных. К ним можно обращаться из команд Transact-SQL, как к обычным представлениям. Информация в этих представлениях динамически меняется, отражая изменения в работе сервера или базы данных.

Информацию о подключениях пользователей можно просмотреть при помощи динамических представлений `sys.dm_exec_sessions` и `sys.dm_exec_connections`. В обоих случаях возвращаемой записи будет соответствовать одно соединение, но набор столбцов у этих представлений разный. Представление `sys.dm_exec_sessions` ориентировано на предоставление информации о сеансах пользователей (фактически возвращается та же информация, что и в Activity Monitor), а `sys.dm_exec_connections` возвращает информацию о сетевых параметрах подключений пользователей (тип сетевой библиотеки, размер пакета, номера портов и т. п.).

Информацию о блокировках, которые применены к объектам на SQL Server, можно просмотреть при помощи динамического представления `sys.dm_tran_locks`. Информация о сеансе пользователя, который применил данную блокировку, можно найти в столбце `request_session_id` этого представления.

### 11.2.3. Работа с профилировщиком

Одно из самых полезных средств мониторинга активности пользователей — это *профилировщик (Profiler)*. При помощи этого программного средства можно узнать, какие команды в настоящее время выполняет сервер SQL Server. Необходимость в применении профилировщика возникает очень часто. Вот несколько стандартных ситуаций, когда без него обойтись бывает очень сложно:

- вы хотите проанализировать работу приложения и посмотреть, какие команды оно выполняет на сервере. Эта информация может пригодиться:
  - чтобы понять, с какими таблицами в базе данных работает это приложение при выполнении определенных операций. Очень часто на предприятиях возникает необходимость создать отчеты по форме, которая не предусмотрена приложением, а разработчики предоставляют подробную информацию о структуре базы данных редко;
  - чтобы выяснить, насколько оптимальные с точки зрения производительности запросы передает на сервер приложение. На практике при

использовании профилировщика часто можно выявить совсем неоптимальные запросы, например, когда фильтрация или сортировка данных выполняется на клиенте;

- чтобы понять, при выполнении какой команды Transact-SQL из приложения на сервере возникает ошибка;

- для сбора информации о пользовательской активности в течение продолжительного промежутка времени (например, можно собрать все запросы, которые передавались на сервер определенным приложением в течение рабочего дня). Затем собранную информацию можно проанализировать вручную или передать программе Database Tuning Advisor для проведения автоматизированного анализа;
- для проведения мониторинга работы сервера в режиме реального времени. Например, если работа сервера вдруг замедлилась, в окне профилировщика можно просмотреть, какие команды в данный момент на нем выполняются.

В SQL Server 2005 у профилировщика появилось много нового:

- добавилась возможность профилировки Analysis Services. Теперь вы можете просматривать команды и события не только для обычных баз данных, но и для баз данных OLAP;
- появилась профилировка событий Integration Services. Теперь вы можете при помощи профилировщика отслеживать ход выполнения новых пакетов DTS;
- появилась возможность при записи информации выполнения команды записывать также показания счетчиков из Системного монитора;
- в профилировщик добавлено много новых событий и источников информации, которые могут выбираться для записи в файл трассировки. Определение того, что нужно записывать в файл трассировки, теперь можно сохранять в формате XML;
- в формате XML теперь можно сохранять и результаты трассировки (возможность записи в форматы ANSI, OEM, UNICODE также сохранена);
- в формате XML можно сохранять даже планы выполнения команд Transact-SQL, перехваченных профилировщиком. Затем эти планы можно открыть в SQL Server Management Studio для дальнейшего анализа;
- появилась возможность группировать события прямо в окне профилировщика. С помощью группировки, например, можно очень просто посчитать, сколько раз в течение дня на сервере выполнялась та или иная команда Transact-SQL.

Работа с профилировщиком выглядит очень просто. Это приложение можно запустить из меню **Пуск | Программы | Microsoft SQL Server 2005 | Performance Tools | SQL Server Profiler**. Для того чтобы начать работу, в открывшемся окне профилировщика в меню **File** (Файл) нужно выбрать **New Trace** (Новая трассировка) и подключиться к серверу SQL Server 2005, работу которого вы будете отслеживать. Под словом "трассировка" подразумевается сеанс сбора информации о работе SQL Server 2005. Однако перед тем, как приступить к сбору информации, нужно настроить параметры этого сеанса. Эта настройка производится в окне **Trace Properties** (Свойства трассировки), которое открывается автоматически перед началом сеанса трассировки (рис. 11.1).

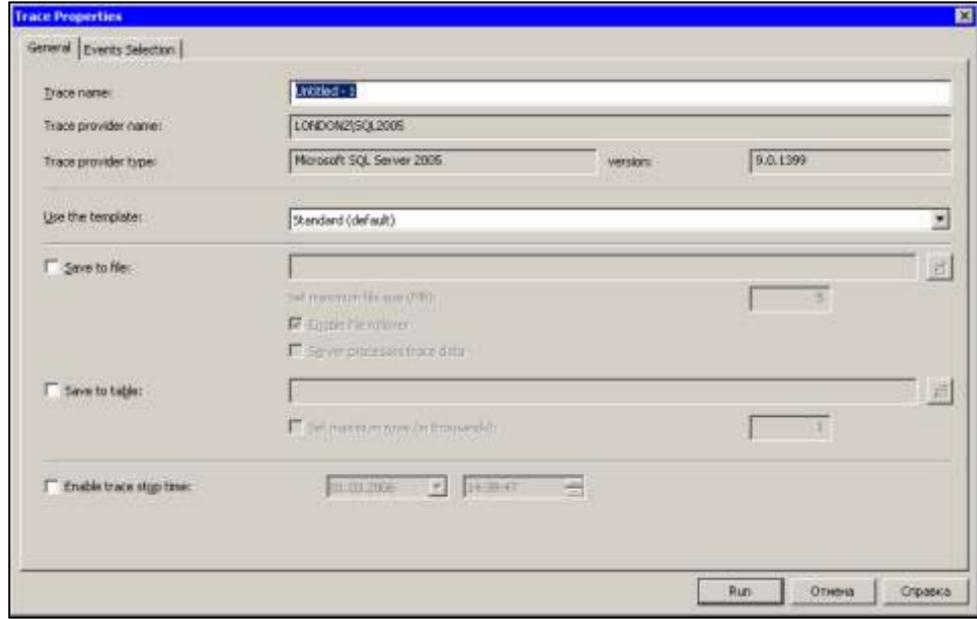


Рис. 11.1. Настройка параметров сеанса трассировки

На вкладке **General** (Общие) в списке **Use the template** (Использовать шаблон) вы можете выбрать наиболее подходящий шаблон для сбора информации в рамках вашего сеанса. В принципе, можно и не обращать внимание на настройки шаблона, а вручную определить параметры сбора информации (при помощи соседней вкладки **Events Selection** (Выбор событий)). Однако указание правильного шаблона поможет сэкономить время и избежать ошибок. Поэтому на шаблонах остановимся подробнее.

Шаблон — это сохраненные в специальном файле с расширением tdf настройки сеанса трассировки. Работа с шаблонами (добавление новых, изменение существующих, удаление) производится в окне **Template Manager**.

нение существующих, импорт и экспорт отчетов в другие каталоги) производится при помощи меню **File | Templates** (Файл | Шаблоны) в SQL Server Profiler. Изначально в вашем распоряжении есть восемь шаблонов:

- **Standard (default)** — как понятно из названия, этот шаблон подходит для большинства ситуаций и поэтому выбирается по умолчанию. Он позволяет отслеживать все запускаемые на выполнение хранимые процедуры и команды Transact-SQL;
- **SP\_Counts** — собирается информация о запускаемых на выполнение хранимых процедурах и функциях. При этом информация в окне профилировщика сортируется (в терминологии профилировщика — группируется) по именам хранимых процедур;
- **TSQL** — собирается информация о всех командах Transact-SQL, запускаемых на выполнение на сервере. Кроме кода команд, записывается также информация об идентификаторах пользовательских процессов и времени запуска. Обычно этот шаблон используется для мониторинга команд, передаваемых на сервер приложением;
- **TSQL\_Duration** — почти то же самое, что и предыдущий шаблон, но вместо записи информации о времени запуска команды Transact-SQL записывается время, которое потребовалось на ее выполнение. Обычно этот шаблон используется для мониторинга производительности работы сервера "вручную";
- **TSQL\_Grouped** — кроме информации о коде команды Transact-SQL и времени ее запуска, записывается информация о имени приложения, учетной записи пользователя в операционной системе и логине пользователя, который был использован для подключения. При этом записи группируются по логину. Обычно этот шаблон используется в тех ситуациях, когда вы хотите отследить активность конкретного приложения;
- **TSQL\_Replay** — будет записываться максимально подробная информация о выполняемых командах Transact-SQL. Потом эту информацию можно использовать для того, чтобы с максимальной точностью воспроизвести нагрузку на сервер. Обычно этот шаблон применяется для записи набора команд, который будет потом использоваться для тестирования разных настроек сервера с точки зрения производительности;
- **TSQL\_SPs** — кроме записи информации о начале запуска всей хранимой процедуры (событие SP:Starting), этот вариант трассировки записывает также информацию о выполнении каждой команды данной хранимой процедуры (событие SP:StmtStarting). Такой шаблон обычно используется для мониторинга работы сложных хранимых процедур;

□ **Tuning** — этот шаблон предназначен для записи информации, наиболее подходящей для передачи Database Tuning Advisor. Про работу с этим средством автоматизированного анализа и оптимизации производительности будет рассказано в разд. 11.5.5.

Как уже говорилось, совсем необязательно ограничиваться только набором готовых шаблонов. Можно использовать свои параметры сеанса трассировки, настроив их на вкладке **Events Selection**. В таблице на этой вкладке вы должны выбрать требуемые события (в строках) и информацию (в столбцах), которая будет для них записываться. Обратите внимание, что по умолчанию видна только небольшая часть доступных строк и столбцов. Чтобы включить отображение всех строк и столбцов, нужно установить флагки **Show All Events** (Показать все события) и **Show All Columns** (Показать все столбцы).

Очень часто бывает так, что нужно отслеживать только действия, выполняемые в определенной базе данных, или определенным приложением, или определенным пользователем, или выбрать все эти условия одновременно. Фильтры на сбор информации можно настроить, нажав кнопку **Column Filters** (Фильтры столбцов) на вкладке **Events Selection**. Для каждого столбца можно настроить запись только определенных значений (**Like**) или запрет записи определенных значений (**Not Like**). По умолчанию настроен единственный фильтр — **Not Like** для столбца **ApplicationName**. Он заставляет игнорировать все события приложения SQL Server Profiler, т. е. все события, относящиеся к самому процессу сбора информации трассировки. Этот фильтр лучше не удалять, потому что в противном случае может возникнуть положительная обратная связь с бесконечной записью информации.

При помощи еще одной кнопки **Organize Columns** (Организовать столбцы), которая расположена на вкладке **Events Selection**, можно настроить порядок столбцов для отображения или записи в профилировщике. Обратите внимание на раздел **Group** (Группа) в этом списке. Для тех столбцов, которые в него помещены, будет автоматически производиться группировка. Если вы поместите в этот раздел только один столбец, то при просмотре у вас появится возможность использовать очень удобный режим **Aggregated View** (Агрегированное представление) (когда информация автоматически сгруппирована, например, по базе данных, по приложению, имени пользователя и т. п., и записи для нужной базы данных, приложения или пользователя можно раскрывать и сворачивать).

После того как выбран нужный шаблон или настроен свой собственный набор событий для протоколирования, вам остается вернуться на вкладку **General** и настроить несколько дополнительных параметров сеанса трассировки.

Информация трассировки может быть запротоколирована в файл. Этот файл можно использовать в разных ситуациях:

- можно передать в качестве источника информации Database Tuning Advisor;
- можно "проиграть" повторно в профилировщике, повторив все записанные команды, например, для оценки производительности при разных настройках сервера;
- можно предъявить разработчикам в подтверждение своих претензий к приложению.

Отметим некоторые моменты, которые связаны с протоколированием сеанса трассировки в файл:

- 5 Мбайт, которыми ограничивается размер файла по умолчанию, это очень мало. При профилировке рабочего сервера этот размер набирается за минуты. Правда, по умолчанию установлен флагок **Enable file rollover** (Включить смену файлов), т. е. после заполнения одного файла автоматически будет создан второй файл, к имени которого добавится номер 1, потом — 2 и т. п., но работать с большим количеством файлов не всегда удобно. Если вы собираете информацию для передачи Database Tuning Advisor, то лучше настроить предельный размер файла в 1 Гбайт (при помощи параметра **Set maximum file size** (Настроить максимальный размер файла) на вкладке **General**). Запись трассировки в файл чаще всего производится с рабочей станции администратора, поэтому место на диске потребуется именно на рабочей станции, а не на сервере;
- параметр **Server processes trace data** (Сервер обрабатывает данные трассировки) можно использовать для увеличения надежности записи информации трассировки. По умолчанию обработкой данных трассировки занимается SQL Server Profiler, и происходит это на том компьютере, на котором он запущен (не обязательно на сервере). Если установить этот флагок, то обработкой информации трассировки будет заниматься сервер. Это гарантирует, что вся информация трассировки будет собрана (при снятом флагке в моменты пиковой нагрузки сервера часть информации может быть пропущена), но увеличит нагрузку на сервер.

Другой вариант записи информации трассировки — запись в таблицу SQL Server. Таблица с нужным набором столбцов будет создана автоматически. Вы можете лишь настроить максимальное количество записей в этой таблице. Обратите внимание, что на этой вкладке максимальное количество записей указывается в тысячах.

Последний параметр на вкладке **General** — **Enable Trace stop time** (Включить время остановки трассировки). Вы можете указать время, когда трассировка будет отключена автоматически. Обычно имеет смысл отключать трассировку перед началом каких-то служебных операций, которые с точки зрения протоколирования вас не интересуют (резервное копирование, массовая загрузка данных, процессинг кубов OLAP и т. п.).

После того, как все параметры трассировки будут настроены, можно нажать на кнопку **Run** (Запустить) на вкладке **General** и приступить к трассировке (рис. 11.2).

The screenshot shows the SQL Server Profiler interface. The main pane displays a table of trace results with columns: EventClass, TextData, ApplicationName, NTUserName, LoginName, CPU, Reads, Writes, Duration. The data includes various RPC and SQL commands. The bottom pane shows a detailed view of a selected trace row, displaying the raw T-SQL code used to generate the event.

| EventClass        | TextData                                 | ApplicationName | NTUserName | LoginName           | CPU | Reads | Writes | Duration |
|-------------------|------------------------------------------|-----------------|------------|---------------------|-----|-------|--------|----------|
| RPCCompleted      | exec sp_executesql N'...                 | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| RPCCompleted      | exec sp_replcmds 500,0,-1,0x5507,0...    | Repl-Logread... | N          | LONDON2\N           | 0   | 14    | 0      | 0        |
| RPCCompleted      | exec sp_replsub_Adjust_Identity          | Repl-Logread... | N          | LONDON2\N           | 0   | 46    | 0      | 0        |
| RPCCompleted      | exec sp_replsub_Adjust_Identity          | Repl-Logread... | N          | LONDON2\N           | 0   | 14    | 0      | 0        |
| RPCCompleted      | exec sp_reset_connection                 | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| SQLBatchStarting  | ...                                      | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| SQLBatchCompleted | ...                                      | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| SQLBatchStarting  | SELECT N'Testing Connection...',         | SQLAgent - A... | N          | LONDON2\N           | 0   | 0     | 0      | 0        |
| SQLBatchCompleted | SELECT N'Testing Connection...',         | SQLAgent - A... | N          | LONDON2\N           | 0   | 0     | 0      | 0        |
| SQLBatchStarting  | EXECUTE msdb.dbo.sp_startagent_get_pe... | SQLAgent - A... | N          | LONDON2\N           | 0   | 0     | 0      | 0        |
| SQLBatchCompleted | EXECUTE msdb.dbo.sp_stopagent_get_pe...  | SQLAgent - A... | N          | LONDON2\N           | 0   | 3     | 0      | 0        |
| RPCCompleted      | exec sp_reset_connection                 | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| RPCCompleted      | exec sp_executesql N'...                 | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| RPCCompleted      | exec sp_replcmds 500,0,-1,0x5507,0...    | Repl-Logread... | N          | LONDON2\N           | 0   | 14    | 0      | 0        |
| RPCCompleted      | exec sp_replsub_Adjust_Identity          | Repl-Logread... | N          | LONDON2\N           | 0   | 46    | 0      | 0        |
| RPCCompleted      | exec sp_replsub_Adjust_Identity          | Repl-Logread... | N          | LONDON2\N           | 0   | 14    | 0      | 0        |
| RPCCompleted      | exec sp_reset_connection                 | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |
| SQLBatchStarting  | ...                                      | Report Server   | SYSTEM     | NT AUTHORITY\SYSTEM | 0   | 0     | 0      | 0        |

```

declare @notifcount int=0
declare @notifid int=0
set @notifcount = 0
set @notifid = 0
update [notifications] with (tablock)
set [notifID] = @notifid,
[notifCount] = @notifcount,
[notifLastTime] = getdate(),
[notifLastTime2] = getdate()
from [notifications]
select top 1 [notifCount] from [notifications] with (tablock) where [notifLastTime] is null or [notifLastTime] < getdate() order by [notifLastTime] asc
if @@trancount > 0
begin
 rollback transaction
end
else
begin
 if @notifcount > 0
 begin
 update [notifications] set [notifID] = @notifid + 1 where [notifID] = @notifid
 end
 else
 begin
 insert into [notifications] ([notifID])
 values (@notifid)
 end
end
select top 3 * from [notifications]
-- NOLOCK option data

```

Рис. 11.2. Просмотр информации в ходе сеанса трассировки

Работа в окне просмотра информации трассировки достаточно очевидна: в верхней части показываются события, которые происходят на сервере, а в нижней части для них приводится подробная информация (например, код команд SQL). Отметим некоторые возможности, доступные в этом окне:

- если на вкладке **Organize Columns** в свойствах шаблона вы выбрали столбцы для группировки, то можно сгруппировать по этим столбцам записи в окне просмотра. Для этой цели в меню **View** (Вид) предусмотрена команда **Grouped View** (Сгруппированное представление);
- если на той же вкладке в свойствах шаблона в список **Group** был помещен только один столбец, то можно использовать еще более удобный режим отображения **Aggregated View** (рис. 11.3). Этот режим включается при помощи команды **Aggregated View** из того же меню **View** и позволяет превратить значения из выбранного вами столбца в узлы дерева, которые

можно сворачивать и разворачивать. Кроме того, для каждого из этих узлов автоматически подсчитывается количество событий.

| ApplicationName                                    | EventClass         | LoginName |
|----------------------------------------------------|--------------------|-----------|
| (2)                                                |                    |           |
| Microsoft SQL Server Management Studio - Query (1) |                    |           |
| Microsoft SQL Server Management Studio - Query     | ExistingConnection | LONDON2\R |
| + Repl-LogReader-0-AdventureWorks-8 (5)            |                    |           |
| + Report Server (4)                                |                    |           |
| SQLAgent - Alert Engine (3)                        |                    |           |
| SQLAgent - Alert Engine                            | ExistingConnection | LONDON2\R |
| SQLAgent - Alert Engine                            | SQL:BatchStarting  | LONDON2\R |
| SQLAgent - Alert Engine                            | SQL:BatchStarting  | LONDON2\R |
| - SQLAgent - Generic Refresher (1)                 |                    |           |
| SQLAgent - Generic Refresher                       | ExistingConnection | LONDON2\R |
| + SQLAgent - Job invocation engine (1)             |                    |           |

Рис. 11.3. Режим отображения Aggregated View

- в профилировщике можно отобразить не только те события, которые были пойманы только что, но также сохраненные файлы и таблицы трассировки. Кроме того, вы можете открывать обычные скрипты SQL Server с командами Transact-SQL. Информация из этих файлов или таблиц может быть использована для того, чтобы повторить запротоколированные операции. Для этой цели предназначены команды меню **Replay** (Повторить);
- в профилировщике SQL Server 2005 появилась новая возможность — связывание информации трассировки с показателями счетчиков производительности Системного монитора. Для того чтобы воспользоваться этой возможностью, нужно:
  - определить сеанс трассировки, в ходе которого обязательно должна записываться информация для столбцов StartTime И EndTime;
  - запустить сеанс трассировки с записью информации в файл или таблицу. Одновременно с ним собрать в файл протокол показаний счетчиков Performance Monitor;
  - открыть собранную информацию из файла трассировки в профилировщике, а затем воспользоваться командой **Import Performance Data** (Импортировать данные производительности) из меню **File**.

В SQL Server 2005 предусмотрен заменитель для профилировщика. Это хранимые процедуры трассировки. Их функциональные возможности практически идентичны возможностям профилировщика. Например, вы можете также выбрать события для трассировки и записать их в текстовый файл. Главное отличие заключается в том, что все настройки придется производить из кода Transact-SQL.

Работать с хранимыми процедурами трассировки сложнее и менее удобно, чем с профилировщиком, а дополнительных возможностей они не предоставляют. Поэтому подробно рассматривать их не будем. Приведем только список таких хранимых процедур с краткой характеристикой:

- ❑ `sp_trace_create` — позволяет настроить параметры сеанса трассировки;
- ❑ `sp_trace_setevent` — позволяет выбрать для созданного сеанса трассировки требуемые события;
- ❑ `sp_trace_setfilter` — позволяет настроить фильтр для сбора информации трассировки;
- ❑ `sp_trace_setstatus` — позволяет запустить трассировку, остановить ее или удалить созданное хранимой процедурой `sp_trace_create` текущее определение сеанса;
- ❑ `sp_trace_generateevent` — позволяет сгенерировать пользовательское событие, которое будет перехвачено в ходе трассировки.

## **Задание для самостоятельной работы 11.1. Сбор информации о запросах, выполняемых приложением**

### **Ситуация:**

На вашем предприятии к рабочей базе данных постоянно обращаются пакеты SSIS, которые копируют информацию в другие базы данных. Пакеты SSIS написаны сторонними разработчиками, и доступа к их коду у вас нет. Скорость работы этих пакетов не устраивает ваших сотрудников. Вам нужно выяснить, какие действия выполняют эти пакеты на сервере.

### **Задание:**

Соберите в файл трассировки информацию об операциях, которые выполняет в базе данных AdventureWorks созданный вами в самостоятельной работе 10.1 пакет CopyPerson.dtsx. Информация должна быть собрана при помощи шаблона **Tuning**.

### **Решение:**

1. Запустите профилировщик (меню **Пуск | Программы | Microsoft SQL Server 2005 | Performance Tools | SQL Server Profiler**).
2. В меню **File** выберите **New Trace** и подключитесь к серверу *имя\_вашего\_сервера\SQL2005*. Откроется окно **Trace Properties**.

3. В этом окне в поле **Trace name** (Имя трассировки) введите AdventureWorksTrace, в списке **Use the template** выберите шаблон **Tuning**. Установите флажок **Save to file** (Сохранить в файл) и введите имя файла C:\AdventureWorksTrace.trc. В поле **Set maximum file size** (Установить максимальный размер файла) установите значение 500 Мбайт и снимите флажок **Enable file rollover**. Затем нажмите на кнопку **Run**.
4. Удалите средствами Microsoft Access все таблицы из файла Person.mdb и запустите на выполнение созданный вами в самостоятельной работе 10.1 пакет CopyPerson.dtsx (для этого можно использовать пакетный файл CopyPerson.bat). Просмотрите информацию о выполняемых командах в окне профилировщика. После окончания копирования воспользуйтесь командой **Stop Trace** (Остановить трассировку) в меню **File**.

#### 11.2.4. Применение триггеров DDL

Все средства, которые были рассмотрены ранее, — Activity Monitor, системные хранимые процедуры, профилировщик — подразумевают, что администратор должен сидеть перед монитором и просматривать информацию, которую они возвращают. Однако очень часто у администратора есть другие дела, которыми он должен заниматься, и непрерывно осуществлять мониторинг работы сервера он не может. В этой ситуации было бы очень удобным применение средства, которое оповещало бы администратора о выполнении на сервере определенных действий.

Первое средство, которое может использоваться для непрерывного мониторинга сервера и уведомления администратора, — триггеры DDL. Это новая возможность сервера SQL Server 2005. В предыдущих версиях SQL Server триггеров DDL не было.

DDL расшифровывается как Data Definition Language (Язык определения данных). Фактически это поднабор команд Transact-SQL, используемых для создания, изменения и удаления объектов в базе данных. Соответственно, при помощи триггеров DDL можно отслеживать выполнение различных операций, которые производятся с объектами базы данных, например, удаление таблицы, создание нового индекса, изменение определения хранимой процедуры и т. п. Триггеры DDL обычно используются для мониторинга операций пользователей и для уведомления администратора о важных событиях, связанных с выполнением команд DDL.

Кроме триггеров DDL, для мониторинга можно использовать и обычные триггеры. Они срабатывают при изменении данных в таблице (вставке новой записи, изменении или удалении существующей). Если у вас есть какая-то таблица особой важности, и администратор должен немедленно уведомляться о всех вносимых в нее изменениях, то вы вполне можете настроить триггеры

ры на эту таблицу и поместить в них, например, команды на отправку сообщения по электронной почте средствами Database Mail. Однако применение обычных триггеров относится скорее к вопросам разработки баз данных, чем к администрированию, и поэтому здесь не рассматривается.

Триггеры DDL создаются для определенной команды Transact-SQL. При этом набор команд, для которых можно определить триггеры DDL, очень большой. Вот лишь некоторые из этих команд:

- CREATE DATABASE, ALTER DATABASE, DROP DATABASE;
- CREATE LOGIN, ALTER LOGIN, DROP LOGIN;
- CREATE USER, ALTER USER, DROP USER;
- CREATE TABLE, ALTER TABLE, DROP TABLE;
- CREATE VIEW, ALTER VIEW, DROP VIEW;
- CREATE PROCEDURE, ALTER PROCEDURE, DROP PROCEDURE и т. п.

Обратите внимание, что когда вы указываете конкретную команду при создании триггера, между частями команды должен ставиться знак подчеркивания, например, `DROP_TABLE`.

В самом простом варианте триггер DDL может выглядеть так:

```
CREATE TRIGGER DDLTrigger1 ON DATABASE FOR DROP_TABLE AS
PRINT 'Таблица была удалена';
```

Если выполнить эту команду в текущей базе данных, то при любом удалении таблицы в этой базе будет выдано сообщение 'Таблица была удалена'.

Конечно, в реальной работе вы будете использовать более функциональные триггеры, например, триггеры, отправляющие уведомления по электронной почте при помощи хранимых процедур Database Mail.

Во многих случаях вам потребуется получить в триггере более подробную информацию о сути происходящих событий (например, чтобы отправить эту информацию по электронной почте). Для этого удобнее всего использовать специальную функцию `EVENTDATA()`. Например, чтобы получить информацию о конкретном запросе (с именем таблицы), который произвел ее удаление, можно использовать код вида:

```
CREATE TRIGGER DDLTrigger1 ON DATABASE FOR DROP_TABLE AS
SELECT EVENTDATA().value(
'(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]', 'nvarchar(max)');
```

Триггеры считаются частью транзакции, которая их вызывала, поэтому в триггеры можно помещать команды на откат транзакций. Например, при помощи такого триггера можно вообще запретить удалять таблицы в данной базе данных:

```
CREATE TRIGGER DDLTrigger1 ON DATABASE FOR DROP_TABLE AS
PRINT 'Удалять таблицы в этой базе данных запрещено!';
ROLLBACK;
```

Если триггер DDL нужно на время отключить, можно воспользоваться командой `DISABLE TRIGGER`, например:

```
DISABLE TRIGGER DDLTrigger1 ON DATABASE;
```

Удаление триггера DDL производится похожей командой:

```
DROP TRIGGER DDLTrigger1 ON DATABASE;
```

## 11.2.5. Другие средства мониторинга активности пользователей и уведомления о событиях

В SQL Server 2005 предусмотрены и другие возможности уведомления администратора о событиях, происходящих на сервере.

Первая возможность — использование предупреждений SQL Server Agent. Это очень простое в настройке, удобное и надежное средство (см. разд. 8.1.7). В основном предупреждения предназначены для оповещения администратора о системных проблемах сервера, но их можно использовать и для мониторинга действий пользователя. Для этого обычно применяются пользовательские ошибки, которые генерируются при помощи оператора `RAISEERROR`. Сам этот оператор можно поместить в триггеры DDL или обычные триггеры.

Вторая возможность — использование постоянно работающего скрипта WMI, который обращается к поставщику WMI Provider for Server Events (см. разд. 9.4.8). Преимуществом этого подхода является то, что вы не зависите от работы сервера при отправке уведомлений. Для оповещения администратора можно использовать, например, средства объектной модели CDO, предназначенной для работы с электронной почтой.

Наконец, еще одна возможность, которая появилась только в SQL Server 2005, — применение уведомлений о событиях (*event notifications*). После настройки уведомлений о событиях информация об интересующих администратора событиях (к ним относятся выполнение команд DDL и события трассировки) передается в очередь программного модуля Service Broker в виде файлов XML.

Надо сказать, что средствами уведомлений о событиях отслеживаются те же события, что и триггерами DDL, и скриптами WMI Provider for Server Events. К преимуществам уведомлений по сравнению с триггерами DDL можно отнести асинхронный режим работы: сообщение отправляется в очередь Service Broker, не мешая выполнению текущих команд. Однако уведомления о событиях

тиях — это средство скорее для разработчиков, чем для администраторов. Для работы с ними, кроме создания самих уведомлений, необходимо также создавать программный код, который будет извлекать эти сообщения из очереди Service Broker и разбирать код XML этих сообщений. Поэтому работа с уведомлениями о событиях в этой книге рассматриваться не будет.

## 11.3. Журналы SQL Server 2005

В предыдущем разделе были рассмотрены средства мониторинга действий пользователей (и приложений) на SQL Server. Однако кроме мониторинга действий пользователей, администраторам необходимо отслеживать и события, связанные с работой сервера SQL Server 2005. Главная цель при этом — как можно быстрее обнаруживать проблемы, возникающие на сервере, и оперативно реагировать на них.

Самое простое средство мониторинга работы SQL Server 2005 — журналы сервера. Считается, что рабочий день администратора на предприятии должен начинаться с просмотра журналов событий на всех серверах. Для SQL Server 2005 ему придется заглянуть в четыре журнала: журнал событий самого SQL Server 2005, журнал событий SQL Server Agent, журнал событий операционной системы Windows и журнал событий приложений Windows.

Журналы можно просматривать разными способами. Самый простой и рекомендованный — использовать просмотрщик, который встроен в SQL Server Management Studio. Запустить его можно из контейнера **Management | SQL Server Logs** (Управление | Журналы SQL Server). В списке журналов нужно щелкнуть правой кнопкой мыши по требуемому журналу и в контекстном меню выбрать **View SQL Server Log** (Просмотреть журнал SQL Server). Откроется окно просмотрщика журналов (рис. 11.4).

Обратите внимание, что при помощи просмотрщика журналов:

- можно просматривать не только журналы SQL Server, но и журналы SQL Server Agent, Windows и Database Mail;
- можно экспорттировать данные из журналов при помощи кнопки **Export** (Экспортировать), в том числе и в очень удобный для загрузки в базу данных формат CSV;
- можно настраивать фильтрацию и производить поиск нужной информации.

Журналы событий SQL Server можно просматривать и "вручную", при помощи любого текстового редактора. По умолчанию они находятся в каталоге C:\Program Files\Microsoft SQL Server\MSSQL\LOG. Там же находятся и журналы SQL Server Agent.

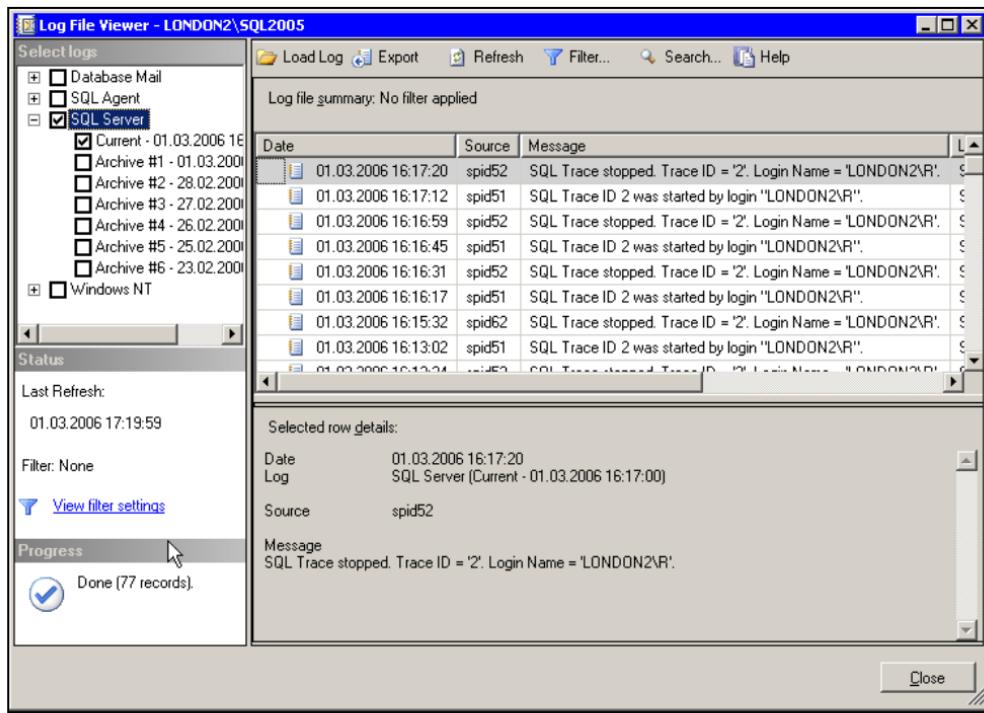


Рис. 11.4. Окно просмотрщика журналов SQL Server Management Studio

Если вам нужен более подробный протокол событий, происходящих на сервере, можно воспользоваться параметром **C2 Audit Tracing**. Его можно установить на вкладке **Security** (Безопасность) свойств сервера в Management Studio. В этом режиме в каталоге **Data** (Данные) для данного экземпляра сервера будут автоматически создаваться текстовые файлы с очень подробной информацией, которая может потребоваться для аудита в соответствии со стандартом безопасности C2.

## 11.4. Мониторинг производительности SQL Server 2005

### 11.4.1. Основы мониторинга производительности

Кроме наблюдения за событиями, происходящими на сервере, и регулярного просмотра журналов, у администраторов SQL Server 2005 есть еще одна важная обязанность — мониторинг производительности работы сервера.

Главная цель мониторинга производительности — обеспечить комфортную работу пользователей с приложением, который использует SQL Server 2005.

При этом нужно не только оценивать текущую производительность, но и прогнозировать развитие тенденций в будущем. Хороший администратор не ждет ситуации, когда пользователи в один голос сообщат руководству, что работать с задачей стало невозможно, а решает проблемы с производительностью еще до того, как они станут мешать работе пользователей.

Конечно, на большинстве предприятий, особенно небольших, никто не занимается мониторингом производительности и не отслеживает тенденции, ограничиваясь прикидками "быстро/не быстро". Однако на больших предприятиях для важных задач такой мониторинг производится обязательно. Да и на любом предприятии при приеме задачи в эксплуатацию стоит задавать вопрос: а что будет через несколько лет, когда в базе данных накопятся десятки гигабайт информации? Обычно намного проще заставить разработчика уделить внимание вопросам, связанным с производительностью, при приеме задачи, чем пытаться сделать что-то через 3—4 года после ввода задачи в эксплуатацию.

Существует большое количество книг и статей, которые посвящены вопросам мониторинга оптимизации производительности SQL Server. В качестве отправной точки для поиска информации можно посоветовать замечательный Web-сайт [www.sql-server-performance.com](http://www.sql-server-performance.com). Это компилитивный Web-сайт, на котором систематизируются и выкладываются статьи, главы из книг и утилиты из самых разных источников.

Есть также специальный учебный курс Microsoft, который посвящен исключительно вопросам производительности SQL Server. Правда, он был создан еще для SQL Server 7.0 (и в настоящее время для заказа в учебных центрах недоступен), но большинство тем в этом курсе не потеряло актуальности и для SQL Server 2005.

Для мониторинга и оптимизации производительности существует своя терминология. С этой терминологией (и общим планом мониторинга производительности) вы познакомитесь в следующем разделе.

## 11.4.2. Терминология и общие принципы мониторинга производительности

Как правило, мониторинг производительности начинается с определения *performance baseline*. Официальный перевод этого термина на русский язык от Microsoft — *эталонный график производительности*, но из такого перевода понять, что это такое, сложно.

Физически, эталонный график производительности — это набор самых важных показателей производительности, который был собран при приеме задачи в эксплуатацию, когда нагрузка на сервер и размер базы данных еще ми-

нимальны. Такая информация служит отправной точкой для дальнейшего анализа.

После того, как показатели эталонного графика производительности собраны, задача администратора — на регулярной основе собирать показатели счетчиков производительности. Такие замеры обычно называются *измерениями производительности при рабочей нагрузке* (*workload performance measurements*). Обычно они производятся раз в несколько месяцев, хотя для очень важных задач они выполняются и чаще. Собранная информация о показателях сравнивается с собранной ранее и с эталонным графиком. Главная задача здесь — определить тенденции в развитии производительности и понять, когда у вас могут возникнуть проблемы в работе пользователей. Проще всего произвести такой анализ, если загрузить собранные данные за разные периоды в Excel.

Если в ходе анализа выяснилось, что в скором времени могут возникнуть проблемы с производительностью, то главная обязанность администратора — решить их заблаговременно, не допуская, чтобы они стали помехой при выполнении пользователями своих обязанностей. Обычно администратор определяет *узкие места* (*bottlenecks*), т. е. те ресурсы, недостаток которых тормозит работу всей системы. Это может быть физический компонент сервера (например, оперативная память или дисковая подсистема), система индексов в базе данных SQL Server и т. п. После выявления узких мест администратор должен принять меры к их устранению.

И еще два термина, которые часто используются в литературе по производительности SQL Server 2005.

Первый термин — *время отклика системы на запросы пользователей* (*response time*). Этот показатель считается субъективным. Главная задача администратора при оптимизации производительности — сделать значение этого показателя приемлемым для пользователей, т. е. добиться того, чтобы пользователям было комфортно работать со своей задачей. Реальные требования пользователей зависят от конкретной задачи. Например, в одном случае пользователи могут потребовать, чтобы информацию о заказчике можно было бы просмотреть во время телефонного разговора с ним, в другом случае они просят, чтобы формирование отчета продолжалось не более минуты и т. п.

Второй термин — *пропускная способность* (*throughput*). Это объективный показатель работы сервера, например, сколько транзакций в секунду он может обработать. Обычно этот показатель используется при сравнении различных физических конфигураций сервера или при оценке вариантов настройки.

После того, как вы определились с основной терминологией, рассмотрим средства, которые можно использовать для мониторинга и оптимизации производительности SQL Server.

### 11.4.3. Средства для мониторинга и анализа производительности

На производительность приложения, работающего с SQL Server, влияют самые разные факторы. Для анализа каждого из таких факторов вам потребуются свои средства.

Наиболее универсальное средство мониторинга и анализа производительности — это Системный монитор (*System Monitor*). В Windows NT 4.0 он назывался Монитором производительности (*Performance Monitor*). Это средство предназначено для работы со счетчиками производительности как для операционной системы, так и для SQL Server. Про работу с Системным монитором подробно рассказывается в следующих разделах. Здесь отметим только некоторые моменты:

- в Windows Server 2003 появилась версия Системного монитора, которая предназначена для работы из командной строки. Это утилита `logman`. Она позволяет выполнять протоколирование показаний счетчиков в файлы на диске или источники данных ODBC точно так же, как и графическая версия Системного монитора. Если вам постоянно приходится выполнять замеры производительности, собирая показатели одного и того же набора счетчиков, то будет удобнее написать пакетный файл с командной строкой на запуск этой утилиты с необходимыми параметрами;
- утилита командной строки `relog`, которая также появилась только в Windows Server 2003, позволяет производить конвертацию файлов с собранными показаниями счетчиков из одного формата Системного монитора в другой (например, из двоичного формата BLG в текстовый CSV);
- еще одна новая утилита Windows Server 2003, которая называется `turperf`, предоставляет возможность для просмотра показаний счетчиков Системного монитора в командной строке;
- существуют утилиты третьих фирм, которые позволяют работать со счетчиками Системного монитора, например HostMonitor и AppManager. Эти утилиты предоставляют более удобный графический интерфейс для просмотра и анализа данных по производительности. Но, с точки зрения автора, те дополнительные возможности, которые дают эти утилиты, не принципиальны. Обычно они предлагают более удобные возможности просмотра тех же счетчиков Системного монитора. Однако вы всегда можете загрузить данные, собранные Системным монитором, в Excel и использо-

вать для просмотра и анализа мощные средства этого приложения Microsoft Office.

При оптимизации производительности всегда имеет смысл ориентироваться на максимальную нагрузку сервера. Обычно проблемы с производительностью возникают именно в такие пиковые часы, когда с системой работает максимальное количество пользователей. Однако часто возникают ситуации, когда администратору приходится оценивать производительность приложения и принимать решение о целесообразности его приобретения до того, как оно развернуто на предприятии. Другая возможная ситуация — нужно принять решение о покупке нового сервера, но не совсем понятно, насколько мощное оборудование потребуется приложению при максимальной нагрузке.

В таких ситуациях может быть очень полезной искусственная имитация нагрузки со стороны пользователей — так называемое *нагрузочное тестирование* (*stress testing*). Средства для проведения нагрузочного тестирования будут рассматриваться в *разд. 11.4.4*.

Еще одно важное средство для анализа производительности работы приложения — профилировщик (SQL Server Profiler). Работа с ним была описана в *разд. 11.2.3*. Это средство позволяет найти и запротоколировать команды, которые передаются на сервер приложением, а также найти запросы, на выполнение которых требуется много времени.

Если вы подозреваете, что работу вашего приложения тормозят проблемы с сетью, то для анализа таких проблем вам потребуется Сетевой монитор (*см. разд. 11.4.10*).

Для анализа и оптимизации системы индексов удобнее всего использовать программное средство Database Tuning Advisor (Index Tuning Wizard в SQL Server 2000). Про работу с ним будет рассказываться в *разд. 11.5.5*.

Для оптимизации запросов к SQL Server очень удобно использовать средства SQL Server Management Studio (*см. разд. 11.5.8*).

## 11.4.4. Нагрузочное тестирование

Очень часто перед администратором встает необходимость протестировать работу сервера под реальной нагрузкой. Например, на предприятии развертывается новое приложение, и не понятно, насколько мощное оборудование ему необходимо. И вам хотелось бы посмотреть, как поведет себя сервер, если с ним будут работать одновременно, например, 100 пользователей, но не понятно, как можно организовать такую проверку.

В такой ситуации на помощь могут прийти средства нагрузочного тестирования (*stress testing utility*). Такие утилиты специально предназначены для того, чтобы имитировать работу большой группы пользователей.

Microsoft рекомендует для нагружочного тестирования SQL Server использовать свои же утилиты. Главный набор таких утилит называется Microsoft SQL Server Support Escalation Services Support Utilities. Его можно бесплатно скачать с сайта Microsoft. Другое название этого набора — RML (от Replay Markup Language — XML-совместимый язык, который используется в этих утилитах). Это название используется чаще, т. к. утилиты скачиваются с сайта Microsoft в виде файла rml.exe и устанавливаются в каталог rml.

В наборе RML есть три главные утилиты: `Ostress`, `Read80Trace` и `ORCA`.

`Ostress` — главное средство для выполнения нагружочного тестирования. Это консольная утилита с большим количеством параметров командной строки. Например, если вы хотите запустить на выполнение на локальном сервере файл `C:\MyApp.sql` с командами SQL от имени одновременно 50 пользователей, соответствующая команда может выглядеть так:

```
ostress -E -dNorthwind -i "C:\MyApp.sql" -n50
```

Параметр `-E` означает, что вы подключаетесь при помощи логина Windows, при помощи параметра `-d` передается имя базы данных, параметр `-i` определяет имя файла с командами SQL (его можно создать автоматически, например, средствами профилировщика), параметр `-n` определяет, сколько потоков (т. е. подключений пользователей) запустится одновременно. Будьте внимательны: утилита чувствительна к регистру параметров!

Утилита `Ostress` может работать в двух режимах: *нагружочном режиме* (*stress mode*, этот режим используется по умолчанию) и в *режиме повторного выполнения* (*replay mode*). Первый режим предназначен главным образом для оценки производительности сервера при определенной нагрузке: в произвольной последовательности выполняются команды с использованием указанного вами количества потоков (подключений пользователей). Второй режим предназначен для точного повторного выполнения тех команд, которые были запротоколированы средствами профилировщика. Однако утилите `Ostress` нельзя напрямую передать файлы, созданные профилировщиком. Вначале их нужно перевести в специальный XML-совместимый формат RML. Эта операция производится при помощи утилиты `Read80Trace`. Кроме того, для взаимодействия со службой DTC (Distributed Transaction Coordinator — координатор распределенных транзакций) и отслеживания идентификаторов процессов (SPID) на SQL Server эта утилита использует специальный программный компонент `ORCA` (`Ostress Replay Control Agent` — управляющий агент повторного выполнения утилиты `Ostress`).

К сожалению, при использовании `Ostress` смущают два момента. Во-первых, эта утилита может работать только по ODBC. Второй момент связан с тем, что выполнение одних и тех же команд при использовании этой утилиты занимает во много раз больше времени, чем при работе в `Query Analyzer` или в

других аналогичных средствах. При отключении вывода на экран (параметр `-q`) и при работе с единственным потоком выполнение команд все равно замедляется в разы, и это не только из-за применения драйверов ODBC. Поэтому использовать `Ostress` для оценки того, с какой скоростью будут выполняться запросы пользователей при определенном их количестве, затруднительно.

Специально для нагрузочного тестирования дисковой подсистемы предназначена утилита `SQLIOStress`. Ее также можно бесплатно скачать с сайта Microsoft.

Еще одно программное средство от Microsoft, которое можно использовать для нагрузочного тестирования, называется `SQL Hammer`. Его можно найти на компакт-диске Resource Kit для SQL Server 2000. Фактически эта программа представляет собой шаблон приложения на языке Visual Basic, который можно изменять для своих целей.

Однако, с точки зрения автора, самым удобным средством для проведения нагрузочного тестирования является бесплатная программа `DBStressUtility`. Эта программа использует средства для работы с потоками .NET, и ее можно загрузить из Интернета вместе с исходным кодом (он совсем несложный, поэтому при желании его вполне можно доработать самостоятельно). Интерфейс этой программы представлен на рис. 11.5.

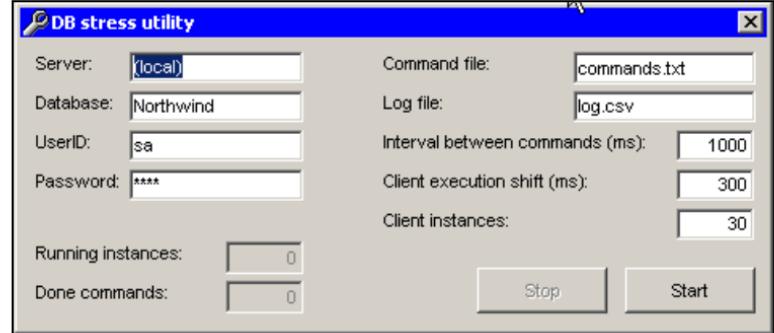


Рис. 11.5. Интерфейс DBStressUtility

Программа настолько проста, что каких-либо подробных объяснений не требуется. Отметим только один момент, с которым автор столкнулся на практике. Если сохранять результаты трассировки из окна SQL Server Profiler в файл скрипта с расширением `sql`, то профилировщик автоматически добавляет между командами в скрипте ключевое слово `GO`. `DBStressUtility`, встретив такое слово, прекращает чтение файла команд и выполняет только те команды, которые были до первого появления команды `GO`. Точно также `DBStressUtility`

относится и к пустым строкам. Поэтому перед передачей этой программе файла команд необходиомо очистить его от команд `GO` средствами поиска и замены любого текстового редактора. Другой вариант (более правильный) — запротоколировать информацию трассировки в таблицу SQL Server, а затем любыми средствами (пакет DTS/SSIS, bcp, Excel) экспорттировать эту таблицу в текстовый файл. Из таблицы, создаваемой профилировщиком, вам нужен будет только столбец `TextData`.

## 11.4.5. Приемы работы с Системным монитором

Главное средство для мониторинга производительности сервера SQL Server 2005 — это Системный монитор. Он может использоваться как для общей оценки загрузки сервера, так и для мониторинга специфических показателей, относящихся к работе SQL Server (например, какое количество запросов пользователей полностью обслуживается за счет кэша).

Запустить Системный монитор проще всего из меню **Пуск | Программы | Администрирование | Производительность**. Отметим сразу, что мониторинг настоятельно рекомендуется производить не с консоли сервера, а с рабочей станции администратора. При этом можно в ходе одного сеанса мониторинга собирать информацию о производительности сразу с нескольких серверов SQL Server 2005.

В консоли Системного монитора (рис. 11.6) предусмотрено два главных узла: **Системный монитор** и **Журналы и оповещения производительности**. В Системный монитор сведены возможности для интерактивного анализа производительности, когда администратор сидит у монитора и смотрит на изменения показателей счетчиков. В узел **Журналы и оповещения производительности** сведены возможности для автоматизации сбора информации.

В Системном мониторе предусмотрено три режима представления информации. Каждый из этих режимов используется в своей ситуации:

- **Просмотр диаграммы** — этот режим используется по умолчанию (он показан на рис. 11.6). Его удобнее всего использовать, чтобы производить мониторинг изменения показаний какого-то счетчика по времени. Например, вы запустили на SQL Server генерацию сложного отчета и хотите посмотреть, как меняются показатели нагрузки сервера. Обратите внимание, что выделить график для нужного счетчика можно, выбрав этот счетчик в таблице внизу экрана и нажав кнопку **Выделить** панели инструментов (или комбинацию клавиш `<Ctrl>+<H>`);
- **Просмотр гистограммы** — в этом режиме информация будет представлена в виде столбиков разной высоты. Обычно этот режим используется, чтобы сравнивать значения одних и тех же счетчиков с разных серверов;

- Просмотр отчета** — информация будет представлена в цифровом виде. Это самый удобный способ представления в ситуации, когда нужно одновременно отслеживать значения большого количества счетчиков.

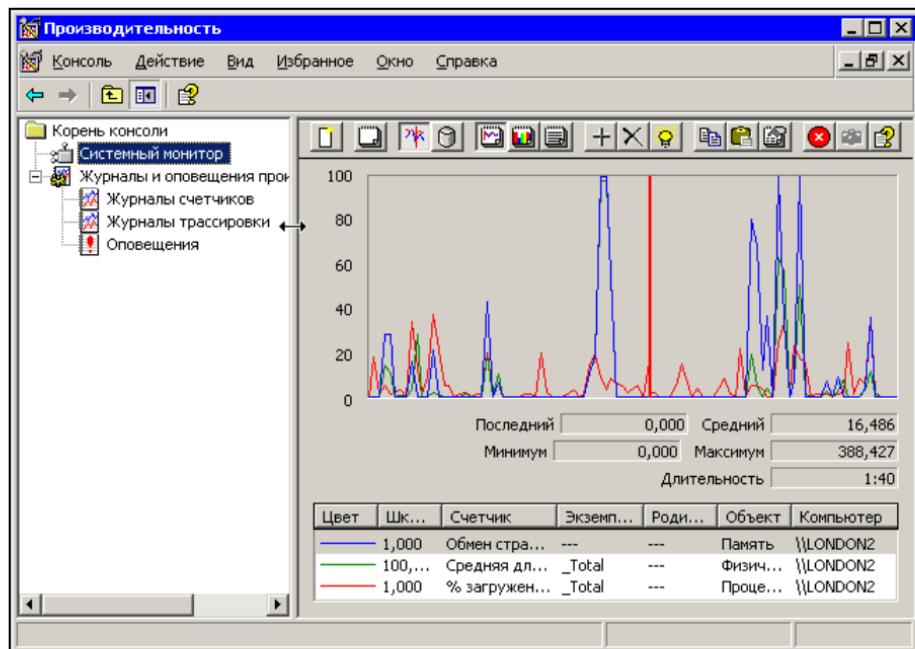


Рис. 11.6. Системный монитор

Изменять текущий режим можно при помощи нажатия соответствующей кнопки на панели инструментов.

Просмотр информации в окне Системного монитора очень удобен, однако сохранять собранную информацию для анализа тенденций нельзя. Удобнее всего для сбора и сохранения информации использовать журналы счетчиков из узла **Журналы и оповещения производительности**. Поскольку это средство используется чаще всего, рассмотрим его подробнее.

Для сохранения информации счетчиков вначале нужно создать журнал (при помощи команды **Новые параметры журнала** контекстного меню контейнера **Журналы счетчиков**). Вам будет предложено ввести имя нового журнала, а затем откроется окно его свойств. На первой вкладке **Общие** вы можете выбрать объекты (т. е. полные наборы счетчиков для определенной подсистемы) и отдельные счетчики, информация о которых будет записываться в журнал. Про объекты и счетчики, которые используются для мониторинга работы SQL Server 2005, будет рассказано в разд. 11.4.7—11.4.11. Кроме того, на этой вкладке вы можете настроить интервал сбора значений счетчиков.

Системный монитор — совсем нересурсоемкая задача. В большинстве ситуаций он не создает сколько-нибудь заметной нагрузки ни для сервера, мониторинг которого производится, ни для рабочей станции администратора, на котором он запущен. Однако для автоматизированной записи журнала показаний счетчиков вполне можно установить значения, равные нескольким минутам — чаще обычно и не нужно. При необходимости можно также определить учетную запись пользователя для подключения к серверу.

На второй вкладке **Файлы журнала** вы можете выбрать тип файлов журналов и указать каталог, в который они будут сохраняться. Обратите внимание, что по умолчанию запись производится в файл двоичного формата с расширением `blg`. Такой файл удобно просматривать в Системном мониторе, однако для других целей использовать его сложно. Пожалуй, самый удобный формат файла — **CSV (Comma-separated Values)** — значения, разделенные запятыми). Этот файл по умолчанию открывается в Excel (что очень удобно для анализа тенденций), кроме того, информацию из файла такого формата очень просто загрузить в базу данных, например, SQL Server. В Системном мониторе в Windows XP и Windows Server 2003 предусмотрена возможность загрузки информации напрямую в базу данных SQL Server (для этого нужно указать источник данных ODBC).

На последней вкладке **Расписание** вы можете определить время начала и окончания сбора информации, а также определить команду операционной системы, которая будет выполнена после окончания сбора данных.

Если вы не стали настраивать расписание, не забудьте после завершения настроек запустить сбор данных из контекстного меню для вашего журнала. Активный журнал (в который в настоящее время происходит сбор информации) выделяется зеленым цветом.

В Системном мониторе предусмотрена очень удобная возможность — интересующий вас набор счетчиков можно сохранить в виде файла HTML. Затем этот набор счетчиков можно открыть и использовать на другом компьютере. На предприятиях часто встречаются ситуации, когда нужно проводить мониторинг нескольких серверов, используя один и тот же набор счетчиков. Если счетчиков много, то использование этой возможности позволит вам сэкономить время: вам не придется выбирать одни и те же счетчики на разных компьютерах. Очень удобно то, что нигде в файлах HTML не прописывается имя компьютера, мониторинг которого производится, поэтому файлы, созданные на одном компьютере, можно использовать на другом. Но проблемы все-таки случаются:

- когда на компьютерах используется разное оборудование, информация о котором записывается в определении счетчика (обычно такая проблема возникает с сетевыми адаптерами);

- когда на разных компьютерах используются версии операционной системы с разными языками (например, с русским и английским). В русской версии счетчики операционной системы называются по-русски, и использовать созданный на таком компьютере файл HTML в англоязычной системе не удастся.

Создать файл HTML с набором счетчиков можно очень просто. Для этого достаточно щелкнуть правой кнопкой мыши в области графика в Системном мониторе и в контекстном меню выбрать команду **Сохранить как**. Если вам необходимо сохранить значения счетчиков из журнала, нужно точно так же щелкнуть правой кнопкой мыши по созданному журналу и выбрать в контекстном меню команду **Сохранить параметры как**.

Для того чтобы создать новый журнал на основе файла HTML, нужно щелкнуть правой кнопкой мыши по контейнеру **Журналы счетчиков** и в контекстном меню выбрать **Новые параметры журнала из**.

Сам файл HTML — это не просто хранилище набора счетчиков. Если открыть его в Internet Explorer, то будет загружен компонент ActiveX, в котором будут графически представлены сохраненные значения счетчиков аналогично тому, как это выглядит в Системном мониторе.

В Windows Server 2003 появилась еще одна версия Системного монитора, которая реализована в виде консольного приложения и представлена файлом logman.exe. У этого варианта Системного монитора те же возможности, что и у графического (не поддерживается только работа с файлами HTML). Эту утилиту очень удобно использовать для запуска сбора информации в журнал по расписанию или в ответ на какое-то событие.

## 11.4.6. Основы работы с объектами и счетчиками

Главное, что есть в Системном мониторе, — это объекты и счетчики, при помощи которых производится сбор информации. Добавление счетчиков производится при помощи окна, аналогичного представленному на рис. 11.7. Это окно открывается при нажатии на кнопку **Добавить счетчики** на вкладке **Общие** свойств журнала.

Отметим некоторые моменты, связанные с выбором счетчиков:

- в один журнал (или в одно окно Системного монитора) вполне можно выбрать счетчики с разных серверов. Так обычно и делается: в рамках одного сеанса мониторинга производительности собираются значения счетчиков со всех рабочих серверов SQL Server 2005;
- объект — это набор счетчиков, который относится к определенной подсистеме. Для некоторых объектов (например, для объекта **Databases**) можно выбрать конкретные экземпляры объекта на данном сервере при

помощи списка справа (они почему-то называются "вхождениями"). Для других объектов (например, для оперативной памяти) существует только один экземпляр объекта;

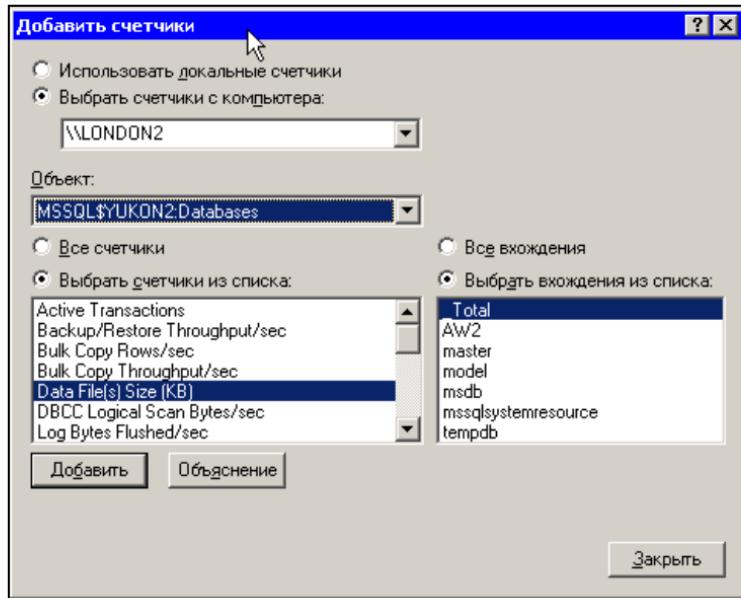


Рис. 11.7. Добавление счетчиков в Системном мониторе

- очень часто по названию счетчика понять, что он показывает, невозможно. Краткую справку по данному счетчику можно прочитать, нажав кнопку **Объяснение**. Более подробную справку по счетчикам SQL Server 2005 можно просмотреть в Books Online (раздел **SQL Server Books Online | SQL Server Database Engine | Administering the Database Engine | Monitoring and Tuning for Performance | Monitoring Resource Usage | Using SQL Server Objects** (Ядро базы данных SQL Server | Администрирование ядра базы данных | Мониторинг и тюнинг производительности | Мониторинг использования ресурсов | Использование объектов SQL Server)).

Для мониторинга сервера SQL Server 2005 вам потребуются как счетчики операционной системы, так и самого SQL Server 2005.

#### 11.4.7. Счетчики для анализа загрузки процессора

Главный счетчик для анализа загрузки процессора — **% загруженности процессора** (% Processor Time) для объекта **Процессор** (Processor). Значение этого счетчика не должно превышать 80% в течение длительного промежутка

времени. Если оно стабильно больше, то это значит, что процессорная подсистема работает с перегрузкой и является узким местом сервера. Пора обновлять процессор, или снимать с сервера часть нагрузки, или переходить на многопроцессорную систему, или задуматься о применении 64-разрядной платформы. Отметим только один важный момент: высокое значение счетчика загрузки процессора может возникнуть из-за дефицита оперативной памяти или перегрузки дисковой подсистемы. Поэтому вполне может получиться так, что добавление оперативной памяти или замена RAID-массива на более быстрый одновременно решит проблему с процессором.

Если вы производите мониторинг производительности мультипроцессорной системы, то для оценки загрузки всех процессоров можно использовать экземпляр **\_Total** того же объекта **Процессор**. В предыдущих версиях операционной системы для этой цели был предназначен счетчик **% общее время загруженности процессоров** для объекта **Система** (System), но в Windows Server 2003 его уже нет.

Есть еще несколько счетчиков, относящихся к работе процессора, которые нужно принимать во внимание:

- **Длина очереди процессора** (Processor Queue Length) для объекта **Система**. Этот счетчик показывает, сколько запросов стоит в очереди на обработку центральным процессором. Значение этого счетчика не должно в течение продолжительного времени превышать количество процессоров в системе, умноженное на 2 (например, если на сервере 4 процессора, то это значение не должно быть больше 8);
- **% загруженности процессора** (% Processor Time), но уже для другого объекта — **Процесс** (Process), и конкретно для экземпляра объекта **sqlservr** с нужным номером. Этот счетчик показывает, сколько времени затрачивается процессором на обслуживание запросов SQL Server. Этот показатель рекомендуется сравнивать с аналогичным показателем для других процессов. Может выясниться, что ваш сервер тратит свои силы на что-то еще;
- **% работы в привилегированном режиме** (% Privileged Time) для объекта **Процессор**. Этот параметр показывает, сколько времени тратится на обслуживание запросов ядра. Это время должно быть как можно меньше: на загруженном сервере все ресурсы должны отдаваться серверным приложениям (т. е. SQL Server). Если значение счетчика большое (составляет десятки процентов), то нужно проверить, нет ли у вас проблем с системой. Например, сильно загрузить процессор может неправильно сконфигурированное оборудование. Проверить это можно при помощи следующего счетчика;
- **% времени прерываний** (% Interrupt Time) для объекта **Процессор**. Этот параметр показывает, сколько времени процессор тратит на обработку

прерываний от оборудования. Если при установке какого-либо оборудования значение этого счетчика сильно возрастает, с этим оборудованием наверняка какие-то проблемы.

При работе на многопроцессорных системах иногда возникает еще одна проблема. Она заключается в том, что SQL Server не всегда корректно производит распараллеливание при выполнении запросов. За счет избыточного числа переключений между процессорами (*context switching*) скорость выполнения запроса на многопроцессорной системе может оказаться заметно ниже, чем на однопроцессорной. В этой ситуации в вашем распоряжении есть два решения:

- использовать параметр настройки сервера `cost threshold for parallelism` (его можно настроить при помощи хранимой процедуры `sp_configure`). Он определяет стоимость запроса, начиная с которой выполнения этого запроса будет распараллеливаться. По умолчанию в SQL Server 2000 и SQL Server 2005 для этого значение установлено значение 5. Это значит, что будут распараллеливаться запросы, ожидаемое время выполнения которых больше 5 секунд (на самом деле такой порог будет заметно меньше 5 секунд, поскольку "калибровка" единиц стоимости производилась в середине 90-х годов при разработке SQL Server 7.0). Увеличение значения этого параметра приведет к тому, что будет распараллеливаться меньшее количество запросов. Однако проблема распараллеливания больших запросов все равно останется;
- использовать флаг трассировки 8687. Он просто запрещает распараллеливание запросов. Кроме решения проблем с некорректным распараллеливанием, его применение дает дополнительные важные преимущества: особо "тяжелый" запрос не сможет захватить все ресурсы всех процессоров, и он повлияет на работу других пользователей в меньшей степени.

По опыту проведения автором курсов по SQL Server, отмечено, что этот флаг используется специалистами на очень многих предприятиях, поскольку без него работать бывает достаточно сложно. Включить флаг трассировки можно так:

```
-- Вначале определяем, что флаг будет распространен на все подключения
DBCC TRACEON (-1) ;
-- Затем включаем сам флаг трассировки
DBCC TRACEON (8687) ;
```

Можно настроить автоматическое включение этого флага трассировки при запуске сервера. Для этого при запуске службы SQL Server используется параметр командной строки -t с номером флага трассировки. В этом случае флаг будет автоматически применяться ко всем клиентским подключениям.

Для просмотра информации о всех установленных флагах трассировки можно использовать команду:

```
DBCC TRACESTATUS (-1)
```

## 11.4.8. Счетчики для анализа загрузки оперативной памяти

С точки зрения Microsoft, самая важная подсистема для баз данных OLTP (к которым относятся почти все используемые в повседневной работе базы данных) — это подсистема оперативной памяти. Более 90% запросов пользователей на чтение и запись обслугивается из буфера в оперативной памяти, без необходимости немедленного обращения к диску. Кроме того, нехватка памяти ведет к свопингу (активному обращению к файлу подкачки) и может привести к дополнительной нагрузке на процессорную и дисковую подсистемы.

Главный счетчик для анализа загрузки оперативной памяти с точки зрения операционной системы — это счетчик **Обмен страниц в сек** (Pages/sec) объекта **Память** (Memory). Физически этот счетчик показывает количество обращений в секунду к файлу подкачки (неважно, на чтение или запись). В курсах по SQL Server 6.5 и 7.0 пороговое значение этого счетчика определялось равным 5, в курсах по SQL Server 2000 — равным 10. В курсах и документации по SQL Server 2005 пороговое значение для этого счетчика вообще не определяется (пишут только о недопустимости его большого значения). Когда системе на самом деле не хватает оперативной памяти, значение этого счетчика измеряется сотнями, так что ошибиться здесь трудно. Отметим, что критична только информация, собранная за длительный промежуток времени — кратковременные скачки этого счетчика вполне допустимы.

Главный счетчик для определения того, хватает ли памяти самому SQL Server 2005, — счетчик **Buffer cache hit ratio** (Процент попаданий в кэш буфера) объекта **Buffer Manager** (Менеджер буфера) для данного экземпляра сервера. Он показывает, сколько запросов пользователей (в процентах к общему) обслуживаются из буфера без необходимости обращения к диску. Как уже говорилось ранее, рекомендуется, чтобы в течение длительного промежутка времени значение этого счетчика было не меньше 90%.

Для мониторинга подсистемы оперативной памяти можно использовать и другие счетчики:

- если вы подозреваете, что в расходе памяти виноват не SQL Server 2005, а другое приложение, имеет смысл обратиться к объекту **Процесс** и посмотреть на значения счетчиков **Рабочее множество** (Working Set) и **Ошибка страницы/сек** (Page faults/sec). Первый счетчик показывает, сколько пам-

мяти в настоящий момент использует каждый конкретный процесс, второй — сколько раз для этого процесса пришлось обращаться к файлу подкачки. Значения этих счетчиков для объекта процесса SQL Server нужно сравнить с экземпляром **\_Total** и другими процессами;

- счетчик **Доступно байт** (Available baits) для объекта **Память** представляет объем физической памяти компьютера, которая свободна и может быть немедленно выделена какому-либо процессу. Значение этого счетчика в течение длительного промежутка времени не должно приближаться к 0;
- счетчик **Total Server Memory (KB)** (Общая память сервера (KB)) для объекта **Memory Manager** (Менеджер памяти) соответствующего экземпляра SQL Server определяет, сколько именно виртуальной памяти (включая страницы в файле подкачки) используют подсистемы SQL Server. Это значение должно быть существенно ниже, чем объем физической оперативной памяти на сервере.

Если вы определили, что на сервере не хватает оперативной памяти, то естественно для решения проблемы ее нужно добавить. Однако отметим некоторые моменты, которые связаны с оперативной памятью для SQL Server 2005.

В 32-разрядных системах существуют ограничения на объем адресуемой оперативной памяти — 4 Гбайт. Если на сервере оперативной памяти больше (а такие системы все чаще встречаются на предприятиях), то для того, чтобы SQL Server мог использовать всю память, нужно предпринять необходимые действия и в операционной системе, и на самом SQL Server. В операционной системе нужно прописать для ее строки загрузки (пути ARC) в файл boot.ini дополнительный параметр /PAE (Physical Address Extension — расширение физических адресов), например:

```
multi(0)disk(0)risk(0)partition(2)\WINDOWS="Windows Server 2003,
Enterprise RU" /no execute=opt out /fast detect /PAE
```

На SQL Server нужно включить параметр AWE enabled (Address Windowing Extensions). Включение его может выглядеть так:

```
sp_configure 'show advanced options', 1;
RECONFIGURE;
GO
sp_configure 'awe enabled', 1;
RECONFIGURE;
GO
```

После этого нужно перезапустить сервер.

Работа SQL Server в режиме AWE сопряжена с определенными проблемами. В частности, SQL Server никогда не отдает выделенную ему память AWE обратно, эта память никогда не выгружается в файл подкачки, могут возникнуть

проблемы с распределением памяти AWE между экземплярами SQL Server, работающими на одном компьютере. Поэтому настоятельно рекомендуется при использовании режима AWE всегда устанавливать для SQL Server параметры `min server memory` (минимальное количество памяти, используемой SQL Server) и `max server memory` (максимальное количество памяти), например, так:

```
sp_configure 'min server memory', 1024;
RECONFIGURE;
GO
sp_configure 'max server memory', 6144;
RECONFIGURE;
GO
```

Вообще, если ваша задача "доросла" то таких требований к ресурсам, имеет смысл подумать о приобретении 64-разрядного сервера. Конечно, работать с 64-разрядной версией SQL Server 2005 не очень удобно (например, из-за необходимости запускать некоторые средства администрирования только на 32-разрядной системе, т. е. удаленно), но 64-разрядная система действительно может обеспечить большой выигрыш в производительности. Кроме того, 64-разрядные системы могут работать с памятью практически неограниченного объема.

Отметим еще один момент, который связан с увеличением физического количества оперативной памяти. При принятии решения о добавлении на сервер физической оперативной памяти всегда нужно помнить еще про один параметр — кэш второго уровня (*L2 cache*), который уже многие годы является компонентом процессора. Размер кэша второго уровня — это один из главных параметров, который отличает процессоры линейки Celeron от обычных Pentium и обычные Pentium — от серверных вариантов. Если кэша второго уровня недостаточно, то увеличение производительности при добавлении оперативной памяти будет происходить очень нелинейно, а в некоторых (совсем редких) случаях может и упасть. Оптимальный вариант — когда 2 Кбайт кэша второго уровня приходятся на каждый мегабайт физической оперативной памяти, хотя могут быть и другие варианты. В этом случае лучше посмотреть техническую документацию по серверной платформе — какой объем оперативной памяти для нее является рекомендуемым.

## 11.4.9. Счетчики для анализа производительности дисковой подсистемы

Третья подсистема, мониторинг которой имеет смысл производить для сервера SQL Server 2005, — это дисковая подсистема. Она считается главной с точки зрения производительности для задач Data Warehousing, т. е. для хра-

нилищ данных. Объяснение достаточно простое — по сравнению с обычными системами OLTP, наиболее распространенная операция в системах Data Warehouse — это полное сканирование всей таблицы. Операции изменения данных в системах Data Warehouse также очень редки. Поэтому роль кэша в них, в отличие от систем OLTP, невелика, а вот требования к дискам предъявляются намного серьезнее.

По опыту практической работы можно отметить, что и для систем OLTP именно увеличение скорости работы дисковой подсистемы дает наибольший эффект.

Главный счетчик для дисковой подсистемы — **Процент загрузки диска (%) Disk Time** для объекта **Логический диск** (Logical Disk) или **Физический диск** (Physical Disk). Объект **Логический диск** удобнее, поскольку информация показывается по разделам, но он почему-то не всегда доступен. Пороговое значение для этого счетчика формулируется так: в течение продолжительного промежутка времени значение не должно приближаться к 100%. Если дисковая подсистема постоянно загружена почти на 100%, то очевидно, что она является узким местом системы и ее надо заменить. Если вы не уверены, что за активность дисковой подсистемы ответственен именно SQL Server, имеет смысл проверить значение счетчика **Обмен данными, байт в секунду (I/O bytes/sec)** для объекта **Процесс** (сравнить это значение для экземпляра SQL Server и объекта **\_Total**).

Отметим лишь одну особенность счетчиков **Процент загрузки данных** для объектов **Логический диск** и **Физический диск**: они не всегда показывают точные значения при использовании RAID-массивов. Так, в практической работе иногда можно увидеть значение этого параметра более 100%. Однако больших проблем это не создает: при нормальной нагрузке на дисковую подсистему среднее значение этого счетчика должно быть существенно ниже 50%.

Еще один счетчик, который можно использовать для проверки, — **Текущая длина очереди диска** (Current Disk Queue Length) для тех же объектов **Физический диск** и **Логический диск**. Этот параметр показывает, сколько запросов стоит в очереди на обработку дисковой подсистемы. Его значение на протяжении длительного промежутка времени не должно превышать 2 для одного диска (если, например, в RAID-массиве находится 10 дисков, то среднее значение этого счетчика не должно быть больше 20).

Если вы не уверены, чем именно вызвана загрузка диска — операциями чтения или записи, то определить это помогут счетчики **% Активности диска при чтении (% Disk Read Time)** и **% Активности диска при записи (% Disk Write Time)** для объектов **Физический диск** и **Логический диск**.

Если узким местом системы является диск (а это случается очень часто), то вариантов решения может быть несколько. Самое простое и очевидное из

них — обновить дисковую подсистему, например, купив новый RAID-массив. Однако большой выигрыш могут дать и другие, более дешевые варианты решения.

Очень часто лишние операции с диском возникают из-за проблем с индексами. Про оптимизацию системы индексов будет рассказываться далее в разд. 11.5.5.

Большое влияние на производительность и отказоустойчивость работы SQL Server оказывает распределение файлов по дискам сервера.

## 11.4.10. Счетчики для анализа производительности сетевой подсистемы

Последняя физическая подсистема сервера, мониторинг которой имеет смысл производить, — сетевая подсистема. Как правило, пользователи подключаются к SQL Server по сети.

Обычно какие-то проблемы с производительностью работы сети, вызванные именно работой SQL Server, возникают редко. Как правило, сеть намного активнее загружается другими приложениями (например, приложениями на основе настольных баз данных — FoxPro, Access, Clipper, Paradox и т. п.).

На предприятиях часто встречаются ситуации, когда за работу сети отвечает один человек (или подразделение), а за работу приложения, использующего SQL Server, — другой. И договориться им часто бывает сложно. Рассмотрим возможности мониторинга загруженности сети с точки зрения администратора SQL Server.

Проблемы с производительностью работы сети могут встретиться в двух вариантах. Первый вариант — не хватает производительности конкретного сетевого адаптера, установленного на сервере. Чтобы проверить это, можно воспользоваться счетчиком **Всего байт/сек** (Bytes total/sec) для объекта **Сетевой интерфейс** (Network Interface) (выбрав нужный сетевой интерфейс из списка). Значение этого счетчика предлагается сравнивать с паспортным значением для данного сетевого интерфейса. Если сетевой интерфейс действительно работает на пределе своих возможностей, то его можно заменить на специальный серверный мультипортовый адаптер.

Намного чаще возникает другая ситуация — когда проблема не в сетевом интерфейсе, а в общем уровне загрузки сети. В этом случае работа клиентов с SQL Server может серьезно замедлиться. Могут происходить тайм-ауты при установке соединений, разрывы уже установленных соединений и т. п. Как отследить общий уровень загрузки сети и в случае необходимости представить доказательства сетевому администратору?

В Системном мониторе подходящего объекта нет. Он был в Windows NT 4.0 (и назывался **Сетевой сегмент** (Network Segment)), но в Windows 2000 и 2003 его убрали (мотивируя это заботой о безопасности). Поэтому придется использовать другие способы. Средства оценки производительности сети встроены в большинство снiffeров (одни из самых мощных — в Net Observer), в том числе в снiffeр от Microsoft, который называется Сетевой монитор (Network Monitor). Правда, использовать для анализа загрузки сети ту версию Сетевого монитора, которая поставляется с Windows 2000 и 2003 (ее можно установить, выбрав в списке необязательных компонентов операционной системы), бесполезно. Этот снiffeр видит только трафик, который передается с этого компьютера, на этот компьютер, широковещательный трафик и трафик групповой рассылки. Если трафик генерируется другими компьютерами, то стандартная версия Сетевого монитора просто его не увидит и, соответственно, не отобразит при анализе уровня загрузки. Поэтому нужно использовать версию Сетевого монитора, которая поставляется с Systems Management Server.

После установки запуска Сетевого монитора вам нужно выбрать сетевой адаптер, который будет использоваться для перехвата пакетов в сети. Затем нужно будет в меню **Захват** (Capture) выбрать команду **Старт** (Start), чтобы начать захват данных. Информация о загрузке сети (**% загрузки сети**) будет представлена как в графическом, так и в цифровом виде (в правой части экрана).

Отметим, что для обычной сети на хабах (без применения свитчей) пороговое значение этого счетчика, согласно Microsoft, составляет всего 40%. Превышение этого значения ведет к прогрессирующему уменьшению производительности (чем больше нагрузка, тем больше коллизий и повторов передачи данных, а соответственно опять происходит увеличение нагрузки, и так по нарастающей).

В реальной работе часто возникают ситуации, когда производительность работы сети ограничена по объективным причинам. Например, пользователи подключаются к SQL Server из филиалов по низкоскоростным каналам связи. Иногда нужно обеспечить возможность подключения пользователей через брандмауэр, за настройки которого ответственны другие администраторы. И в том и в другом случае имеет смысл подумать о двух вариантах решения:

- использовать приложение с Web-интерфейсом;
- использовать терминальный доступ при помощи Microsoft Terminal Services или Citrix MetaFrame.

Первый вариант требует наличия специальной Web-версии клиентского приложения, которая есть далеко не всегда (а самостоятельно создать ее бывает сложно, особенно учитывая то, что подробной информации о структуре базы

данных в вашем распоряжении может и не быть). Зато второй вариант можно использовать в большинстве ситуаций. При этом пользователи работают с обычным вариантом клиентского приложения, но физически оно запускается не на их компьютере, а на специальном сервере. С клиентской рабочей станции на терминальный сервер передаются только нажатия клавиш и щелчки мышью, а с сервера на рабочую станцию — изменения в экране. Преимущества такого решения — множество:

- терминальные решения требуют минимального трафика в сети и обычно нормально работают даже по модемным коммутируемым соединениям;
- в случае разрыва соединения пользователь может заново подключиться к своему сеансу и продолжить работу с того же места;
- на клиентский компьютер не нужно устанавливать практически никаких клиентских приложений (кроме клиента терминального сервера). Это хорошо и с точки зрения безопасности, и с точки зрения снижения нагрузки на администратора;
- на клиентском компьютере может быть установлена практически любая операционная система (при использовании Citrix MetaFrame, в том числе и DOS, и UNIX). Требования к системным ресурсам компьютера также минимальны;
- трафик Citrix MetaFrame и Microsoft Terminal Server автоматически шифруется (в отличие от трафика работы обычных клиентов с SQL Server, которые подключаются, например, по OLE DB или ODBC);
- если сетевое соединение защищено брандмауэром, то открыть доступ для терминальных соединений обычно проще, чем для подключения обычных приложений к SQL Server.

Недостатков у терминальных решений не так много: требуются лицензии на терминальный сервер; нужен дополнительный сервер (достаточно мощный, если он будет обслуживать большое количество пользователей); некоторые специальным образом защищенные (например, ключами HASP) или очень ресурсоемкие приложения могут не работать на терминальном сервере. Однако в настоящее время популярность терминальных решений растет очень быстро (например, для клиент-серверной версии 1С), и помнить о такой возможности определенно стоит.

Другие варианты повышения производительности сетевой подсистемы при работе с SQL Server обычно включают в себя использование свитчей вместо хабов, обновление сетевой инфраструктуры (с 10 Мбит до 100, с 100 Мбит — до 1 Гбит и т. п.), а также выделение клиентов вместе с SQL Server в отдельный сегмент сети.

## 11.4.11. Объекты Системного монитора для мониторинга работы SQL Server 2005

Как ни удивительно, для оценки производительности работы сервера чаще используются счетчики операционной системы, чем счетчики самого SQL Server. Однако для SQL Server 2005 в Системном мониторе также предусмотрен большой набор счетчиков производительности, которые могут оказаться очень полезными.

У счетчиков SQL Server есть интересная особенность: интерфейс для доступа к ним продублирован на уровне самого SQL Server. Просмотреть их можно из кода Transact-SQL, обратившись к специальному представлению `sysperfinfo` в базе данных `master`. Использование этого представления может оказаться очень удобным для прямого протоколирования информации о производительности, без необходимости обращения к Системному монитору. К сожалению, большая часть значений счетчиков представлена в `sysperfinfo` в кумулятивном виде (т. е. эта информация постоянно складывается с момента запуска сервера). Поэтому попытка посмотреть, например, значение счетчика **Buffer cache hit ratio** (Процент попаданий в кэш буфера) для объекта **Buffer manager** при помощи запроса вида:

```
SELECT * FROM sysperfinfo WHERE counter_name = 'Buffer cache hit ratio';
```

вернет вам совершенно невообразимую цифру (при том, что это значение никак не может быть выше 100%). Для получения значимой информации из этой таблицы приходится использовать специальные приемы.

Предположим, что вы хотите запротоколировать информацию о количестве транзакций в секунду для базы данных `Foodmart`. При этом сбор информации должен производиться в течение 1 минуты, а интервал для сбора должен составлять 5 секунд. Код Transact-SQL для решения этой задачи может быть таким:

```
-- Создаем временную таблицу со столбцами времени и значений счетчика
CREATE TABLE #PerfTable(ts DATETIME, TranPerSec BIGINT);
DECLARE @iVar1 INT, @iVar2 INT, @dStartTime DATETIME
-- Получаем текущее время
SET @dStartTime = getdate();
-- Получаем текущее значение счетчика и ждем 5 секунд
SELECT @iVar2=cntn_value FROM master..sysperfinfo WHERE counter_name =
'Transactions/sec' AND instance_name = 'Foodmart';
WAITFOR DELAY '00:00:05'
-- Запускаем цикл, который будет работать минуту
WHILE getdate() < dateadd(mi,1,@dStartTime)
```

```
BEGIN
-- Получаем еще раз значение счетчика (когда прошло 5 секунд)
SELECT @iVar1=cntr_value FROM master..sysperfinfo WHERE counter_name =
'Transactions/sec' AND instance_name = 'Foodmart';
-- Вставляем разницу между двумя значениями в таблицу #PerfTable
INSERT INTO #PerfTable (ts,TranPerSec) VALUES (getdate(),
@iVar1 - @iVar2);
-- ... и записываем текущее значение в первую переменную,
SET @iVar2 = @iVar1;
-- Опять ждем пять секунд
WAITFOR DELAY '00:00:05'
END
-- Когда цикл закончится, выводим записанные в таблицу значения
SELECT * FROM #PerfTable;
```

Далее перечислена информация об объектах SQL Server 2005 в Системном мониторе:

- **Access Methods** (Методы доступа) — этот объект представляет большой набор счетчиков по статистике выполнения пользовательских операций (сколько было произведено обращений к индексам, сколько операций полного сканирования таблиц и т. п.);
- **Backup Device** (Устройство резервного копирования) — для этого объекта предусмотрен единственный счетчик, который показывает скорость резервного копирования или восстановления;
- **Broker Activation** (Активация брокера) — статистика по хранимым процедурам, которые активизируют очереди Service Broker для определенной базы данных;
- **Broker Statistics** (Статистика брокера) — общая статистика по производительности Service Broker (количество сообщений в очереди, количество перенаправленных сообщений в секунду и т. п.);
- **Broker/DBM Transport Object** (Объект транспорта брокера/зеркального отображения баз данных) — статистика по зеркальному отображению баз данных (DBM, DataBase Mirroring). В этом объекте содержатся счетчики для операций, в которых принимает участие Service Broker (количество сообщений, байт, передаваемых в секунду, и т. п.);
- **Buffer Manager** (Менеджер буфера) — это очень важный объект, который предназначен для мониторинга статистики кэша буферов базы данных. Самый важный счетчик, про который уже говорилось, — **Buffer cache hit ratio** (Процент попаданий в кэш буфера);
- **Buffer Partition** (Раздел буфера) — статистика по пустым страницам кэша буферов базы данных;

- **CLR** — в этом объекте представлена информация по обращениям к сборкам .NET в коде Transact-SQL. В нем предусмотрен единственный счетчик **CLR Execution** (Выполнение CLR) — время выполнения текущего обращения в микросекундах;
- **Cursor Manager by Type** (Менеджер курсоров по типам) — этот объект предназначен для мониторинга производительности при работе с курсорами. В качестве экземпляров этого объекта представлены типы курсоров;
- **Cursor Manager Total** (Общее по менеджеру курсоров) — это общая статистика по работе с курсорами, независимо от их типа;
- **Database Mirroring** (Зеркальное отображение баз данных) — этот объект (вместе с объектом **Broker/DBM Transport Object**) предназначен для мониторинга зазеркаливаний баз данных. При помощи его счетчика можно узнать скорость передачи данных и размер очереди;
- **Databases** (Базы данных) — это, наверное, самый популярный объект Системного монитора для SQL Server. В нем предусмотрено большое количество счетчиков для всех параметров баз данных: размер файлов данных и журналов транзакций, количество активных транзакций и т. п. Очень удобен счетчик **Percent Log Used** (Процент журнала использован), который можно использовать для мониторинга оставшегося пространства в журнале транзакций;
- **ExecStatistics** (Статистика выполнения) — этот объект предназначен для получения информации о выполнении распределенных запросов, вызовов распределенного координатора транзакций, расширенных хранимых процедур и вызовов OLE DB;
- **General Statistics** (Общая статистика) — как понятно из названия, это общая статистика работы сервера SQL Server 2005. Например, при помощи счетчика **User Connections** (Подключения пользователей) можно узнать число пользователей, которые в настоящее время подключены к SQL Server;
- **Latches** (Кратковременные блокировки) — статистика по этому специальному типу блокировок. *Кратковременные* блокировки — это аналоги блокировок, которые налагаются на ресурсы баз данных во избежание проблем. Но кратковременные блокировки налагаются не на страницы в базе данных, а на специальные области в оперативной памяти (например, которые используются для организации страниц в буфере). В основном используется значение счетчика **Latch Waits/Sec** (Количество ожиданий на установку кратковременных блокировок). Слишком большое значение этого счетчика может говорить о недостатке оперативной памяти для SQL Server;

- **Locks** (Блокировки) — это информация о блокировках и взаимоблокировках SQL Server. Если время ожидания для блокировок слишком большое, или при работе пользователей регулярно возникают взаимоблокировки, то транзакции в приложении построены неправильно, и есть повод предъявить претензии разработчикам;
- **Memory manager** (Менеджер памяти) — этот объект можно использовать для отслеживания информации о том, как SQL Server распределяет оперативную память между своими подсистемами;
- **Plan Cache** (Кэш планов) — этот объект предназначен для получения информации о процедурном кэше, т. е. о кэше, в котором хранятся планы выполнения команд Transact-SQL, хранимых процедур и триггеров;
- **SQL Errors** (Ошибки SQL) — для этого объекта предусмотрен единственный счетчик, который показывает количество ошибок в секунду, возникающих при выполнении команд SQL;
- **SQL Statistics** (Статистика SQL) — это статистика по выполнению команд SQL на сервере;
- **Transactions** (Транзакции) — это статистика по транзакциям на сервере, открытых в настоящий момент;
- **Wait Statistics** (Статистика ожиданий) — этот важный объект показывает информацию об ожиданиях на SQL Server. Обычно такие ожидания возникают, когда системе не хватает какого-то ресурса (оперативной памяти, скорости сетевого ввода/вывода и т. п.). Однако в отличие от сервера Oracle, вмешаться и выделить вручную место для определенной области в оперативной памяти вы не можете;
- **User Settable** (Настраиваемые пользователем) — это специальный объект, значения для 10 счетчиков которого устанавливаются из кода Transact-SQL при помощи хранимых процедур `sp_user_counter1`, `sp_user_counter2` и т. п.

Свой набор счетчиков предусмотрен и для SQL Server Agent:

- **Alerts** (Предупреждения) — этот объект представляет информацию о сработавших предупреждениях SQL Server Agent. В нем есть всего два счетчика: **Activated Alerts** (Сработавшие предупреждения) (общее количество предупреждений, которые сработали с момента запуска сервера) и **Alerts Activated/Minute** (Активированные предупреждения в минуту) (количество оповещений, которые сработали за последнюю минуту);
- **Jobs** (Задания) — информация о заданиях SQL Server Agent (сколько было выполнено успешно, сколько завершилось с ошибками, сколько работает в настоящее время);

- **Job Steps** (Этапы заданий) — статистика по этапам заданий SQL Server Agent;
- **Statistics** (Статистика) — для этого объекта предусмотрен единственный счетчик, который показывает, сколько раз SQL Server был перезапущен в автоматическом режиме службой SQL Server Agent.

Счетчиков для SQL Server предусмотрено очень много, но на практике обычно используются лишь несколько из них. Далее представлен перечень наиболее важных счетчиков SQL Server:

- **Buffer Manager: Buffer Cache Hit Ratio** (Менеджер буфера: процент попаданий в кэш) — как уже говорилось, это главный счетчик, который позволяет определить, достаточно ли памяти отведено самому SQL Server. В системах OLTP значение этого счетчика в течение длительного времени работы сервера должно быть выше 90% (в некоторых источниках говорится, что выше 95%). Для хранилищ данных допускаются более низкие значения;
- **Buffer Manager: Cache Size (pages)** (Менеджер буфера: размер кэша в страницах) — этот счетчик показывает, сколько памяти выделено под кэш SQL Server. Обратите внимание, что его значение определяется в страницах, поэтому, чтобы получить размер памяти в байтах, полученное значение нужно умножить на 8192. В идеале значение этого параметра должно быть близко к размеру физической оперативной памяти на сервере. Если оно существенно меньше, и на сервере остается много свободной памяти, то возможно SQL Server по каким-то причинам не может использовать оперативную память. Проблема может заключаться в том, что не включено использование AWE, или используется неверная редакция SQL Server, или на сервере настроены статически параметры работы с оперативной памятью;
- **SQL Server Access Methods: Page Splits/sec** (Методы доступа SQL Server: Разбиений страниц в секунду) — большое значение этого показателя означает, что при выполнении операций вставки и изменения данных SQL Server приходится выполнять большое количество ресурсоемких операций по разбиению страниц и переносу части существующей страницы на новое место. Таких операций по возможности следует избегать. Проблему можно попытаться решить двумя способами:
  - создать кластерный индекс для столбцов с автоприращением. В этом случае новые записи не будут помещаться внутрь страниц, уже занятых данными, а будут последовательно занимать новые страницы;
  - перестроить индексы, увеличив значение параметра `Fillfactor`. Этот параметр позволяет зарезервировать в страницах индексов свободное

место, которое будет использоваться для размещения новых данных, без необходимости производить операции разбиения страниц;

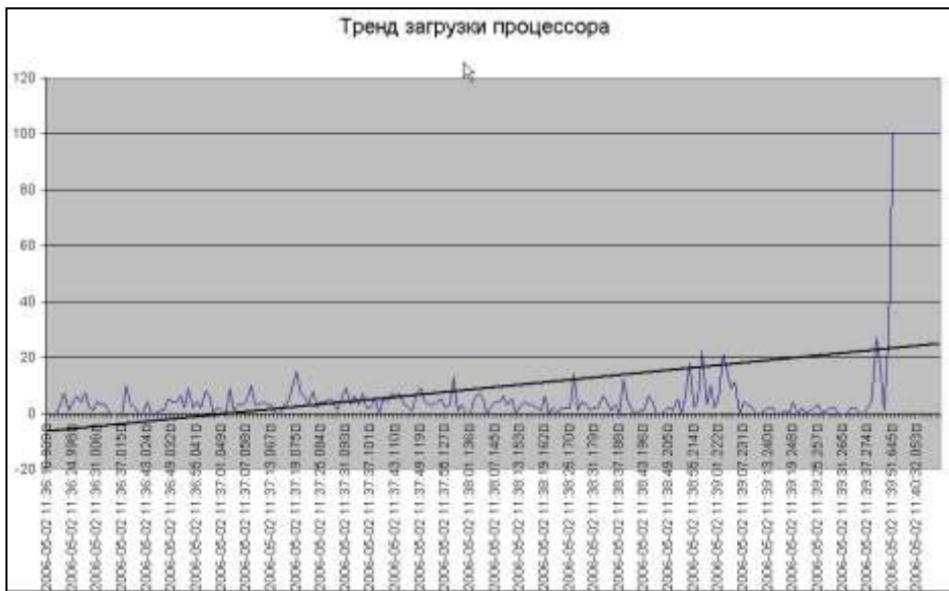
- **SQL Server Locks: Average Wait Time (ms)** (Блокировки SQL Server: Среднее время ожидания (миллисекунды)) — этот счетчик показывает, сколько в среднем процессам пользователей приходится проводить в очереди, чтобы наложить на ресурс блокировку. Максимально допустимое значение этого счетчика полностью зависит от вашей задачи, какое-то среднее значение для всех приложений здесь определить сложно. Слишком высокое значение этого счетчика может означать проблемы с блокировками в вашей базе данных. Подробно про работу с блокировками будет рассказываться в *разд. 11.5.7*;
- **SQL Server Access Methods: Full Scans/sec** (Методы доступа SQL Server: Количество операций полного сканирования таблиц в секунду) — этот счетчик определяет, сколько операций полного сканирования таблиц (без обращения к индексам) SQL Server выполняет в секунду. Для хранилищ данных такие операции совершенно нормальны и каких-то подозрений вызывать не должны. Однако если слишком большое количество операций полного сканирования проводится в базе данных, обслуживающей OLTP-систему, то вполне возможно, что индексов в этой базе данных недостаточно. Другая причина слишком высокого значения этого счетчика может заключаться в том, что разработчики предпочли фильтровать данные уже в приложении, а не на сервере. Про оптимизацию системы индексов будет рассказываться в *разд. 11.5.5*, а про анализ запросов — в *разд. 11.5.8*.

## Задание для самостоятельной работы 11.2. Приемы работы с Системным монитором

### Задание:

1. Сохраните в файл HTML набор счетчиков Системного монитора, предлагаемый по умолчанию.
2. Используйте этот файл HTML для создания нового журнала. Информация счетчиков при этом должна записываться в таблицу `PerfTable` в базе данных `AdventureWorks`.
3. При помощи утилиты командной строки `logman` запустите созданный вами журнал, запустите пакет `CopyPerson.dtsx`, созданный вами в самостоятельной работе 10.1, а потом при помощи той же утилиты `logman` остановите сбор информации.
4. Средствами Excel создайте график на основе собранной информации и проанализируйте тенденцию загрузки процессора на вашем сервере. Соот-

ветствующая диаграмма может выглядеть, например, так, как представлено на рис. 11.8.



**Рис. 11.8.** Построение тренда при анализе собранных данных нагрузки процессора

### Решение:

К пункту 1 задания — сохранение набора счетчиков Системного монитора в файл HTML:

1. Откройте консоль **Производительность на вашем компьютере** (**Пуск | Программы | Администрирование | Производительность**).
2. Раскройте узел **Системный монитор** и щелкните правой кнопкой мыши по графикам в правой части экрана. В контекстном меню выберите команду **Сохранить как** и введите имя, например, C:\PerfCounters.html.

К пункту 2 задания — использование файла HTML для создания журнала:

1. В меню **Пуск | Программы | Администрирование** откройте консоль **Источники данных (ODBC)**. В открывшемся окне перейдите на вкладку **Системный DSN** и нажмите кнопку **Добавить**.
2. В окне выбора драйвера выберите драйвер **SQL Server** и нажмите кнопку **Готово**.
3. На первом экране настройки свойств SQL Server введите имя источника данных (например, `PerfAW`) и выберите ваш локальный сервер SQL Server 2005.

4. На следующем экране оставьте параметры аутентификации, предлагаемые по умолчанию (**Проверка подлинности учетной записи Windows NT**).
5. На следующем экране установите флажок **Использовать по умолчанию базу данных** и выберите базу данных AdventureWorks. Нажмите кнопку **Далее**, а затем **Готово**. В последнем окне настройки драйвера ODBC нажмите кнопку **Проверить источник данных**, а потом — кнопку **OK**.
6. В консоли **Производительность** раскройте узел **Журналы и оповещения производительности**. Щелкните правой кнопкой мыши по контейнеру **Журналы счетчиков** и в контекстном меню выберите **Новые параметры журнала из**. В окне выбора файла введите имя созданного вами файла (C:\PerfCounters.html) и нажмите кнопку **Открыть**, а затем нажмите кнопку **OK** в окне предупреждения.
7. В окне **Новые параметры журнала** введите имя создаваемого журнала (например, Log1) и нажмите **OK**. Откроется окно свойств журнала.
8. На вкладке **Общие** в поле **От имени** введите имя вашей учетной записи и при помощи кнопки **Задать пароль** определите пароль.
9. Перейдите на вкладку **Файлы журналов** и в списке **Тип файла журналов** выберите **База данных SQL**, а затем нажмите кнопку **Настроить**.
10. В открывшемся окне настройки в списке **DSN системы** выберите созданный вами источник данных PerfAW и нажмите кнопку **OK**.
11. Перейдите на вкладку **Расписание** и в группе **Запуск журнала** установите переключатель в положение **Вручную**. Нажмите кнопку **OK**, чтобы закрыть окно свойств журнала с сохранением внесенных изменений.

К пункту 3 задания — запуск журнала из командной строки:

1. Для запуска журнала выполните в командной строке команду:

```
logman start Log1
```

2. Запустите пакет CopyPerson.dtsx на выполнение.

3. Выполните команду:

```
logman stop Log1
```

чтобы остановить сбор данных.

К пункту 4 задания — анализ собранных данных в Excel:

1. Запустите Microsoft Excel. В меню **Данные** выберите **Импорт внешних данных | Создать запрос**.

2. В открывшемся окне **Выбор источника данных** в списке **Базы данных** выберите созданный вами источник данных **PerfAW** и нажмите кнопку **OK**.
3. В списке таблиц выберите таблицы **CounterData** и **CounterDetails**, переместите их со всеми столбцами в список **Столбцы запроса** справа и нажмите кнопку **Далее**.
4. На экране отбора данных выберите столбец **CounterName**, в списке операторов выберите "равно", а в списке значений — **% процент загруженности процессора**. Дважды нажмите кнопку **Далее**.
5. На последнем экране убедитесь, что переключатель установлен в положение **Вернуть данные в Microsoft Office Excel** и нажмите кнопку **Готово**. В окне выбора места вставки убедитесь, что выбрана ячейка **A1** на имеющемся листе и нажмите кнопку **OK**.
6. В созданной таблице выделите все значения в столбцах **CounterDateTime** и **CounterValue** и в меню **Вставка** выберите **Диаграмма**. В списке типов диаграмм выберите **График**, первый шаблон диаграммы (с комментарием "График отображает развитие процесса во времени или по категориям") и нажмите кнопку **Далее**.
7. На втором шаге мастера диаграмм оставьте значения, предлагаемые по умолчанию, и нажмите кнопку **Далее**.
8. На следующем шаге мастера на вкладке **Легенда** снимите флажок **Добавить легенду** и нажмите кнопку **Далее**.
9. На последнем шаге мастера установите переключатель в положение **На отдельном листе** и нажмите кнопку **Готово**. Будет создан новый лист с диаграммой по значениям счетчика **% процент загрузки процессора**.
10. Щелкните правой кнопкой мыши по линии на графике и в контекстном меню выберите **Добавить линию тренда**. В открывшемся окне выберите тип линии тренда **Линейная** и нажмите кнопку **OK**. На диаграмму будет добавлена линия тренда.

## 11.5. Оптимизация работы SQL Server

### 11.5.1. Общие вопросы оптимизации работы SQL Server

В предыдущих разделах вы познакомились с основными средствами мониторинга производительности SQL Server 2005. Однако на практике часто нужно не только отслеживать производительность работы SQL Server, но и повышать ее.

Сразу скажем, что производительность SQL Server — предмет очень комплексный, на который оказывает влияние множество факторов. При этом считается, что на 80% производительность приложения зависит от его архитектуры (которая определяется разработчиком, а не администратором). По опыту автора, наибольшее влияние на производительность работы SQL Server оказывает один-единственный фактор — размер наиболее часто используемых таблиц в базе данных. Если они небольшие, то обычно проблем с производительностью не возникает. Если же размер одной таблицы составляет десятки гигабайт, то любые приемы оптимизации производительности будут бесполезны.

Конечно, чтобы рабочие таблицы оставались небольшими, необходимо, чтобы старая информация удалялась из них регулярно. Чаще всего используются для способа реализации этой задачи:

- старая информация с определенной периодичностью (каждую ночь, раз в неделю, месяц, квартал и т. п. — в зависимости от скорости накопления информации) переносится пакетами SSIS в специальную базу данных-хранилище (Data Warehouse). Затем информация, перенесенная в хранилище данных, используется для изготовления отчетов и для создания кубов OLAP (если они используются на предприятии). Это наиболее рекомендуемый способ организации работы с данными на предприятии;
- когда заканчивается определенный период времени (обычно год), старая база данных закрывается, а часть информации из нее (остатки по счетам, незавершенные операции и т. п.) переносится в новую базу данных.

Конечно, и первый, и второй способы требуют обязательного участия разработчиков. Самостоятельно реализовать перенос остатков в конце года или переделать все отчеты администратору (у которого обычно нет полной документации по структуре базы данных и механизмам работы приложения) бывает очень сложно. Однако такое разделение архивных и текущих данных — вещь абсолютно необходимая, и поэтому администратору следует потребовать реализацию соответствующих механизмов от разработчика уже при принятии задачи в эксплуатацию.

Однако кроме архитектуры приложения, которая от администратора обычно не зависит, существуют и другие средства оптимизации производительности. Про те возможности, которые находятся в распоряжении администратора, будет рассказано в следующих разделах.

## 11.5.2. Оптимизация операционной системы для работы с SQL Server 2005

Возможностей оптимизации операционной системы Windows для работы с SQL Server 2005 не так много. Поскольку разработчики Windows и SQL

Server работают в одной фирме, то для оптимизации изначально сделано очень многое. Если на сервере Windows работает только SQL Server 2005, то практически все ресурсы сервера будут отданы именно серверу. Отметим лишь несколько моментов.

По опыту автора, максимальное влияние на оптимизацию Windows для работы SQL Server 2005 оказывает единственный параметр, который спрятан в достаточно неожиданном месте. Добраться до него можно так:

- открыть папку **Сетевые подключения** (Network Connections) из меню **Пуск | Настройка**;
- далее открыть свойства любого сетевого адаптера (параметр все равно один для всего компьютера);
- на вкладке **Общие** (General) выделить компонент **Служба доступа к файлам и принтерам сетей Microsoft** (File and Print Sharing for Microsoft Networks) и нажать кнопку **Свойства** (Properties). Откроется окно, аналогичное представленному на рис. 11.9.

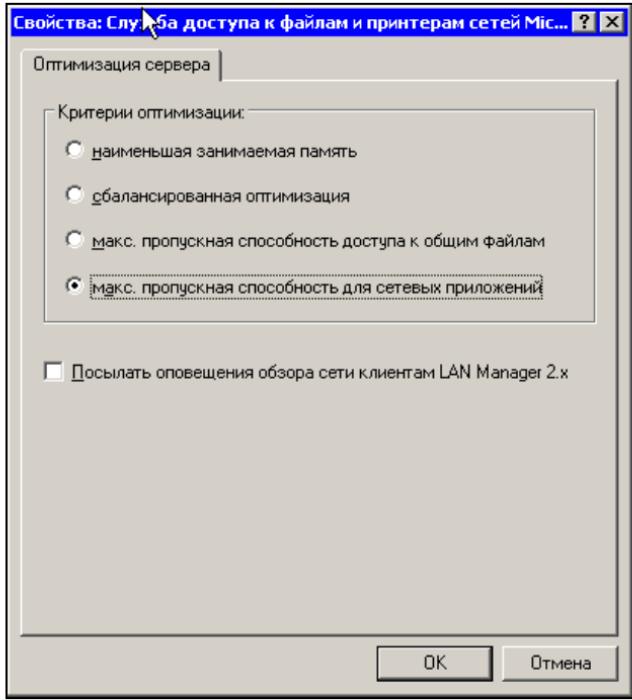


Рис. 11.9. Параметры службы **Сервер** на компьютере Windows

В этом окне представлены настройки для службы **Сервер** (Server). При этом при помощи переключателей в группе **Критерии оптимизации** (Server

optimization) вы можете определить, как именно будет распределяться оперативная память на вашем сервере:

- наименьшая занимаемая память** (minimize memory used) — это положение переключателя означает, что максимум оперативной памяти будет выделяться обычным приложениям, запускаемым пользователем. Это значение имеет смысл использовать в следующих ситуациях:
  - когда сервер используется как рабочая станция (т. е. как настольный компьютер);
  - когда главная задача сервера — обеспечивать поддержку задач, реализованных как пользовательские приложения, а не как службы (например, у вас может быть сервер, предназначенный только для выполнения пакетов SSIS);
- сбалансированная оптимизация** (balance) — это компромиссный вариант. Память будет равномерно распределяться между пользовательскими приложениями и серверными службами;
- максимальная пропускная способность доступа к общим файлам** (maximize data throughput for file sharing) — максимум оперативной памяти будет выделен под файловый кэш. Это значение оптимально для работы файлового сервера и сервера печати. По умолчанию для серверов Windows 2000 и Windows 2003 устанавливается именно это значение;
- максимальная пропускная способность для сетевых приложений** (maximize data throughput for network applications) — максимум оперативной памяти будет выделен серверным службам, например, SQL Server, Oracle, Exchange Server и т. п. При установке SQL Server и Exchange Server переключатель автоматически переводится в это положение (при установке Oracle — нет).

Надо сказать, что, например, скорость копированияния по сети большого объема файлов при переводе переключателя в третье положение может возрастать очень сильно (явление, которое автор неоднократно наблюдал при установке учебного класса). Точно так же этот переключатель оказывает большое влияние и на скорость работы SQL Server 2005 и Exchange Server. Отметим, что он по умолчанию устанавливается в нужное положение, однако знать про этот параметр необходимо.

Дополнительные параметры настройки производительности Windows доступны в свойствах компьютера, если на вкладке **Дополнительно** нажать кнопку **Параметры**. Откроется окно **Параметры быстродействия**. Основные параметры настраиваются на вкладке **Дополнительно** (рис. 11.10). Для оптимизации производительности под SQL Server 2005 переключатели должны быть установлены так, как на рис. 11.10. Отметим, что переключатель в

группе **Использование памяти** работает с тем же параметром реестра HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SessionManager\MemoryManagement\LargeSystemCache, что и настройки для службы Server, рассмотренные ранее.

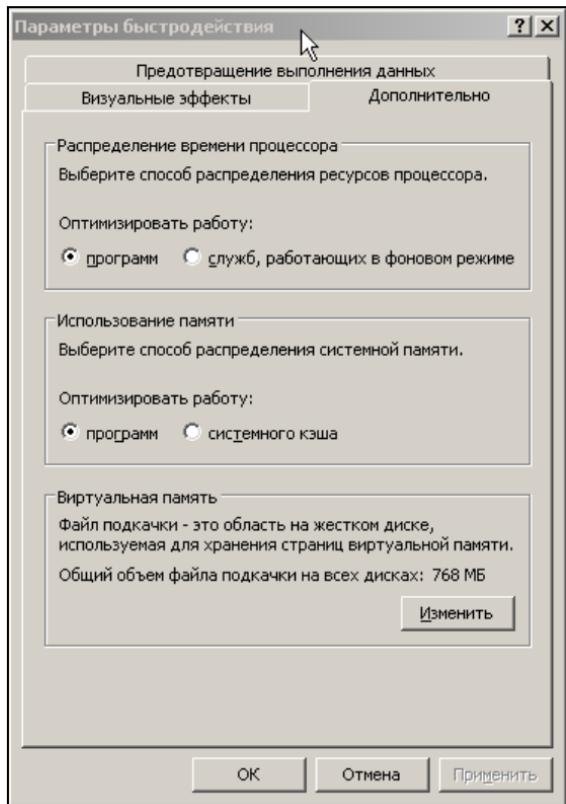


Рис. 11.10. Параметры быстродействия Windows

Отметим еще несколько достаточно очевидных моментов:

- все лишние службы, которые на сервере не используются, лучше отключать. Это хорошо и с точки зрения производительности, и с точки зрения защищенности сервера;
- лучше не запускать утилиты администрирования на самом сервере (администраторы очень любят для этого использовать терминальные средства, например, Remote Desktop Connection — удаленное управление рабочим столом). Сами по себе средства администрирования SQL Server достаточно ресурсоемки, и лучше всего запускать их только на рабочей станции администратора;

- конечно, категорически не рекомендуется использоваться на сервере заставки OpenGL;
- антивирусное программное обеспечение может очень сильно снизить производительность работы компьютера, на котором установлен SQL Server 2005. В большинстве ситуаций для SQL Server антивирусные программы абсолютно не нужны.

### 11.5.3. Общая оптимизация работы SQL Server 2005 с помощью Best Practices Analyzer

Очень часто администратору, который знакомится с задачей, представленной ему разработчиком, хочется использовать автоматизированное программное средство, которое анализировало бы SQL Server и базу данных приложения и автоматически определило проблемы, в том числе те, которые могут привести к снижению производительности. Такое средство есть. Оно находится в свободном доступе на Web-сайте Microsoft и называется Best Practices Analyzer Tool (BPA).

Его можно использовать и разработчикам для анализа своих собственных баз данных, и администраторам просто для анализа рабочих баз данных.

Основное назначение этого средства — проведение проверки SQL Server и выбранных вами баз данных на соответствие определенным правилам. Встроенных правил в Best Practices Analyzer около 80. Нужный набор правил, который будет применяться для анализа определенной базы данных, вы можете выбрать сами (рис. 11.11).

Правила распределены по нескольким категориям:

- **Backup and Recovery** (Резервное копирование и восстановление) — проверка с использованием этого правила возвращает информацию о том, для каких баз данных резервное копирование не выполнялось или выполнялось давно, какие были ошибки при резервном копировании, проводятся ли резервное копирование системных баз данных, есть ли базы данных с режимом восстановления **Simple** и т. п.;
- **Configuration Options** (Настройки конфигурации сервера) — это правило определяет, нет ли неоптимальных настроек с точки зрения производительности и безопасности;
- **Database Design** (Проект базы данных) — есть ли таблицы без первичного ключа и соблюдены ли правила именования объектов;
- **Database Administration** (Администрирование базы данных) — создается ли статистика в автоматическом режиме, какие индексы необходимо дефрагментировать, достаточно ли места на дисках для файлов баз данных и журналов транзакций, правильно ли настроены параметры баз данных;

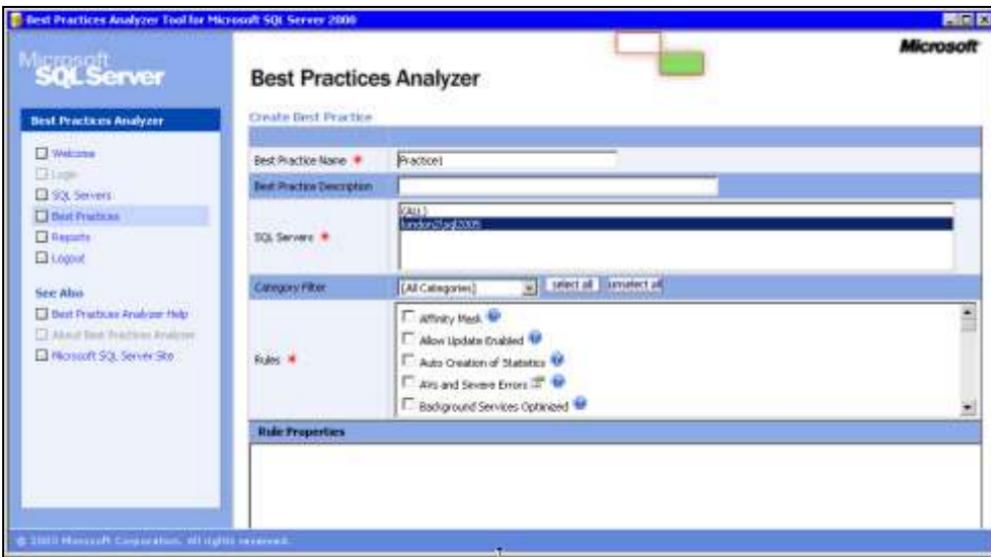


Рис. 11.11. Окно выбора правил в Best Practices Analyzer

- Deprecation** (Нежелательное) — проверка синтаксиса объектов на использование нерекомендованных команд, функций, хранимых процедур;
- General Administration** (Общее администрирование сервера) — какие проблемы возникали на используемых серверах, какие важные ошибки на них обнаружены, нет ли пользовательских объектов в системных базах данных и т. п.;
- TSQL** — применение нерекомендованных возможностей Transact-SQL в базах данных;
- Yukon Readiness** (готовность к Юкону) — группа правил, относящихся к новым возможностям SQL Server 2005;
- Full-Text** (Полнотекстовое) — ошибки при реализации полнотекстового индексирования;
- Generic** (Общее) — в этом наборе правил вы можете определить только требования к префиксам и суффиксам для объектов баз данных.

К сожалению, на момент написания этой книги Best Practices Analyzer для SQL Server 2005 пока еще не появился. Разработчики SQL Server 2005 рекомендуют пока использовать Best Practices Analyzer для SQL Server 2000 (поскольку набор правил во многом схож). Однако некоторые правила при применении к SQL Server 2005 вызывают ошибку в работе Best Practices Analyzer. Такие правила придется исключить и произвести проверку заново.

## 11.5.4. Оптимизация подключений к SQL Server 2005

Первый уровень оптимизации приложений с использованием SQL Server 2005 — это оптимизация подключений к серверу. Подключаться к SQL Server можно разными способами. Чаще всего используются три варианта:

- подключение по OLE DB — наиболее рекомендуемый и современный вариант. Если клиентская часть реализована под Windows (что бывает в большинстве случаев), использование подключений по OLE DB — это и самый быстрый способ;
- подключение по ODBC — унаследованный, но до настоящего времени очень популярный способ;
- подключение с использованием BDE (Borland Database Engine) — обычно используется в приложениях, написанных на языке Delphi.

Быстрее всего работают подключения по OLE DB. Причина проста: при подключениях такого типа полностью используются возможности COM-технологий. Кроме того, по сравнению с ODBC и BDE приходится производить меньшее количество преобразований. Выигрыш в скорости особенно заметен в тех случаях, когда с сервера на клиентский компьютер передается большое количество информации (например, при создании больших отчетов).

Конечно, способ подключения выбирается разработчиками клиентских приложений (а они чаще всего выбирают наиболее привычный для себя вариант, а не самый быстрый). Однако существует прием, который позволяет заменить подключение по ODBC на подключение по OLE DB, при этом сделать это абсолютно прозрачно для приложения.

Предположим, что у вас есть системный источник данных ODBC, который называется `SQL1`, настроенный на подключение к базе данных SQL Server 2005. Именно этот источник данных ODBC используется клиентским приложением. Переключить его на использование OLE DB можно следующим образом.

Первое, что нужно сделать, — создать на диске файл подключения по OLE DB (файл UDL — User Data Link). Сгенерировать его в автоматическом режиме можно так:

- создать на диске любой пустой файл (для этого, например, можно щелкнуть правой кнопкой мыши по пустому пространству в папке в Проводнике Windows и в контекстном меню выбрать **Создать | Текстовый документ**);
- изменить у этого файла расширение на `udl`. Естественно, при этом должен быть включен показ расширений для известных типов файлов. Включить

его можно на вкладке **Вид** в окне **Свойства папки** (меню **Сервис** в Проводнике);

- обратите внимание, что после изменения расширения имени файла значок для данного файла в Проводнике изменился. Правда, файл пока пустой. Чтобы настроить в нем параметры подключения по OLE DB, достаточно щелкнуть по нему два раза мышью. Откроется окно настройки свойств подключения OLE DB;
- на вкладке **Поставщик данных** необходимо выбрать нужный тип источника данных (в данном случае — **Microsoft OLE DB Provider for SQL Server**). Затем на вкладке **Подключение** необходимо настроить параметры подключения к SQL Server 2005 (например, так, как представлено на рис. 11.12). После этого рекомендуется нажать кнопку **Проверить подключение**, чтобы протестировать возможность установки соединения, и нажать кнопку **OK**, чтобы закрыть свойства файла с сохранением изменений.

Если открыть этот файл в Блокноте, то в нем будет готовая строка подключения, которую, например, можно использовать при создании клиентских приложений. Однако вам нужен сам файл.

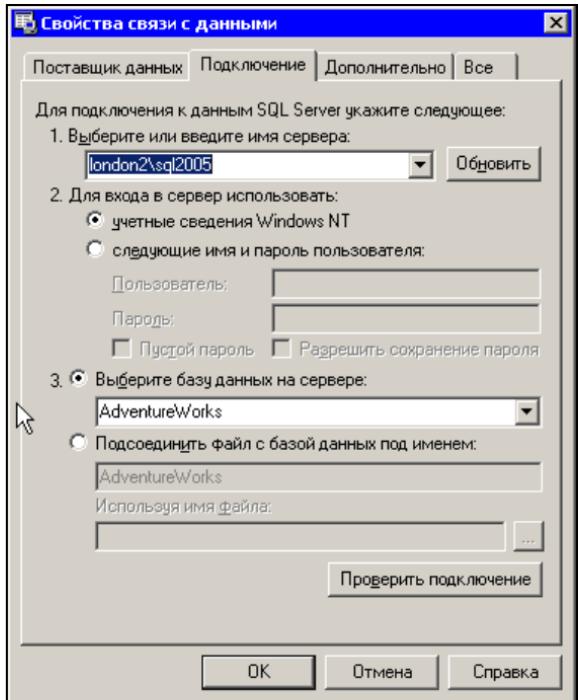


Рис. 11.12. Настройка подключения по OLE DB

Следующее, что нужно сделать, — использовать этот файл для перенастройки источника данных ODBC. В соответствии с заданием, у вас есть системный источник данных ODBC (System DSN), который называется `SQL1` и обращается к тому же серверу SQL Server 2005 с тем же режимом аутентификации. "Исправить" этот источник, чтобы он работал по OLE DB, можно очень просто:

- откройте редактор реестра (набрав в командной строке `regedit`);
- раскройте раздел `HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI\имя_вашего_источника_данных` (в данном случае `SQL1`);
- вместо файла DLL для параметра **Driver** (Драйвер) укажите созданный файл UDL (рис. 11.13).

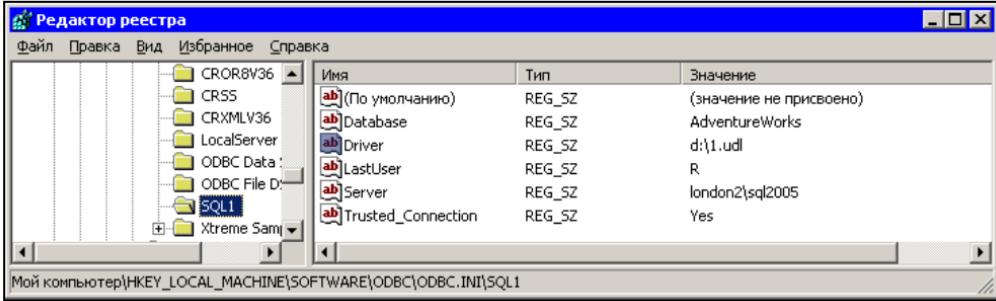


Рис. 11.13. Замена подключения по ODBC на подключение по OLE DB

После этого ваше приложение, обращаясь к этому источнику данных ODBC, на самом деле будет работать по OLE DB. В некоторых ситуациях это может существенно ускорить скорость передачи данных между сервером и клиентом.

Отметим еще некоторые моменты, связанные с настройкой клиентских подключений к SQL Server:

- в SQL Server 2005 предусмотрена специальная надстройка над OLE DB — SQL Native Client с большим количеством новых возможностей. Применение одной из таких возможностей, которая называется MARS (Multiple Access Result Set — множественные активные наборы результатов), в некоторых случаях позволит очень сильно повысить производительность при работе клиента с SQL Server. При использовании технологии MARS клиент может работать в асинхронном режиме и отправлять следующие запросы на SQL Server, не дожидаясь, пока придет ответ на запросы, отправленные ранее;
- если клиентов у вас очень много (счет идет на сотни), и время установки соединения становится неприемлемым (типичная ситуация, например, для

call-центра), то имеет смысл подумать о промежуточном сервере для обслуживания подключений пользователей. Обычно такой сервер работает с пулем открытых подключений к серверу SQL Server (технология называется *connection pooling*) и выдает свободные подключения пользователям.

### 11.5.5. Оптимизация системы индексов

Одна из возможностей повышения производительности, которая может дать сильный эффект, — это оптимизация системы индексов. Конечно, необходимые индексы в базе данных в идеале должны реализовывать разработчики, но практика показывает, что для многих задач это не так. Чаще всего разработчики создают только служебные индексы (которые, например, обеспечивают уникальность значений в определенном столбце таблицы), особо не задумываясь над применением индексов для оптимизации запросов. В то же время во многих задачах создание нескольких дополнительных индексов может привести к повышению производительности выполнения запросов в разы. Отметим, что оптимизацию системы индексов может производить и администратор. При этом функциональность приложения никак не пострадает.

В теории при оптимизации системы индексов следовало бы собрать информацию о наиболее важных и частых операциях, которые выполняют пользователи, посмотреть планы выполнения запросов и вручную определить, какие индексы могут потребоваться дополнительно. Однако на практике при наличии нескольких сотен таблиц и представлений в рабочей базе данных и нескольких десятков стандартных запросов, которые выполняют пользователи (а код многих из них спрятан за хранимыми процедурами), рассчитывать оптимальную систему индексов вручную — это непроизводительная траты времени. Намного удобнее провести такой анализ в автоматическом режиме средствами программы Database Tuning Advisor. Это средство использует собранную информацию о выполненных запросах для того, чтобы определить наилучший набор индексов для базы данных.

Первое, что нужно сделать при применении Database Tuning Advisor, — собрать информацию о запросах пользователей. Проще всего это сделать средствами SQL Server Profiler (см. разд. 11.2.3), выбрав шаблон **Tuning**. Правильнее будет собирать информацию о работе пользователей в течение продолжительного времени, например рабочего дня. Удобнее всего производить протоколирование в таблицу на сервере SQL Server, но при желании можно использовать и файлы трассировки на диске, и просто набор команд Transact-SQL в файле скрипта с расширением sql.

После того как исходная информация о командах, выполняемых пользователями, готова, можно запускать Database Tuning Advisor (меню **Пуск | Программы | Microsoft SQL Server 2005 | Performance Tools | Database Engine**).

**Tuning Advisor**). Первое, что вам потребуется сделать, — подключиться к серверу, для которого будет производиться оптимизация индексов. Database Tuning Advisor будет автоматически запрашивать с этого сервера необходимую служебную информацию. Кроме того, этот сервер будет использоваться для хранения временной информации Database Tuning Advisor, поэтому выполнение анализа лучше планировать на нерабочее время.

После подключения к серверу перед вами откроется интерфейс Database Tuning Advisor. В левой части расположено окно **Session Monitor** (Монитор сеанса) со списком сеансов оптимизации. Зеленым флагом при этом будут помечены те сеансы, для которых завершен анализ и созданы рекомендации. Из контекстного меню для объектов сеанса вы можете открыть сеанс, скопировать его, переименовать, удалить, запустить анализ и т. п. Можно использовать и еще одну интересную возможность: если воспользоваться командой **Import Session Definition** (Импортировать определение сеанса), то можно передать Database Tuning Advisor файл с расширением xml с параметрами сеанса, включая, например, информацию об индексах, влияние которых на производительность при их гипотетическом создании можно оценить. В результате у вас появляется возможность использовать средства Database Tuning Advisor для того, чтобы проверить, какое реальное влияние может оказывать конкретный индекс, который вы планируете создать.

Справа в Database Tuning Advisor расположено основное окно с информацией о текущем сеансе. Изначально в нем есть две вкладки: **General** и **Tuning Options** (Параметры оптимизации). Первое, что необходимо сделать, — выбрать протокол рабочей нагрузки (*workload*). Им может быть таблица трассировки на SQL Server (созданная профилировщиком) или файл одного из трех типов:

- двоичный файл трассировки, созданный профилировщиком (для таких файлов используется расширение trc);
- файл трассировки в формате XML (собранный протокол можно сохранить в этом формате, используя команду **File | Save As** (Файл | Сохранить как) в профилировщике);
- просто текстовый файл с командами Transact-SQL (для него используется расширение sql).

Обратите внимание на параметр **Database for workload analysis** (База данных для анализа рабочей нагрузки). Он определяет базу данных, в которой будут создаваться временные объекты Database Tuning Advisor. Конечно, такие временные объекты не должны создаваться ни в базе данных master (что предлагается по умолчанию), ни в рабочей базе данных. Лучше всего использовать для временных объектов тюнинга новую базу данных, специально созданную для этой цели.

В нижней части вкладки **General** вы можете выбрать базы данных и таблицы, для которых будет производиться анализ системы индексов (чтобы открыть список таблиц, нужно щелкнуть на стрелку в строке для базы данных). Отличительной особенностью Database Tuning Advisor по сравнению с Index Tuning Wizard в SQL Server 2000 является то, что теперь можно производить анализ сразу нескольких баз данных.

Вкладка **Tuning Options** предназначена для настройки параметров тюнинга. В большинстве случаев оптимальными параметрами будут следующие:

- не ограничивать время анализа (снять флажок **Limit tuning time**). Как уже говорилось, анализ лучше планировать на нерабочее время. Кроме снятия нагрузки с рабочей базы данных такое решение одновременно позволит отвести на анализ столько времени, сколько потребуется Database Tuning Advisor. Конечно же, снятие ограничения на время обеспечит максимальное качество анализа;
- анализировать возможность применения индексов и индексированных представлений (установить переключатель **Physical Design Structures to use in database** (Структуры физического проекта для использования в базе данных) в положение **Indexes and indexed views**). Это обеспечит максимальную гибкость при проведении анализа. Все остальные положения переключателя (анализировать применение только индексов, только некластерных индексов, только индексированных представлений или только существующие индексы без выработки предложений по созданию новых) ограничивают варианты, которые будут рассматриваться при анализе;
- использовать полные возможности секционирования (установить переключатель **Partitioning strategy to employ** (Стратегия секционирования для применения) в положение **Full partitioning**);
- не сохранять какие-либо существующие физические структуры хранения (установить переключатель **Physical Design Structures to keep in database** (Физические структуры данных для сохранения в базе данных) в положение **Do not keep any existing PDS**). При этом будут генерироваться предложения не только по созданию новых индексов, но и по удалению существующих, если они практически не используются. Информация о том, какие индексы не используются и только замедляют операции по изменению данных, может оказаться очень полезной, а немедленно удалять индексы никто вас не заставляет.

После того как все настройки на этой вкладке произведены, можно нажать кнопку **Start Analysis** (Начать анализ) на панели инструментов. Если протокол рабочей нагрузки большой, а время для проведения анализа не ограничено, то для завершения этой операции может потребоваться несколько часов — еще один аргумент в пользу того, чтобы запускать его на ночь перед

уходом с работы. Наблюдать за ходом выполнения вы можете на вкладке **Progress** (Прогресс).

После того как анализ завершен, откроется вкладка **Recommendations** (Рекомендации). На ней будет представлен список объектов (индексов, индексированных представлений, разделов), которые нужно создать или удалить. При этом названия объектов, которые нужно удалить, будут зачеркнуты. Однако не стоит спешить внедрять эти рекомендации сразу же. Намного правильнее будет вначале ознакомиться с результатами анализа на вкладке **Reports** (Отчеты). Кроме общей статистики анализа, на этой вкладке представлены отчеты, при помощи которых можно получить информацию:

- о том, какие запросы выполнялись чаще всего (отчет **Event frequency report**);
- насколько выигрывает каждый запрос в производительности и какие индексы он использует или будет использовать в текущей и предлагаемой конфигурации индексов (отчеты **Statement-index relations report (current)** и **Statement-index relations report (recommended)**);
- о системе индексов в текущей и рекомендованной конфигурации (отчеты **Index detail report (current)** и **Index detail report (recommended)**);
- какие индексы используются активно, а какие совсем не используются (отчеты **Index usage report**).

При помощи других отчетов можно получить множество другой полезной информации, например, к каким представлениям, таблицам, столбцам в таблицах чаще всего выполнялись обращения, насколько улучшится производительность каждого запроса и т. п.

Если после изучения результатов анализа вы придетете к мнению, что предложенные рекомендации имеет смысл воплотить в жизнь, то в вашем распоряжении — меню **Actions** (Действия) Database Tuning Advisor. Вы можете использовать три пункта этого меню:

- Apply recommendations** (Применить рекомендации) — просто выполнить команды на создание или удаление объектов, немедленно или по расписанию, в зависимости от вашего выбора;
- Save recommendations** (Сохранить рекомендации) — рекомендации будут сохранены в файле скрипта. При этом вы сможете выбрать, использовать ли команды на создание и удаление объектов в оперативном режиме (с параметром **ONLINE**) или в автономном режиме (**OFFLINE**);
- Evaluate recommendations** (Оценить рекомендации) — это значит создать новый сеанс анализа, в котором предлагаемая конфигурация индексов, индексированных представлений и разделов будет считаться действующей, и

проводите анализ типа "а что, если ...".

Database Tuning Advisor намного эффективнее своего предшественника Index Tuning Wizard. Часто на одной и той же базе данных для одной и той же нагрузки они дают совершенно разные рекомендации. Однако полностью полагаться на Database Tuning Advisor не стоит — ситуации, когда предлагаемые им рекомендации не оптимальны, иногда встречаются. Однако Database Tuning Advisor удобно использовать для получения информации, которая послужит отправной точкой для дальнейшего анализа, и для резкого сокращения трудоемкости при оптимизации системы индексов.

У Database Tuning Advisor есть и консольный вариант, представленный исполняемым файлом dta.exe. Возможности этой утилиты совпадают с возможностями графического варианта Database Tuning Advisor, поэтому консольный вариант используется в основном для автоматизации сеансов тюнинга. Например, если написать соответствующую команду на запуск dta.exe и сохранить ее в файле bat, то такой файл можно использовать для запуска сеанса анализа в ночное время.

Отметим и некоторые проблемы, которые связаны с применением Database Tuning Advisor:

- Database Tuning Advisor иногда отказывается запускаться на относительно маломощных компьютерах (на которых, тем не менее, нормально работает SQL Server 2005). При попытке подключиться к серверу он может выдать ошибку 2812 "Could not find stored procedure 'msdb..sp\_DTA\_help\_session'" (Не могу найти хранимую процедуру 'msdb..sp\_DTA\_help\_session'). Такой хранимой процедуры действительно нет, но только потому, что Database Tuning Advisor должен создать ее вместе с другими необходимыми объектами в базе данных msdb. На маломощных компьютерах этого не происходит;
- полные возможности Database Tuning Advisor доступны только в редакции SQL Server 2005 Enterprise Edition. Во всех остальных редакциях некоторые возможности будут недоступны.

## **Задание для самостоятельной работы 11.3. Оптимизация системы индексов**

### **Задание:**

Используйте собранный вами в работе 11.1 протокол выполнения команд (файл C:\AdventureWorksTrace.trc) для анализа системы индексов средствами Database Tuning Advisor. Сохраните предлагаемые рекомендации в файле C:\IndexTuning.sql.

## Решение:

1. Запустите Database Tuning Advisor (меню Пуск | Программы | Microsoft SQL Server 2005 | Performance Tools | Database Engine Tuning Advisor) и подключитесь к своему локальному серверу SQL Server 2005.
2. На вкладке **General** окна Database Tuning Advisor выберите созданный вами файл C:\AdventureWorksTrace.trc, а в списке баз данных — базу данных AdventureWorks.
3. На вкладке **Tuning Options** снимите флажок **Limit Tuning Time** и установите соответствующие переключатели в положения **Indexes and indexed views**, **Full partitioning**, **Do not keep any existing PDS**. Нажмите кнопку **Start Analysis** на панели инструментов.
4. Дождитесь окончания анализа, затем перейдите на вкладку **Reports** и просмотрите созданные отчеты.
5. В меню **Actions** выберите команду **Save Recommendations** и сохраните сгенерированные команды SQL по выполнению рекомендаций в файле C:\IndexTuning.sql.

## 11.5.6. Дефрагментация индексов и таблиц

Кроме создания дополнительных индексов и удаления ненужных, в некоторых ситуациях значительный выигрыш в производительности можно также получить, дефрагментировав индексы и таблицы в базе данных. Конечно, такой метод будет наиболее действенным для давно существующих баз данных, в которых производилось большое количество операций по изменению и удалению данных.

Сразу отметим, что отдельной операции по дефрагментации таблиц в SQL Server 2005 не предусмотрено. Данные в таблицах автоматически упорядочиваются по кластерному индексу (он может быть для таблицы только один). Поэтому дефрагментация кластерного индекса автоматически приведет к дефрагментации таблицы.

Прежде чем бороться с фрагментацией, рекомендуется оценить ее уровень. Сделать это можно при помощи команды DBCC SHOWCONTIG. Ее можно запускать с разным набором параметров. Если выполнить эту команду совсем без параметров, то она вернет информацию о степени фрагментации первого индекса для каждой из таблиц в текущей базе данных. Наиболее часто используемый вариант синтаксиса для этой команды выглядит как DBCC SHOWCONTIG (*имя\_таблицы, имя\_индекса*). Например, в базе данных AdventureWorks можно выполнить такую команду:

```
DBCC SHOWCONTIG ('HumanResources.Employee', 'PK_Employee_EmployeeID');
```

Расшифруем то, что возвращает эта команда:

- **Pages scanned** (Просканировано страниц) — количество страниц в базе данных, используемых индексом;
- **Extents switches** (Переключения экстентов) — сколько переходов между экстентами пришлось выполнить при просмотре страниц индекса. В идеале страницы индекса должны идти друг за другом в смежных экстентах, поэтому в идеальной ситуации этот параметр не должен превышать количество страниц, деленное на 8. Но такая ситуация встречается редко;
- **Avg. Pages per Extent** (Средние страницы на экстент) — сколько просмотренных страниц в среднем приходится на экстент. В идеале это значение должно быть равно 8 (или близко к нему). Если это значение существенно меньше, то страницы одного индекса распределены между разными экстентами. Между ними может быть пустое пространство или страницы других объектов в базе данных, но в любом случае такой вариант означает фрагментацию индекса;
- **Scan Density [Best Count: Actual Count]** (Плотность сканирования [Лучший показатель: Реальный показатель]) — отношение идеально возможного количества переходов между экстентами к реальному. Чем ближе этот показатель к 100%, тем лучше;
- **Logical Scan Fragmentation** (Фрагментация логического сканирования) — в идеале все страницы индекса должны идти строго друг за другом не только по номеру, но и по физическому расположению в файле базы данных (в соответствии с таблицей IAM — Index Allocation Map, карта размещения индекса). Этот показатель возвращает, какой процент страниц индекса (из просмотренных) нарушает это требование (и, соответственно, на сколько процентов фрагментирован индекс);
- **Extent Scan Fragmentation** (Фрагментация сканирования экстентов) — это почти то же самое, только оцениваются не страницы индекса, которые идут не по порядку, а экстенты, принадлежащие этому индексу, которые идут не друг за другом. Этот параметр оказывает меньшее влияние на скорость работы (и обычно он сам существенно меньше, чем параметр **Logical Scan Fragmentation**);
- **Avg. Bytes free per page** (Средний размер пустого пространства на страницах) — чем выше этот показатель, тем хуже. Как правило, большое значение этого показателя возникает после удаления большого количества данных из таблицы;
- **Avg. Page Density** (Средняя плотность страницы) — этот показатель, обратный предыдущему, определяет, на сколько в среднем заполнена каждая страница индекса. Чем выше значение, тем лучше.

Основные показатели, на которые следует обращать внимание — это **Scan Density**, **Logical Scan Fragmentation** и **Avg. Page Density**. Плотность сканирования и средняя плотность страницы должны быть максимально близки к 100%, а фрагментация логического сканирования — к 0. Если уровень фрагментации составляет 30—40% и выше, имеет смысл подумать о проведении дефрагментации.

Другая возможность оценить фрагментацию индексов — воспользоваться графическими средствами SQL Server Management Studio. Для этого нужно раскрыть в **Object Explorer** контейнер **имя\_сервера | Databases | имя\_базы данных | Tables | имя\_таблицы | Indexes**, открыть свойства для нужного индекса и перейти на вкладку **Fragmentation** (Фрагментация) (рис. 11.14). Информации по фрагментации на этой вкладке приводится меньше, но самое важное (о заполнении страниц и общая оценка фрагментации) указано.

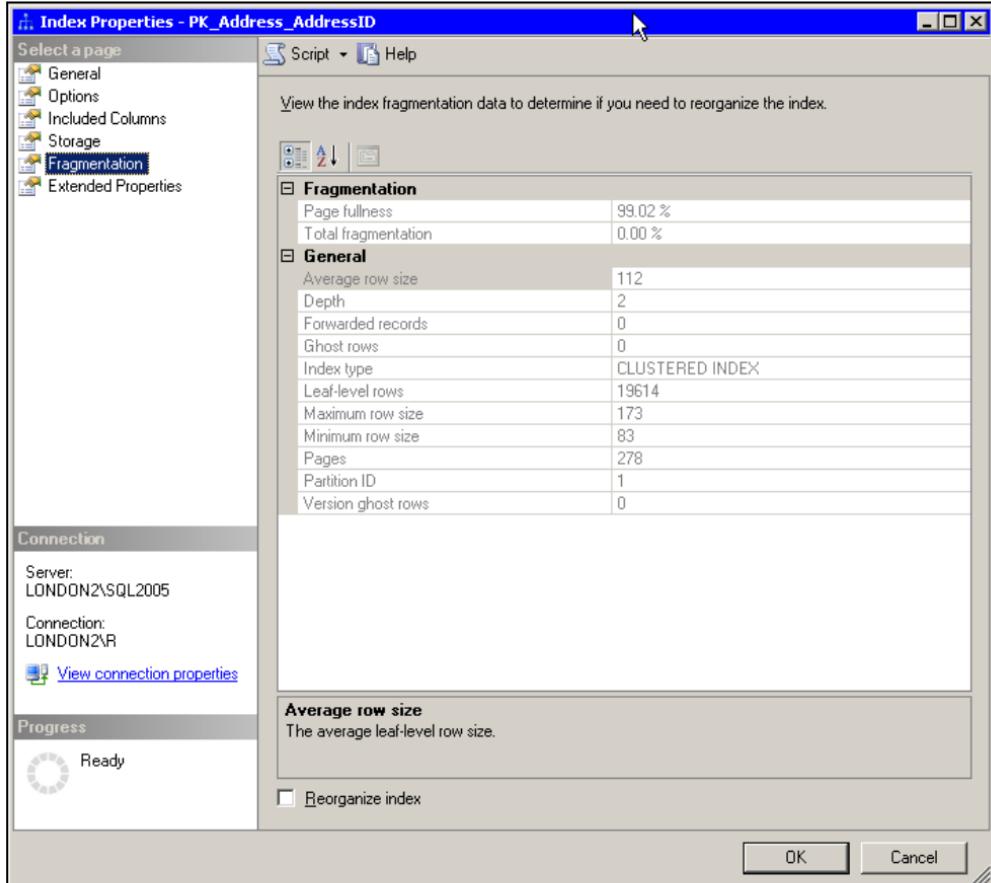


Рис. 11.14. Оценка степени фрагментации индекса из SQL Server Management Studio

Дефрагментацию индексов можно проводить разными способами.

Самый радикальный способ — воспользоваться командой `ALTER INDEX REBUILD`. Эта команда полностью удаляет существующий индекс или индексы и создает их заново, устранив всякую фрагментацию. В SQL Server 2000 эта операция выполнялась при помощи команды `DBCC DBREINDEX`. Она поддерживается и в SQL Server 2005, но только для обеспечения обратной совместимости. Рекомендуется использовать вместо нее команду `ALTER INDEX REBUILD`. Например, чтобы произвести полное перестроение проанализированного индекса `PK_Employee_EmployeeID`, можно воспользоваться командой:

```
ALTER INDEX PK_Employee_EmployeeID ON HumanResources.Employee REBUILD;
```

Можно перестроить все индексы для данной таблицы или представления:

```
ALTER INDEX ALL ON HumanResources.Employee REBUILD;
```

Отметим, что при перестроении некластерного индекса его можно отключить командой `ALTER INDEX DISABLE`, в результате места на диске для этой операции потребуется намного меньше.

Если же вы хотите произвести перестроение всех индексов для всех таблиц в базе данных, придется заняться программированием. Можно написать скрипт Transact-SQL или воспользоваться возможностями объектных моделей SMO или SQL-DMO. Например, следующий скрипт VBScript произведет перестроение всех индексов в базе данных `AdventureWorks`, подключившись к серверу `LONDON2\SQL2005` средствами аутентификации Windows:

```
Dim oServer, oDB, oTable
Set oServer = CreateObject("SQLDmo.SqlServer2")
oServer.LoginSecure = True
oServer.Connect "LONDON2\SQL2005"
Set oDB = oServer.Databases("Northwind")
For Each oTable In oDB.Tables
 If oTable.SystemObject = False Then oTable.RebuildIndexes
Next
```

Еще более интересный, но намного более сложный способ — производить перестроение только фрагментированных индексов, например, в зависимости от значения параметра **Scan Density**, возвращаемого командой `DBCC SHOWCONTIG`. Решение может выглядеть так, как представлено далее (оно было найдено в статье Тома Паллена (Tom Pullen) "Automating Reindexing In SQL Server" (Автоматическая переиндексация SQL Server) на сайте [www.sql-server-performance.com](http://www.sql-server-performance.com)).

Вначале создаем хранимую процедуру, которая будет использоваться для периодической дефрагментации фрагментированных индексов. Соответствующий код может выглядеть так:

```
CREATE PROCEDURE sp_defragment_indexes @maxfrag DECIMAL AS
-- Объявляем необходимые переменные
SET NOCOUNT ON
DECLARE @tablename VARCHAR (128)
DECLARE @execstr VARCHAR (255)
DECLARE @objectid INT
DECLARE @objectowner VARCHAR(255)
DECLARE @indexid INT
DECLARE @frag DECIMAL
DECLARE @indexname CHAR(255)
DECLARE @dbname sysname
DECLARE @tableid INT
DECLARE @tableidchar VARCHAR(255)
-- На всякий случай проверяем, что база данных пользовательская
SELECT @dbname = db_name();
IF @dbname IN ('master', 'msdb', 'model', 'tempdb')
BEGIN
 PRINT 'Эта процедура не может быть запущена для системных БД';
 RETURN
END
-- Начинаем проверку уровня фрагментации
-- Вначале объявляем курсор
DECLARE tables CURSOR FOR
SELECT convert(varchar,so.id) FROM sysobjects so JOIN sysindexes si ON
so.id = si.id WHERE so.type ='U' AND si.indid < 2 AND si.rows > 0;
-- Затем создаем временную таблицу для хранения информации о фрагментации
CREATE TABLE #fraglist (ObjectName CHAR (255), ObjectId INT, IndexName
CHAR (255), IndexId INT, Lvl INT, CountPages INT, CountRows INT,
MinRecSize INT, MaxRecSize INT, AvgRecSize INT, ForRecCount INT, Extents
INT, ExtentSwitches INT, AvgFreeBytes INT, AvgPageDensity INT,
ScanDensity DECIMAL, BestCount INT, ActualCount INT, LogicalFrag DECIMAL,
ExtentFrag DECIMAL);
-- Открываем курсор
OPEN tables
-- Для каждой таблицы в базе данных выполняем команду DBCC SHOWCONTIG
FETCH NEXT FROM tables INTO @tableidchar
WHILE @@FETCH_STATUS = 0
BEGIN
 -- Проходим по всем индексам для таблицы
 INSERT INTO #fraglist
 EXEC ('DBCC SHOWCONTIG (' + @tableidchar + ') WITH FAST, _'
 TABLERESULTS, ALL_INDEXES, NO_INFOMSGS')
 FETCH NEXT
```

```
FROM tables
INTO @tableidchar
END
-- Закрываем курсор
CLOSE tables
DEALLOCATE tables
-- Для проверки выводим информацию из временной таблицы
SELECT * FROM #fraglist
-- Теперь необходимо произвести дефрагментацию
-- Вначале опять объявляем курсор
DECLARE indexes CURSOR FOR
SELECT ObjectName, ObjectOwner = user_name(so.uid), ObjectId, IndexName,
ScanDensity FROM #fraglist f JOIN sysobjects so ON f.ObjectId=so.id
WHERE ScanDensity <= @maxfrag AND INDEXPROPERTY (ObjectId, IndexName,
'IndexDepth') > 0
-- Выводим для проверки информацию о начале дефragmentации
SELECT 'Started defragmenting indexes at ' + CONVERT(VARCHAR, GETDATE())
-- Открываем курсор
OPEN indexes
-- Проходим циклом по всем индексам
FETCH NEXT FROM indexes INTO @tablename, @objectowner, @objectid,
@indexname, @frag
WHILE @@FETCH_STATUS = 0
BEGIN
 SET QUOTED_IDENTIFIER ON
 SELECT @execstr = 'DBCC DBREINDEX (' + ""'' + RTRIM(@objectowner)
+ '.' + RTRIM(@tablename) + ""'' + ', ' + RTRIM(@indexname) + ')
WITH NO_INFOMSGS'
 SELECT 'Выполняем: ' SELECT (@execstr)
 EXEC (@execstr)
 SET QUOTED_IDENTIFIER OFF
 FETCH NEXT FROM indexes INTO @tablename, @objectowner, @objectid,
@indexname, @frag
END
-- Затем закрываем курсор
CLOSE indexes;
DEALLOCATE indexes;
-- Отчитываемся о времени завершения
SELECT 'Finished defragmenting indexes at ' + CONVERT(VARCHAR, GETDATE());
-- Удаляем времененную таблицу
DROP TABLE #fraglist;
GO
```

Если вам нужно выполнить дефрагментацию для всех индексов в интересующей базе данных, для которых значение параметра **Scan Density** меньше 80%, можно использовать команду:

```
EXEC sp_defragment_indexes 80.00;
```

Перестроение индексов — операция достаточно тяжелая. Она требует большого количества времени и места на диске, при этом пользователи не могут работать с таблицами, для которых выполняется такое перестроение. Поэтому во многих случаях имеет смысл производить не перестроение, а реорганизацию индексов при помощи команды ALTER INDEX REORGANIZE (другая устаревшая, но еще поддерживаемая команда — DBCC INDEXDEFRAG). При реорганизации индексов производится дефрагментация только страниц самого нижнего уровня индекса (листьевых). Такой способ, конечно, в меньшей степени снижает фрагментацию, но его можно применять без отключения пользователей, и он требует меньшего количества ресурсов.

Для выполнения операций по перестроению или реорганизации индексов можно также использовать графические средства SQL Server Management Studio. Команды **Rebuild** (Перестроить) и **Reorganize** (Реорганизовать) предусмотрены в контекстном меню для самого индекса и для всего контейнера **Indexes** (Индексы). Можно также воспользоваться средствами планов обслуживания баз данных (*database maintenance plans*).

Важная проблема, которую необходимо учитывать, связана с тем, что при перестроении индексов статистика не обновляется автоматически. Поэтому выполнение практически любого запроса к базе данных будет приводить к автоматическому пересчету статистики. Это, конечно, не ускорит их выполнение. Поэтому, если вы производите перестроение индексов на регулярной основе, очень рекомендуется после завершения этой операции сразу произвести обновление статистики во всей базе данных. Если вы используете для этой цели скрипт SQL-DMO, то можно добавить в него единственную строку с вызовом метода **UpdateStatistics()**:

```
Dim oServer, oDB, oTable
Set oServer = CreateObject("SQLDmo.SqlServer2")
oServer.LoginSecure = True
oServer.Connect "LONDON2\SQL2005"
Set oDB = oServer.Databases("AdventureWorks")
For Each oTable In oDB.Tables
 If oTable.SystemObject = False Then
 oTable.RebuildIndexes
 oTable.UpdateStatistics
 End if
Next
```

Подробно про статистику будет рассказываться в разд. 11.5.8.

## Задание для самостоятельной работы 11.4. Дефрагментация таблиц и индексов

### Задание:

- Произведите анализ степени фрагментации индекса `PK_Address_AddressID` для таблицы `Person.Address` в базе данных `AdventureWorks` на вашем сервере SQL Server 2005.
- Произведите перестроение всех индексов и обновление статистики в базе данных `Foodmart` средствами SQL-DMO. Убедитесь, что обновление статистики произведено.

### Решение:

К пункту 1 задания — анализ степени фрагментации:

- Соответствующая команда может выглядеть так:

```
USE AdventureWorks;
GO
DBCC SHOWCONTIG ('Person.Address', 'PK_Address_AddressID');
```

К пункту 2 задания — перестроение индексов в базе данных:

- Соответствующий код на языке VBScript может выглядеть так:

```
Dim oServer, oDB, oTable
Set oServer = CreateObject("SQLDmo.SqlServer2")
oServer.LoginSecure = True
oServer.Connect
Set oDB = oServer.Databases("AdventureWorks")
For Each oTable In oDB.Tables
 If oTable.SystemObject = False Then
 oTable.RebuildIndexes
 oTable.UpdateStatistics
 End if
Next
```

- Для того чтобы убедиться, что обновление статистики произошло, можно открыть свойства статистики (например, для индекса `PK_Address_AddressID`). Для этого необходимо открыть контейнер `Statistics` (**имя\_сервера** | **Databases** | **имя\_базы\_данных** | **Tables** | **имя\_таблицы** | **Statistics**) в SQL Server Management Studio, затем открыть свойства нужной статистики и на вкладке **General** просмотреть информацию о дате создания статистики.

## 11.5.7. Работа с блокировками

Очень важные возможности оптимизации SQL Server связаны с подсистемой работы с блокировками. Реализация этой подсистемы в SQL Server вызывает справедливые нарекания у специалистов, и для многих практических задач ручная настройка блокировок является необходимой.

В SQL Server предусмотрено пять главных уровней блокировок:

- блокировка уровня базы данных (`DB`). Такие блокировки автоматически накладываются на любую базу данных, к которой подключен пользователь. В основном они предназначены для запрета выполнения с базами данных, к которым подключены пользователи, определенных действий, например, удаления данных;
- блокировки уровня объекта (например, `tab`). Такие блокировки могут накладываться на таблицу или индекс как при выполнении обычных запросов, так и при выполнении служебных операций с этими объектами;
- блокировки уровня экстента (`ext`). Такие блокировки нечасто можно увидеть в SQL Server Management Studio или в результатах выполнения хранимой процедуры `sp_lock`. Они используются только при таких служебных операциях, как создание новых таблиц, увеличении размера файлов баз данных и т. п.;
- блокировки уровня страницы (`PAG`). Такие блокировки используются SQL Server очень часто. При их применении блокируется вся страница размером 8 Кбайт, со всеми записями, которые в ней находятся. Такой тип блокировки может применяться как для страниц данных, так и для страниц индексов;
- блокировки уровня записи/ключа (`RID/KEY`). Такие блокировки накладываются на отдельные записи. Блокировки типа `RID` накладываются на записи в таблицы без кластерного индекса (*heap*), а блокировки типа `KEY` — на записи в таблицах, для которых предусмотрен кластерный индекс.

Блокировки также различаются по типам (общие, исключительные, блокировки ожидания и т. п.), но для целей оптимизации важен только их уровень.

По умолчанию SQL Server управляет уровнем блокировок автоматически. Для большинства операций SQL Server вначале пытается использовать только блокировки на уровне записи. Если запрос на чтение или изменение данных касается большого количества записей, то для экономии ресурсов SQL Server может принять решение об использовании блокировок более высокого уровня. Такое повышение уровня называется *эскалацией блокировок* (*lock escalation*).

Отметим некоторые технические моменты, связанные с эскалацией блокировок.

В большинстве ситуаций SQL Server изначально пытается использовать блокировки уровня записи или страницы. Какой именно уровень блокировок использовать — записи или страницы, определяется перед началом выполнения запроса. Решение об эскалации блокировок SQL Server принимает в двух ситуациях:

- когда запрос пытается применить к одному объекту (к таблице или индексу) более 5000 блокировок на уровне записи или страницы. Значение 5000 блокировок взято из документации, но на практике SQL Server иногда продолжает использовать блокировки уровня записи или страниц и при количестве в сотни тысяч таких блокировок. При этом SQL Server никогда не производит эскалацию с уровня записи до уровня страницы, а сразу пытается наложить блокировку на таблицу;
- когда место в области оперативной памяти, отведенной для работы с блокировками, заканчивается. На каждую блокировку SQL Server отводит 96 байт. Размер области памяти для работы с блокировками SQL Server по умолчанию (при установленном параметре сервера `Lock`, равном 0) настраивается динамически. Как только размер этой области достигает 40% от общего размера памяти, которую использует процесс SQL Server (она ограничивается операционной системой или параметром `MAX SERVER MEMORY`), SQL Server автоматически пытается произвести эскалацию блокировок. Если размер области оперативной памяти достиг 60% от объема памяти, максимально доступной SQL Server, создание новых блокировок не производится, а клиенту возвращается ошибка 1204 "unable to allocate lock resource" (невозможно разместить ресурсы блокировки).

Если SQL Server не удалось повысить уровень блокировки (например, на уровне таблицы уже есть блокировка другой транзакции), то повторные попытки будут производиться через каждые 1250 новых блокировок на уровне записи или страницы.

Мониторинг событий эскалации блокировок можно производить при помощи профилировщика, выбрав в нем событие `Lock:Escalation`.

Внешне механизм эскалации выглядит очень логичным, однако на практике с ним возникают проблемы:

- при выполнении операций, которые должны быть выполнены с большим количеством записей в таблице, SQL Server пытается вначале использовать блокировки уровня записи. В результате на установку и последующее снятие таких блокировок расходуется значительное количество системных ресурсов. Этого можно было бы избежать, если сразу применить для выполнения операции нужный уровень блокировок (`PAG` или `TAB`);

- вторая, более важная проблема, заключается в том, что SQL Server применяет эскалацию блокировок, в том числе и на мощных серверах, с которыми одновременно работает большое число пользователей. Типичная ситуация выглядит таким образом: в базе данных есть большая таблица, с которой постоянно работают пользователи (назовем ее главной таблицей). За счет эскалации блокировок количество записей, которые одновременно блокируют пользователи, автоматически увеличивается, в результате чего другие пользователи не могут получить к ним доступ. Таким образом, при достижении определенного количества пользователей работа с этой таблицей резко затрудняется. Особенно неприятно то, что заблокированными оказываются те записи, с которыми пользователи на самом деле не работают: просто они попали на одну страницу с другими записями, открытыми в данный момент.

Для решения этих проблем в зависимости от ситуации можно использовать разные способы:

- первая возможность, которая позволяет вообще избавиться об блокировок при запросах на чтение данных, — использовать соответствующий уровень изоляции транзакций. При операциях записи блокировки не налагаются при использовании уровней:
- READ COMMITTED, когда для параметра базы данных `READ_COMMITTED_SNAPSHOT` установлено значение `ON`;
  - READ UNCOMMITTED;
  - SNAPSHOT (новый уровень изоляции, который появился в SQL Server 2005).

Такие уровни изоляции вполне можно использовать для запросов со статистической информацией, когда абсолютной точностью информации можно пожертвовать ради производительности. Для финансовых отчетов такой способ может не подойти;

- еще одно радикальное решение — перевести базу данных в состояние `READ-ONLY`. В этом случае при обращении к ней никакие блокировки использоваться не будут, что может дать очень большой выигрыш в производительности и повысить количество пользователей, которые могут работать с таблицей одновременно. Но по очевидным причинам такое решение подходит далеко не всегда;
- если вам нужно изначально повысить уровень блокировки, то можно использовать хинты `NOLOCK`, `ROWLOCK`, `PAGLOCK` и `TABLELOCK` в запросах. Минус такого подхода — не всегда есть возможность изменять код приложения;
- другая возможность, менее известная, но вполне доступная администраторам, — применение команды `ALTER INDEX` для отключения блокировки на

уровне страниц и записей (в SQL Server 2000 для этой цели можно было использовать хранимую процедуру `sp_indexoption`). Этую команду можно использовать для настройки уровня блокировок:

- для некластерного индекса;
- для кластерного индекса (эти же настройки будут одновременно применены для таблицы);
- для таблицы, у которой нет кластерных индексов (данные находятся в "куче" — *heap*). Эта возможность является недокументированной, но ее вполне можно использовать.

Используя значение `OFF` для параметра `ALLOW_PAGE_LOCKS` в этой команде, вы можете избавиться от блокировок уровня страницы — главной причины проблем одновременного доступа пользователей.

В SQL Server 2005 появилась возможность настройки уровня блокировки на графическом экране. Для этого достаточно открыть свойства индекса в **Object Explorer**, перейти на вкладку **Options** и воспользоваться флагами **Use row locks then accessing the index** (Использовать блокировки записей при обращении к индексу) и **Use page locks then accessing the index** (Использовать блокировки страниц при обращении к индексу);

- если проблема, по вашему мнению, может происходить из-за того, что для области блокировки в памяти выделено недостаточно места, то вы можете попробовать настроить размер этой области вручную. Для этого можно использовать параметр конфигурации сервера `Lock`, а также попробовать увеличить количество оперативной памяти, доступной SQL Server, при помощи увеличения физической оперативной памяти на сервере или при помощи параметра `MAX SERVER MEMORY`;
- еще одна радикальная возможность — использовать флаги трассировки. Флаг 1211 просто отключает любую эскалацию блокировок. С этим параметром нужно быть очень осторожным, т. к. вы можете резко снизить производительность работы сервера и увеличить расход оперативной памяти. Кроме того, при исчерпании оперативной памяти, отводимой на блокировки, вы рискуете получить ошибку 1204. Второй флаг — 1224. Он отключает эскалацию блокировок, производимую по счетчику (5000 блокировок на нижнем уровне), однако при исчерпании места в области оперативной памяти, отводимой под блокировки, эскалация все равно будет происходить.

Если одновременно установить флаги 1224 и 1211, то приоритет будет отдан флагу 1224.

## Задание для самостоятельной работы 11.5. Управление уровнем блокировок

### Задание:

1. Откройте новое окно редактора скриптов в SQL Server Management Studio и перейдите к базе данных AdventureWorks.
2. Выполните команды:

```
BEGIN TRAN
UPDATE person.address SET City = 'Test';
```

Засеките время выполнения этих команд (показывается в нижнем правом углу SQL Server Management Studio). Затем выполните команду:

```
sp_lock @@SPID;
```

и просмотрите блокировки, которые наложены вашим процессом. SQL Server должен использовать только блокировки уровня объекта (ТАБ).

Выполните команду:

```
ROLLBACK TRAN
```

3. Отключите эскалацию блокировок и еще раз выполните команду:

```
BEGIN TRAN
UPDATE person.address SET City = 'Test';
```

Засеките время выполнения этой команды, а затем выполните команду:

```
sp_lock @@SPID;
```

Время выполнения запроса должно увеличиться, а хранимая процедура sp\_lock должна показать, что используются блокировки уровня страницы и ключа.

Выполните команду:

```
ROLLBACK TRAN
```

4. Отключите возможность применения блокировок уровня страницы для таблицы Person.Address и еще раз выполните команды:

```
BEGIN TRAN
UPDATE person.address SET City = 'Test';
```

Засеките время выполнения этой команды, а затем выполните команду:

```
sp_lock @@SPID;
```

Хранимая процедура `sp_lock` должна вернуть только информацию о блокировках уровня записи. Выполните команду:

```
ROLLBACK TRAN
```

5. Отключите возможность применения блокировок уровня страницы для таблицы `Person.Address` и еще раз выполните команды:

```
BEGIN TRAN;
```

```
UPDATE person.address SET City = 'Test';
```

Засеките время выполнения этой команды, а затем выполните команду:

```
sp_lock @@SPID;
```

Хранимая процедура `sp_lock` должна вернуть только информацию о блокировках уровня таблицы. Такой уровень блокировок теперь выбран изначально, без эскалации блокировок. Выполните команду:

```
ROLLBACK TRAN
```

### Решение:

К пункту 3 задания — отключение эскалации блокировок:

1. Для этого можно выполнить команду:

```
DBCC TRACEON (1211);
```

К пункту 4 задания — отключение возможности применения блокировок на уровне страницы:

1. Это выполняется с помощью команды:

```
ALTER INDEX ALL ON [Person].[Address] SET (ALLOW_PAGE_LOCKS = OFF);
```

К пункту 5 задания — отключение возможности применения блокировок на уровне записи:

1. Для этого можно выполнить команду:

```
ALTER INDEX ALL ON [Person].[Address] SET (ALLOW_ROW_LOCKS = OFF);
```

## 11.5.8. Оптимизация запросов

Если все остальные способы оптимизации производительности уже исчерпаны, то в распоряжении разработчиков и администраторов SQL Server остается последний резерв — оптимизация выполнения отдельных запросов. Например, если в вашей задаче совершенно необходимо ускорить создание какого-то одного специфического отчета, можно проанализировать запрос,

который используется при создании этого отчета, и постараться изменить его план, если он неоптimalен.

Отношение к оптимизации запросов у многих специалистов неоднозначное. С одной стороны, работа программного модуля Query Optimizer, который генерирует планы выполнения запросов, вызывает множество справедливых нареканий и в SQL Server 2000, и в SQL Server 2005. Query Optimizer часто выбирает не самые оптимальные планы выполнения запросов и в некоторых ситуациях проигрывает аналогичным модулям из Oracle и Informix. С другой стороны, ручная оптимизация запросов — процесс чрезвычайно трудоемкий. Вы можете потратить много времени на такую оптимизацию и, в конце концов, выяснить, что ничего оптимизировать не удалось: план, предложенный Query Optimizer изначально, оказался наиболее оптимальным (так бывает в большинстве случаев). Кроме того, может случиться так, что созданный вами вручную план выполнения запросов через какое-то время (после добавления новой информации в базу данных) окажется неоптимальным и будет снижать производительность при выполнении запросов.

Отметим также, что для выбора наилучших планов построения запросов Query Optimizer необходима правильная информация о статистике. Поскольку, по опыту автора, далеко не все администраторы знают, что это такое, расскажем о статистике подробнее.

*Статистика* — это специальная служебная информация о распределении данных в столбцах таблиц. Представим, например, что выполняется запрос, который должен вернуть всех Ивановых, проживающих в городе Санкт-Петербурге. Предположим, что у 90% записей в этой таблице одно и то же значение в столбце Город — "Санкт-Петербург". Конечно, с точки зрения выполнения запроса вначале выгоднее выбрать в таблице всех Ивановых (их явно будет не 90%), а затем уже проверять значение столбца Город для каждой отобранный записи. Однако для того, чтобы узнать, как распределяются значения в столбце, нужно вначале выполнить запрос. Поэтому SQL Server самостоятельно инициирует выполнение таких запросов, а потом сохраняет информацию о распределении данных (которая и называется статистикой) в служебных таблицах базы данных.

Для баз данных SQL Server 2005 по умолчанию устанавливаются параметры AUTO\_CREATE\_STATISTICS и AUTO\_UPDATE\_STATISTICS. При этом статистика для столбцов баз данных будет создаваться и обновляться автоматически. Для самых больших и важных баз данных может получиться так, что операции по созданию и обновлению статистики могут мешать текущей работе пользователей. Поэтому для таких баз данных иногда эти параметры отключают, а операции по созданию и обновлению статистики выполняют вручную в ночные времена. Для этого используются команды CREATE STATISTICS и UPDATE STATISTICS.

Теперь поговорим об оптимизации запросов.

Первое, что необходимо сделать, — найти те запросы, которые в первую очередь подлежат оптимизации. Проще всего это сделать при помощи профилировщика, установив фильтр на время выполнения запроса (фильтр **Duration** в окне **Edit Filter** (Редактировать фильтр), которое можно открыть при помощи кнопки **Column Filters** на вкладке **Events Selection** окна свойств сеанса трасировки). Например, в число кандидатов на оптимизацию могут попасть запросы, время выполнения которых составило более 5 секунд. Кроме того, можно использовать информацию о запросах, которая предоставляется Database Tuning Advisor.

Затем нужно проверить, установлен ли для ваших соединений, хранимых процедур и функций параметр **NOCOUNT**. Установить его можно при помощи команды **SET NOCOUNT ON**. При установке этого параметра, во-первых, отключается возврат с сервера и вывод информации о количестве строк в результатах запроса (т. е. не отображается строка "N row(s) affected" на вкладке **Messages** (Сообщения) окна работы с кодом при выполнении запроса в Management Studio). Во-вторых, отключается передача специального серверного сообщения **DONE\_IN\_PROC**, которое по умолчанию возвращается для каждого этапа хранимой процедуры. При вызове большинства хранимых процедур нужен только результат их выполнения, а количество обработанных строк для каждого этапа никого не интересует. Поэтому установка параметра **NOCOUNT** для хранимых процедур может серьезно повысить их производительность. Повышается скорость выполнения и обычных запросов, но в меньшей степени (до 10%).

После этого можно приступать к работе с планами выполнения запросов.

План выполнения запроса проще всего просмотреть из SQL Server Management Studio. Для того чтобы получить информацию об ожидаемом плане выполнения запроса, можно в меню **Query** (Запрос) выбрать команду **Display Estimated Execution Plan** (Отобразить ожидаемый план выполнения). Если вы хотите узнать реальный план выполнения запроса, можно перед его выполнением установить в том же меню параметр **Include Actual Execution Plan** (Включить реальный план выполнения). В этом случае после выполнения запроса в окне результатов в SQL Server Management Studio появится еще одна вкладка **Execution Plan** (План выполнения), на которой будет представлен реальный план выполнения запроса. При наведении указателя мыши на любой из этапов можно получить о нем дополнительную информацию (рис. 11.15).

В плане выполнения запроса, как видно на рисунке, может быть множество элементов. Разобраться в них, а также предложить другой план выполнения — задача достаточно сложная. Надо сказать, что каждый из возможных

элементов оптимальен в своей ситуации. Поэтому обычно этапы оптимизации запроса выглядят так:

- вначале в окне Management Studio выполните команду SET STATISTICS IO ON. В результате после каждого выполнения запроса будет выводиться дополнительная информация. В ней нас интересует значение только одного параметра — Logical Reads. Этот параметр означает количество логических чтений при выполнении запросов, т. е. сколько операций чтения пришлось провести при выполнении данного запроса без учета влияния кэша (количество чтений и из кэша, и с диска). Это наиболее важный параметр. Количество физических чтений (чтений только с диска) — информация не очень представительная, поскольку зависит от того, были ли перед этим обращения к данным таблицам или нет. Статистика по времени также является величиной переменной и зависит от других операций, которые выполняет в это время сервер. А вот количество логических чтений — наиболее объективный показатель, на который в наименьшей степени влияют дополнительные факторы;
- затем пытайтесь изменить план выполнения запроса и узнать суммарное количество логических чтений для каждого плана. Обычно план выполне-

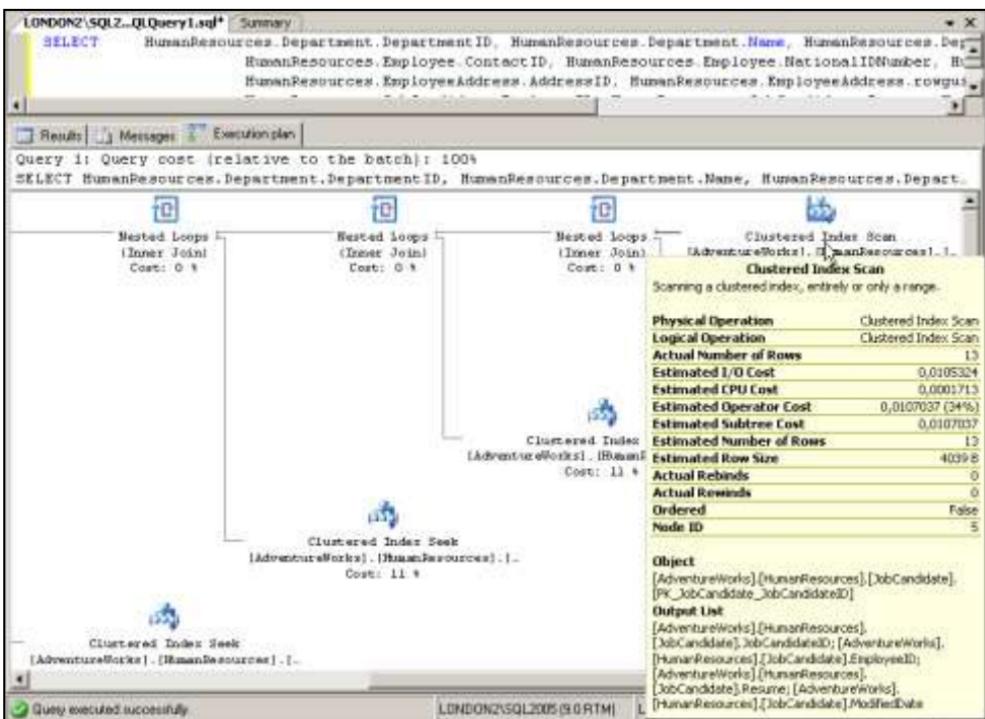


Рис. 11.15. План выполнения запроса в SQL Server Management Studio

ния запроса изменяется при помощи хинтов (подсказок) оптимизатора. Они явно указывают оптимизатору, какой план выполнения следует использовать.

Хинтов оптимизатора в SQL Server 2005 предусмотрено много. Прочитать информацию о них можно в Books Online (в списке на вкладке **Index** (Индекс) нужно выбрать **Query Hints [SQL Server]** (Хинты запросов [SQL Server]), **Join Hints** (Хинты джойнов) или **Table Hints [SQL Server]** (Табличные хинты [SQL Server])). Чаще всего используются следующие хинты:

- ❑ NOLOCK, ROWLOCK, PAGLOCK, TABLOCK, HOLDLOCK, READCOMMITTEDLOCK, UPDLOCK, XLOCK — эти хинты используются для управления блокировками (*см. разд. 11.5.7*);
- ❑ FAST *количество\_строк* — будет выбран такой план выполнения запроса, при котором максимально быстро будет выведено указанное количество строк (первых с начала набора записей). Если пользователю нужны именно первые записи (например, последние заказы), то для их максимально быстрой загрузки в окно приложения можно использовать этот хинт;
- ❑ FORCE ORDER — объединение таблиц при выполнении запроса будет выполнено точно в том порядке, в котором эти таблицы перечислены в запросе;
- ❑ MAXDOP (от Maximum Degree of Parallelism — максимальная степень распараллеливания запроса) — при помощи этого хинта указывается максимальное количество процессоров, которые можно будет использовать для выполнения запроса. Обычно этот хинт используется в двух ситуациях:
  - когда из-за переключения между процессорами (*context switching*) скорость выполнения запроса сильно снижается. Такое поведение было характерно для SQL Server 2000 на многопроцессорных системах;
  - когда вы хотите, чтобы какой-то тяжелый запрос оказал минимальное влияние на текущую работу пользователей;
- ❑ OPTIMIZE FOR — этот хинт позволяет указать, что запрос оптимизируется под конкретное значение передаваемого ему параметра (например, под значение фильтра для WHERE);
- ❑ USE PLAN — это самая мощная возможность. При помощи такого хинта можно явно определить план выполнения запроса, передав план в виде строкового значения в формате XML. Хинт USE PLAN появился только в SQL Server 2005 (в предыдущих версиях была возможность явно определять планы выполнения запросов, но для этого использовались другие средства). План в формате XML можно написать вручную, а можно сгенерировать автоматически (например, щелкнув правой кнопкой мыши по графическому экрану с планом выполнения, представленному на рис. 11.15,

и выбрав в контекстном меню команду **Save Execution Plan As** (Сохранить план выполнения как)).

В SQL Server 2005 появилась новая важная возможность, которая позволяет вручную менять план выполнения запроса без необходимости вмешиваться в текст запроса. Очень часто бывает так, что код запроса нельзя изменить: он жестко "просит" в коде откомпилированного приложения. Чтобы справиться с этой проблемой, в SQL Server 2005 появилась хранимая процедура `sp_create_plan_guide`. Она позволяет создавать так называемые *руководства по планам выполнения* (*plan guides*), которые будут автоматически применяться к соответствующим запросам.

Если вы анализируете запросы, которые направляет к базе данных какое-то приложение, то имеет смысл в первую очередь обратить внимание на следующие моменты:

- насколько часто в планах выполнения запроса встречается операция **Table Scan** (Полное сканирование таблицы). Вполне может оказаться, что обращение к таблице при помощи индексов будет эффективнее;
- используются ли в коде курсоры. Курсоры — очень простое средство с точки зрения синтаксиса программы, но чрезвычайно неэффективное с точки зрения производительности. Очень часто можно избежать применения курсоров, используя другие синтаксические конструкции, и получить большой выигрыш в скорости работы;
- используются ли в коде временные таблицы или тип данных `Table`. Создание временных таблиц и работа с ними требуют большого расхода ресурсов, поэтому по возможности нужно их избегать;
- кроме создания временных таблиц, значительного расхода системных ресурсов требует и изменение их структуры. Поэтому команды на изменение структуры временных таблиц должны сразу привлекать ваше внимание. Обычно есть возможность сразу создать временную таблицу со всеми необходимыми столбцами;
- иногда запросы возвращают больше данных, чем реально требуется приложению (лишнее количество столбцов или строк). Конечно, это не способствует повышению производительности;
- если приложение передает на сервер команды `EXECUTE`, то имеет смысл подумать о том, чтобы заменить их на вызов хранимой процедуры `sp_executesql`. Она обладает преимуществами в производительности по сравнению с обычной командой `EXECUTE`;
- повышения производительности иногда можно добиться, устранив необходимость повторной компиляции хранимых процедур и построения но-

вых планов выполнения запросов. Нужно обратить внимание на применение параметров, постараться не смешивать в коде хранимой процедуры команды DML и DDL и следить за тем, чтобы параметры подключения SET ANSI\_DEFAULTS, SET ANSI\_NULLS, SET ANSI\_PADDING, SET ANSI\_WARNINGS и SET CONCAT\_NULL\_YIELDS\_NULL не изменялись между запросами (любое изменение таких параметров приводит к тому, что старые планы выполнения считаются недействительными). Обычно проблема может возникнуть тогда, когда эти параметры устанавливаются на уровне отдельного запроса или в коде хранимой процедуры.

Отметим, что в любом случае создание планов выполнения запросов вручную и использование хинтов — это крайнее средство, которого следует по возможности избегать.



## ГЛАВА 12

# Репликация в SQL Server 2005

### 12.1. Зачем нужна репликация

Репликация формально определяется как процесс синхронизации информации на разных источниках данных. Чаще всего репликация в SQL Server используется в следующих ситуациях:

- чтобы распространить информацию с одного сервера на несколько серверов. Например, средствами репликации можно из центрального офиса передавать в филиалы новые прайс-листы, курсы валют и т. п.;
- для сбора информации с нескольких серверов на один сервер. Например, при помощи репликации можно собирать и сводить воедино отчеты об операциях, которые были произведены в филиалах;
- для снятия нагрузки с рабочего сервера. Предположим, что какой-то сервер SQL Server 2005 используется для обычной OLTP-задачи, например, для работы с заказами, поступающими на предприятие. Эта же информация требуется для создания отчетов. При этом нагрузка, которая возникает при создании отчетов, может мешать пользователям работать с заказами. В такой ситуации для создания отчетов можно использовать второй сервер, а данные для него передавать при помощи репликации;
- для повышения отказоустойчивости. Например, вы можете установить запасной сервер и через определенные интервалы времени реплицировать на него данные с рабочего сервера. В случае выхода из строя рабочего сервера его можно будет быстро заменить запасным.

Отметим некоторые принципиальные моменты, которые связаны с репликацией.

Репликацию удобнее всего использовать, когда данные между серверами синхронизируются с интервалом от 15—30 минут до нескольких дней. Если

данные нужно синхронизировать быстрее, имеет смысл подумать о применении распределенных транзакций (или, возможно, о использовании зеркального отображения баз данных — см. разд. 7.3). Если данные нужно синхронизировать реже, чем раз в несколько дней, то, возможно, проще будет использовать резервные копии или пакеты SSIS.

Репликация обычно используется тогда, когда нужно передавать не все изменения, которые происходят на базе данных-источнике. Если вам нужно обеспечивать полностью идентичные копии баз данных, то имеет смысл подумать о применении автоматической доставки журналов (см. разд. 7.2) или зеркального отображения баз данных (см. разд. 7.3).

Репликация SQL Server достаточно сложна для настройки, администрирования и диагностики проблем. Кроме того, она требует значительного расхода системных ресурсов серверов и передачи по сети большого объема информации. Поэтому некоторые разработчики используют альтернативные средства репликации более простые, надежные и эффективные. Автору известно несколько примеров реализации таких альтернативных систем репликации.

Например, разработчики, обслуживающие базу данных Комитета финансов мэрии Санкт-Петербурга, столкнулись с проблемами при обмене данных. У Комитета финансов есть множество подразделений в разных районах города, которые должны осуществлять платежи. При этом дать каждому подразделению канал связи, достаточный для использования стандартных средств репликации SQL Server, не представлялось возможным. Фактически большинство подразделений могло использовать для подключения к центральному офису только модемные коммутируемые соединения с самым разным качеством линий. В то же время в конце каждого дня информация о всех проведенных подразделением платежах должна была передаваться в главную базу данных.

Разработчики приняли решение отказаться от стандартных средств репликации SQL Server и создали свою систему репликации. Принцип найденного решения оказался очень простым. Приложение было реализовано таким образом, чтобы все изменения в базу данных могли вноситься только при помощи хранимых процедур. При этом каждая из хранимых процедур при запуске заносила в специальную таблицу информацию для протоколирования: свое имя, время запуска, информацию о пользователе, который ее запустил, а также значения параметров, которые были ей переданы. В конце дня эта таблица экспортировалась в текстовый файл, который и передавался по модему в центральный офис. Затем в центральном офисе этот файл использовался для того, чтобы еще раз "прогнать" все эти процедуры с запротоколированными параметрами, но уже на главной базе данных. Такая система оказалась менее требовательной к ресурсам и пропускной способности сети, более надежной и простой, чем стандартная система репликации.

Автору совершенно не хотелось бы создавать впечатление, что система репликации SQL Server 2005 никуда не годится. Наоборот, во многих ситуациях без нее обойтись очень сложно. Однако не следует забывать и про альтернативные возможности, которые во многих случаях могут оказаться лучше.

## 12.2. Новые возможности репликации SQL Server 2005

Этот раздел предназначен для специалистов, которые хорошо знакомы с системой репликации в SQL Server 2000. В нем приводится обзорная информация о наиболее значительных изменениях системы репликации в SQL Server 2005. Более подробно они будут рассмотрены в следующих разделах.

Если же вы не занимались настройкой репликации в предыдущих версиях SQL Server, то этот раздел можно пропустить и сразу перейти к следующему разделу.

Вот перечень самых важных новых возможностей системы репликации:

- в SQL Server 2005 поддерживается репликация практически любых изменений в структуре опубликованных таблиц (в SQL Server 2000 поддерживалась репликация только добавления и удаления столбцов, если она проводилась специальными хранимыми процедурами). Во многих ситуациях это очень удобно;
- в SQL Server 2005 реализована "умная" доставка моментальных снимков баз данных (*database snapshots*). Если моментальный снимок состоит из нескольких файлов и при его передаче произошел сбой, что при повторе передачи будут переданы только те файлы, которые не были доставлены серверу-получателю. Файлы моментального снимка, которые уже были доставлены, повторно не передаются;
- в SQL Server 2005 появилась модель одноранговой репликации, в которой нет разницы между подписчиком и издателем. При одноранговой репликации изменения можно производить на любом сервере — информация между всеми серверами, которые участвуют в такой репликации, автоматически синхронизируется;
- в репликации слиянием (*merge replication*) можно использовать логические записи — когда запись из двух связанных между собой физических таблиц для целей репликации рассматривается как одна запись. Это позволяет повысить как надежность, так и производительность репликации;
- SQL Server 2005 может напрямую выступать в качестве подписчика для баз данных Oracle. Кроме того, он сам может выступать в качестве издателя для других серверов баз данных, таких как IBM DB2 или Oracle;

- в SQL Server 2005 столбцы идентификатора (*identity columns*) можно реплицировать именно как столбцы идентификатора. В SQL Server 2000 такие столбцы можно было реплицировать только как столбцы с базовым для идентификатора типом данных, например, `int`;
- моментальные снимки баз данных для целей инициализации базы данных-подписчика можно передавать вручную при помощи резервной копии. Если размер базы данных-издателя составляет десятки гигабайт, то такая возможность может оказаться очень полезной;
- в SQL Server 2005 реализована возможность диагностики репликации при помощи отправки специальных небольших диагностических объемов данных (трассировочных маркеров — *tracer tokens*). В результате проверка работоспособности системы репликации и диагностика проблем значительно упрощается;
- в SQL Server 2005 появилась новая объектная модель для программного управления системой репликации, которая называется RMO (Replication Management Objects).

## 12.3. Терминология системы репликации

В этом разделе будут рассмотрены некоторые термины, которые используются для системы репликации в SQL Server 2005.

В репликации SQL Server предусмотрено три главные роли для участвующих в ней серверов: распространитель, издатель и подписчик.

Главный участник системы репликации — это, конечно, **распространитель** (*distributor*). Его роль можно сравнить с насосом, который перекачивает данные из одного места в другое. На нем в специальной базе данных (по умолчанию она называется *Distribution*) хранятся основные настройки репликации. Отметим, что при настройке репликации в SQL Server 2005 и в качестве издателя, и в качестве подписчика могут выступать "посторонние" базы данных (например, базы данных Oracle). Но распространителем может быть только SQL Server! Как правило, настройка системы репликации начинается именно с настройки распространителя.

**Издатель** (*publisher*) — это источник данных для репликации. Издателем в системе репликации SQL Server 2005 может выступать либо экземпляр SQL Server 2005, SQL Server 2000 или 7.0, а также сервер Oracle. В ходе репликации данные с издателя передаются получателям — подписчикам.

**Подписчик** (*subscriber*) — это последний сервер, который принимает участие в репликации. Подписчик — это тот, кто принимает данные с издателя. Подписчиком могут быть как серверы SQL Server, так и другие серверы баз данных (например, Oracle или IBM DB2).

Свой набор терминов предусмотрен и для описания единиц репликации. К ним относятся статьи, публикации и подписки.

**Статья (article)** — это данные из какой-либо таблицы или представления на издателе. Данные из таблицы или представления для статьи можно отфильтровать, выбрав нужный набор столбцов и записей. Подписаться на статью нельзя. Но статьи можно поместить в публикацию и настроить на нее подписку.

**Публикация (publications)** — это основная единица репликации. Подписчики подписываются именно на публикацию. Публикация состоит из одной или нескольких статей, которые для целей публикации рассматриваются как единое целое. Подписаться на часть публикации нельзя.

Если, например, ваша публикация состоит из трех статей, а какому-то серверу-подписчику нужна информация только из одной статьи, то правильным решением в этом случае будет создать еще одну публикацию из одной статьи и подписать этот сервер именно на нее.

**Подписка (subscription)** — это запрос на получение публикации. Подписка определяет, какую публикацию запрашивает подписчик и с какой частотой будут передаваться данные. Подписки могут быть двух видов: *принудительная (push)*, когда передачу информации инициирует издатель, и *запрашивающая (pull)*, когда в качестве инициатора выступает подписчик.

Репликация производится при помощи специальных программных модулей, которые называются *агентами репликации*. Для запуска агентов репликации по расписанию используются задания SQL Server Agent. Главными агентами считаются следующие пять:

□ **Snapshot Agent** — задачей этого программного модуля является создание моментальных снимков (*snapshots*). *Моментальный снимок* — это полная копия информации (и структура, и данные), которая содержится в публикации, на определенный момент времени. Snapshot Agent используется во всех типах репликации (про типы репликации будет рассказываться в разд. 12.4). В репликации моментальных снимков передаются только моментальные снимки через определенные интервалы времени. В репликации транзакций и репликации слиянием этот агент готовит моментальные снимки для исходной инициализации таблиц на подписчике;

□ **Log Reader Agent** — этот агент осуществляет мониторинг журнала транзакций на издателе и заносит информацию о всех транзакциях, которые затрагивают информацию в публикации, в базу данных *Distribution*. Затем эта информация передается подписчику, где используется для изменения его таблиц. Программный модуль Log Reader Agent используется только в репликации транзакций;

- **Distribution Agent** — по своим функциям этот программный модуль больше всего похож на насос по перекачке данных. Он осуществляет физическое перемещение моментальных снимков и информации о транзакциях на подписчиков и применяет к ним эту информацию. Для принудительных (*push*) подписок этот программный модуль работает на распространителях, а для запрашивающих (*pull*) подписок — на подписчиках. Он используется в репликации моментальных снимков и репликации транзакций;
- **Merge Agent** — этот программный модуль при репликации слиянием выполняет функции Distribution Agent. Кроме того, он ответственен за разрешение конфликтов репликации, которые могут возникнуть при использовании репликации слиянием;
- **Queue Reader Agent** — этот агент используется только при репликации транзакций с использованием очередей. Его задача — считывать из очереди сообщения, получаемые с подписчика (например, о том, что такие-то транзакции успешно завершены), и передавать их издателю для выполнения различных операций.

Кроме перечисленного набора в репликации SQL Server 2005 используется еще несколько агентов, которые предназначены в основном для удаления старой информации.

Для каждого агента предусмотрена история работы, которую можно использовать для диагностики проблем.

## 12.4. Типы репликации

Перед настройкой репликации очень важно правильно выбрать ее тип. В SQL Server 2005 предусмотрено три главных типа: репликация моментальных снимков, репликация транзакций и репликация слиянием. У этих главных типов есть подтипы.

**Репликация моментальных снимков** (*snapshot replication*) — это тип репликации, когда информация на подписчике через определенные интервалы времени просто перезаписывается информацией с издателя. Никакого отслеживания происходящих изменений при этом не происходит. Такой тип репликации является самым простым и надежным, но не всегда эффективным. Конечно, его нельзя использовать, если в подписку попадает значительный объем данных: нагрузка на сеть будет очень большой. Этот тип репликации обычно используется в случае, когда нужно передавать небольшой объем данных, и эти данные изменяются достаточно сильно. Например, репликацию моментальных снимков можно использовать для распространения в филиалах прайс-листов или курсов валют.

**Репликация транзакций** (*transactional replication*) — это самый распространенный тип репликации. Под репликацией, в первую очередь, понимают схему работы именно этого типа репликации. При использовании репликации транзакций вначале к подписчику применяется моментальный снимок исходных данных с издателя (подписчик "инициализируется"), а потом через определенные интервалы подписчику передается и применяется информация о произошедших на издателе изменениях — транзакциях. Этот тип репликации сложнее, чем репликация моментальных снимков, но он больше подходит в ситуациях, когда набор данных в подписке достаточно большой. Как уже говорилось, этот тип репликации является наиболее распространенным и поэтому применяется во многих случаях. Вот несколько наиболее типичных примеров использования репликации транзакций:

- для получения отчетов о произведенных операциях с филиалов. При этом у вас будет множество издателей (серверы в филиалах) и один подписчик (сервер в центральном офисе);
- для создания запасного сервера для повышения отказоустойчивости или снятия нагрузки с основного сервера;
- для настройки одноранговой системы репликации, когда информация в одном наборе таблиц может изменяться на любом сервере из нескольких, а затем синхронизироваться;
- в качестве замены пакетов SSIS при сборе данных с разных серверов в базе данных на одном сервере (т. е. для организации хранилища данных).

Для репликации транзакций предусмотрены специальные подтипы, которые позволяют настроить режимы репликации, отличные от стандартного.

Первый подтип — *репликация транзакций с возможностью внесения изменений на подписчиках* (*transactional replication with updatable subscriptions*). По умолчанию в репликации транзакций разрешается внесение изменений только на издателе, но иногда возникают ситуации, когда нужно разрешить внесение изменений и на подписчиках. Заметим, что если на подписчиках изменения нужно вносить постоянно, то, возможно, в этой ситуации лучше подойдет репликация слиянием.

При настройке репликации транзакций с возможностью внесения изменений на подписчиках вам нужно будет выбрать один из двух режимов предупреждения издателя о таких изменениях:

- немедленное извещение об изменениях (*immediate updating*). При выборе этого режима должно существовать постоянно доступное сетевое соединение между издателем и подписчиком. Как только на подписчике вносятся изменения, он немедленно уведомляет о них издателя. Тот, в свою очередь, получив информацию об этих изменениях, уведомляет о них других подписчиков;

- извещение об изменениях с использованием очередей (*queued updating*). Постоянно доступное сетевое соединение при этом не нужно. Информация о внесенных на подписчиках изменениях помещается в очередь и по определенному расписанию передается издателю, который уведомляет о них остальных подписчиков.

Еще один подтип репликации транзакций (он появился только в SQL Server 2005) — это *одноранговая репликация* (*peer replication*), в которой изменения можно вносить на любой из серверов, участвующих в репликации. Вся информация будет автоматически синхронизирована между ними.

Третий главный тип репликации — это *репликация слиянием* (*merge replication*). Это самый сложный тип репликации, и используется он очень редко. При репликации слиянием изменения можно вносить как на издателе, так и на подписчиках. Все изменения сводятся воедино на издателе, который разрешает конфликты в случае их возникновения. Для разрешения конфликтов репликации можно определить свою собственную программную логику. После этого итоговый вариант данных передается подписчикам.

В качестве классического примера такой репликации приводится несколько равноправных магазинов, которые имеют право работать с общим складом. Изменять информацию (забирать товар со склада) имеет право каждый магазин, и каждый магазин должен получить после этого итоговую информацию о состоянии склада. Но, конечно, это скорее теоретический пример. В большинстве реальных ситуаций во избежание конфликтов эта задача будет решаться по-другому.

Еще одна ситуация, для которой предназначена репликация слиянием, — репликация центральной базы данных SQL Server 2005 с базами данных SQL Server Mobile на наладонных компьютерах (например, торговых представителей). Фактически это вариант предыдущего примера, но более приближенный к реальной жизни.

Обратите внимание, что репликацию слиянием имеет смысл использовать лишь тогда, когда итоговая информация нужна не только в центральной базе данных, но и во всех других подразделениях. Если вам нужно просто собрать информацию с филиалов, то правильнее будет использовать репликацию транзакций с одним подписчиком и множеством издателей.

## 12.5. Подготовка к настройке репликации

Прежде чем приступать к настройке репликации, необходимо выполнить определенные подготовительные действия и принять решения.

Первое, что нужно сделать, — определиться с типом репликации и теми данными, которые будут реплицироваться. Возможно, подписчику будут нужны

не все данные издателя. В этом случае можно предварительно отфильтровать нужные данные, создав соответствующий запрос или хранимую процедуру.

Кроме того, необходимо принять решение относительно ролей серверов, которые будут участвовать в репликации. Часто несколько ролей (например, издателя и распространителя или распространителя и подписчика) можно назначить одному серверу. При назначении ролей серверам во внимание нужно принимать:

- загрузку сервера. Очень часто издатель — это самый загруженный сервер, поэтому лучше сделать распространителем другой компьютер;
- топологию сети. Если сеть у вас построена по топологии "звезда", то лучше продумать систему репликации таким образом, чтобы распространитель, который передает информацию множеству подписчиков, находился в центре этой звезды. При этом издатель вполне может находиться в одном из лучей;
- удобство администрирования. Основные операции по настройке, управлению, мониторингу репликации производятся на распространителе, поэтому при выборе распространителя необходимо принимать во внимание и этот момент.

Если в ваших базах данных используется только формат UNICODE для символьных данных, то никаких проблем не возникнет. Не возникнут они и в том случае, когда используются данные не в формате UNICODE, но для баз данных сервера-издателя и сервера-подписчика используются одинаковые кодировки. Если же кодировки разные, то могут возникнуть проблемы. Рекомендуется произвести проверки перед настройкой репликации, и, возможно, при настройке репликации продумать возможность проведения дополнительных преобразований.

В SQL Server можно сделать так, что при репликации будут игнорироваться некоторые объекты базы данных издателя, если использовать для этих объектов параметр NOT FOR REPLICATION. Этот параметр можно применить:

- для внешних ключей (*foreign key*). При вставке или изменении данных средствами репликации в таблицах подписчика это ограничение целостности использовать не будет;
- для условий на значение (объектов *check*);
- для триггеров. Такие триггеры не будут срабатывать при вставке, изменении или удалении данных средствами репликации в таблицах подписчика;
- для столбцов счетчика (*identity columns*). При вставке новых записей средствами репликации такие столбцы будут восприниматься как обычные (например, с целочисленным типом данных). Автоматическое приращение значений для таких столбцов производиться не будет.

Параметр NOT FOR REPLICATION можно определить как при настройке репликации, так и при обычном создании объектов с помощью скриптов Transact-SQL.

Если вы планируете использовать репликацию слиянием, то вам придется внести в структуру опубликованных данных определенные изменения. Для того чтобы уникально идентифицировать каждую запись (например, для разрешения конфликтов), при этом типе репликации в каждой таблице или представлении должен быть столбец типа uniqueidentifier. Его можно определить перед созданием репликации. Если при настройке репликации соответствующий столбец в таблице не обнаружен, вам будет предложено добавить его в автоматическом режиме.

Проблемы могут возникнуть и со столбцами BLOB — типами данных text, ntext и image. Репликация слиянием требует, чтобы после команды WRITETEXT или UPDATETEXT в той же транзакции обязательно присутствовала команда UPDATE (иначе не сработает триггер, и информация об изменениях не будет передана подписчикам). Репликация транзакций при выполнении операций с такими значениями обязательно требует помещения команд WRITETEXT или UPDATETEXT в явно объявленные транзакции. Кроме того, для команд WRITETEXT и UPDATETEXT должен обязательно использоваться параметр WITH LOG, иначе информация о таких операциях не будет занесена в журнал транзакции и, соответственно, не будет реплицирована.

Большую осторожность нужно проявлять и при выполнении операций массовой вставки данных в опубликованные таблицы. Например, в репликации слиянием все изменения отслеживаются при помощи триггеров. В то же время по умолчанию при выполнении операции массовой вставки данных триггеры не срабатывают. Поэтому в этом случае рекомендуется использовать параметр FIRE\_TRIGGERS для утилиты bcp или для команды BULK INSERT.

Имеет смысл также подумать о требованиях к дисковому пространству. Если опубликованные таблицы достаточно большие или в них вносится большое количество изменений, подумайте, достаточно ли у вас места на диске для хранения файлов моментальных снимков и информации об изменениях.

## 12.6. Настройка репликации

После того как все необходимые решения приняты, и все проверки проведены, можно настраивать репликацию.

Вариантов настройки репликации очень много. В этой главе будет рассматриваться только настройка наиболее распространенного типа репликации — репликации транзакций.

Предположим, что вам нужно настроить репликацию транзакций для всех таблиц схемы HumanResources в базе данных AdventureWorks на сервере LONDON2\SQL2005. Все изменения, которые вносятся в любую из этих таблиц, должны не позднее, чем через 10 минут отобразиться в одноименных таблицах новой базы данных AW2 на сервере LONDON2 YUKON2.

Первое, с чего обычно начинается настройка репликации, — настройка распространителя. Проще всего произвести настройку распространителя при помощи мастера Configure Distribution Wizard. Его можно запустить при помощи команды **Configure Distribution** (Настроить распределение) контекстного меню контейнера **Replication** (Репликация) в **Object Explorer** в SQL Server Management Studio.

Перед настройкой распространителя вам нужно принять решение относительно того, на каком сервере он будет работать. Возможны три варианта:

- распространитель работает на выделенном специально для него сервере;
- роль распространителя совмещена с ролью издателя;
- роль распространителя совмещена с ролью подписчика.

Ваш выбор зависит от степени загрузки серверов, принимающих участие в репликации и от топологии сети на предприятии. Для простоты в этом примере роль распространителя будет выполнять сервер-издатель (это очень распространенный на практике вариант). Поэтому запустите мастер Configure Distribution Wizard на сервере LONDON2\SQL2005.

На первом экране мастера **Distributor** (Дистрибутор) вы должны определиться, будет ли наш сервер выполнять роль распространителя для самого себя или он будет обслуживаться внешним сервером-распространителем. В вашем случае, конечно, выберите первый вариант.

На следующем экране мастера **Snapshot Folder** (Каталог моментальных снимков) вы должны определить каталог, в который будут помещаться моментальные снимки данных. По умолчанию предлагается использовать путь в локальной файловой системе. Однако при использовании такого каталога нельзя будет применять запрашивающие подписки, т. е. подписки, при которых процесс передачи данных будет инициировать подписчик. Для ваших целей вполне можно оставить каталог, предлагаемый по умолчанию. В практической работе правильнее было бы создать сетевую папку и предоставить на нее права на чтение и запись для той учетной записи, от имени которой работает SQL Server Agent. В SQL Server 2005 есть возможность определить для каждого агента свою учетную запись, но это используется редко.

На следующем экране **Distribution Database** (База данных распределения) вам потребуется определить имя и местонахождение файлов для создаваемой базы данных распределения. Отметим, что в этой базе данных будут хранить-

ся как настройки репликации, так и все изменения, которые передаются с издателя подписчикам. Поэтому при настройке репликации в реальной работе необходимо позаботиться о наличии для этой базы данных достаточного места на диске.

Следующее, что вам потребуется сделать, — на экране **Publishers** (Издатели) выбрать те серверы, которые смогут использовать этот распространитель. В вашем случае здесь достаточно выбрать только сервер LONDON2\SQL2005.

На последнем экране мастера **Wizard Actions** (Действия мастера) нужно дать команду на настройку распространителя и создание базы данных распределения. Отсюда же можно дать команду на генерацию скрипта с командами Transact-SQL, который будет выполнять выбранные вами на графическом экране действия.

После того, как вы нажмете кнопку **Finish** (Завершить), настройка распространителя будет закончена.

Следующее действие, которое обычно выполняется после настройки распространителя, — это создание публикации. Проще всего создать публикацию при помощи мастера New Publication Wizard. Запустить его можно при помощи команды **New Publication** (Новая публикация) контекстного меню контейнера **Replication | Local Publications** (Репликация | Локальные публикации) в SQL Server Management Studio.

Первое, что вам потребуется сделать, — выбрать на экране **Publication Databases** (Базы данных публикации) базу данных с информацией для публикации. В соответствии с условиями задания это должна быть база данных AdventureWorks.

На следующем экране **Publication Type** (Тип публикации) выберите тип публикации **Transactional Publication** (Транзакционная публикация).

На экране **Articles** (Статьи) вам потребуется создать статью для публикации. Для этого в списке объектов раскройте узел **Tables** (Таблицы) и установите флагки напротив всех таблиц схемы HumanResources. К сожалению, сортировки по схемам в этом окне не предусмотрено. Всего вам нужно установить флагки напротив семи таблиц.

На следующем экране **Filter Table Rows** (Фильтровать строки в таблицах) можно было бы отфильтровать записи в таблицах, использовав выражение WHERE. Однако в этом примере фильтрации не требуется. Отфильтровать столбцы можно было бы и на предыдущем экране. Для этого достаточно развернуть узел для таблицы и снять флагки напротив столбцов, которые не должны реплицироваться.

На следующем экране **Snapshot Agent** вы должны настроить параметры запуска агента снятия моментальных снимков. По умолчанию переключатель

стоит в положении **Create a snapshot immediately** (Создать моментальный снимок немедленно). Это значит, что моментальный снимок данных публикации будет сделан сразу после публикации и будет сохраняться в папке, чтобы можно было бы настраивать подписки. Второй вариант — настроить создание моментального снимка по расписанию (положение переключателя **Schedule the Snapshot Agent to run at the following times**). Этот вариант используется в тех ситуациях, когда данных в публикации очень много и желательно изготавливать снимок в ночное время. Другая ситуация — к этому серверу будут постоянно подключаться новые подписчики, которых придется инициализировать при помощи моментальных снимков (т. е. создавать исходные варианты таблиц). В этом случае регулярное создание моментальных снимков позволит сэкономить время (к подписчикам придется применять меньшее количество изменений).

В данном случае оставьте переключатель в верхнем положении, поскольку вам потребуется единственный моментальный снимок.

На следующем экране **Agent Security** (Безопасность агента) вы должны выбрать учетные записи, от имени которых будут работать Snapshot Agent и Log Reader Agent. Microsoft рекомендует для повышения безопасности использовать для каждого агента свою учетную запись с минимально необходимым набором прав. Но для большинства предприятий такая настройка будет неоправданно сложной. Лучше всего настроить работу агентов от имени учетной записи SQL Server Agent. Для этого на экране настройки безопасности агентов, который открывается при нажатии на кнопку **Security Settings** (Настройки безопасности), предназначен специальный флажок **Run under the SQL Service Agent Account** (Запускать под учетной записью SQL Service Agent).

После этого нажмите кнопку **Finish**, определите имя для публикации (назовите ее **HR**) и выполните ее создание.

Последнее, что вам осталось сделать для настройки репликации, — создать подписку. В этом примере будет использоваться принудительная подписка, в которой все агенты работают на распространителе. Это самый удобный с точки зрения централизации управления репликацией тип подписки.

Настроить подписку также проще всего при помощи мастера. Мастер создания подписки нужно запустить при помощи команды **New Subscriptions** (Новая подписка) контекстного меню контейнера **Local Subscriptions** (Локальные подписчики) для того сервера, который у вас является издателем и дистрибутором (обратите внимание: не для того сервера, который будет подписчиком!).

Первый важный выбор нужно будет произвести на экране мастера **Distribution Agent Location** (Местонахождение Distribution Agent). Здесь выбирается тип создаваемой подписки: принудительная или запрашивающая. В данном случае выберите принудительную подписку.

На следующем экране **Subscribers** (Подписчики) вы должны выбрать сервер, который будет подписчиком для вашей публикации. Скорее всего, нужного вам сервера в этом списке не будет. Чтобы его добавить, нужно нажать кнопку **Add Subscriber** (Добавить подписчика), а затем выбрать значение **New SQL Server Subscriber** (Новый подписчик SQL Server). После этого вам потребуется просто подключиться к данному серверу SQL Server с правами администратора.

Следующее, что вам потребуется сделать на этом экране, — выбрать базу данных, на которую будут передаваться реплицируемые данные. Вы можете предварительно создать нужную базу данных на сервере-подписчике вручную, а можете просто на этом экране выбрать значение **New Database** (Новая база данных) и создать новую базу данных. В данном случае создайте новую базу данных с параметрами по умолчанию и именем AW2.

На следующем экране **Distribution Agent Security** (Безопасность Distribution Agent) вам потребуется настроить учетную запись для Distribution Agent и настроить параметры его подключения к распространителю и подписчику. В этом примере используйте для него ту же учетную запись, что и для SQL Server Agent, а для остальных параметров оставьте значения по умолчанию.

Следующее, что вам потребуется сделать, — настроить расписание репликации. Согласно условию задания, репликация должна производиться с интервалом в 10 минут. Поэтому в списке **Agent Schedule** (Расписание агента) на экране **Synchronization Schedule** (Расписание синхронизации) выберите параметр **Define Schedule** (Определить расписание) и настройте параметры для создаваемого расписания. Соответствующие настройки в вашем случае могут выглядеть так, как представлено на рис. 12.1.

Другие доступные для вас варианты расписания — **Run continuously** (Запускать постоянно, т. е. репликация будет производиться непрерывно с минимальными интервалами) и **Run on demand only** (Запускать только по запросу).

На следующем экране мастера вы должны определиться со временем инициализации базы данных подписчика. В вашем распоряжении есть два варианта — **Immediately** (Немедленно) и **At first synchronization** (При первой синхронизации, т. е. во время первого сеанса репликации согласно настроеному расписанию). На этом же экране при необходимости можно снять флагок **Initialize** (Инициализация). В этом случае инициализация базы данных подписчика (т. е. создание исходного варианта таблиц) производиться не будет. Вам потребуется создать таблицы и загрузить их данными вручную. Такое решение может быть полезным, когда данных в подписке много, и вы хотите произвести изначальную загрузку данных для подписчика при помощи резервной копии.

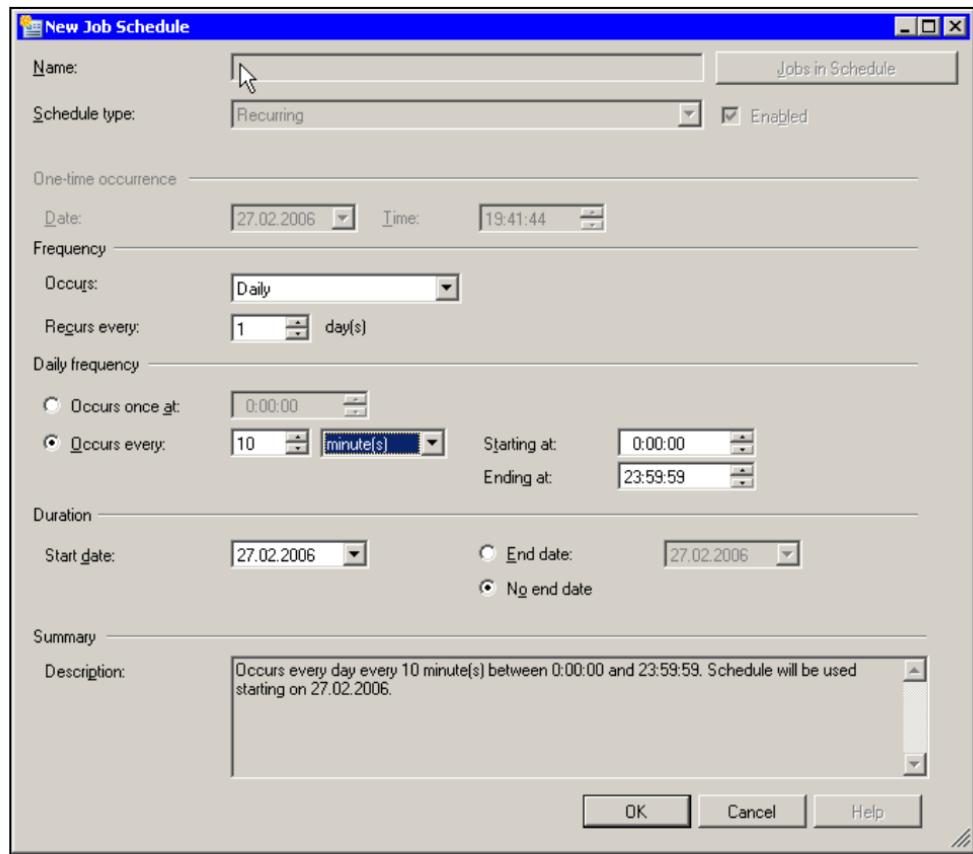


Рис. 12.1. Настройки расписания для репликации

На этом настройка параметров подписки (и всей системы репликации) закончена. Вам осталось только нажать кнопку **Finish** и произвести создание подписки. В базе данных подписки будет автоматически создан требуемый набор таблиц и будет произведена загрузка данных в эти таблицы. Если же вы внесете какие-либо изменения в опубликованные таблицы источника, то в течение 10 минут эти изменения будут отреплицированы и отображены в базе данных подписчика.

## 12.7. Средства администрирования и мониторинга репликации

### 12.7.1. Средства администрирования репликации

В предыдущем разделе была рассмотрена работа с мастерами, которые используются для настройки распространителей, издателей и подписчиков.

Однако часто возникает необходимость внести изменения в параметры репликации после завершения ее настройки.

Средства управления репликацией сконцентрированы в SQL Server Management Studio в контейнере **Replication**. Из контекстного меню этого контейнера можно открыть свойства распространителя (команда **Distributor Properties**) и свойства издателя (**Publisher Properties**). Кроме того, при помощи команды **Disable Publishing and Distribution** (Отключить публикацию и распределение) из этого же меню можно удалить все настройки репликации для этого компьютера (и, например, произвести настройку заново).

Публикации, которые есть на данном сервере, помещаются в контейнер **Local Publications**. Их свойства можно изменить из контекстного меню объекта соответствующей публикации. Обратите внимание, что здесь вам доступно большее число параметров репликации, чем в мастере. Например, вы можете настроить передачу файлов моментальных снимков при помощи протокола FTP или назначить точные права для данной публикации логинам SQL Server.

Если в системе репликации произошел сбой, то после устранения его причин проще всего вернуться к обычному режиму репликации, реинициализировав подписчиков. Это значит, что к ним будет заново применен моментальный снимок публикации, и репликация изменений фактически начнется заново. Реинициализировать всех подписчиков можно при помощи команды **Reinitialize all Subscriptions** (Реинициализировать все подписки) из контекстного меню объекта публикации. Реинициализация конкретного подписчика производится из контекстного меню данной подписки.

Для оптимизации производительности или для диагностических целей можно настроить некоторые низкоуровневые параметры работы агентов репликации. Такая настройка производится при помощи специальных профилей. Вы можете выбрать один из готовых профилей, который лучше всего подходит к данной ситуации, или создать и применить свой профиль. Работа с профайлами агентов производится из свойств распространителя. Нужно выбрать команду **Distributor Properties** контекстного меню контейнера **Replication** и нажать кнопку **Profile Defaults** (Настройки профиля по умолчанию). В открывшемся окне **Agent Profiles** (Профили агента) вы можете выбрать любой из заранее готовых профилей или создать свой профиль. Например, для **Distribution Agent** изначально предусмотрено пять профилей работы (рис. 12.2).

Отметим также, что репликацию можно настраивать не только графическими средствами Management Studio и мастеров. Все операции по настройке и администрированию можно произвести и при помощи специальных хранимых процедур. Их синтаксис достаточно сложен, и приводиться здесь не будет.

Необходимые команды для настройки репликации можно сгенерировать в автоматическом режиме. Для этого нужно воспользоваться командой **Generate Scripts** (Сгенерировать скрипты) контекстного меню контейнера **Replication** в Management Studio. Программный код получится достаточно большим. Например, для примера из предыдущего раздела (вместе со скриптами для соответствующих заданий SQL Server Agent) размер файла со скриптами составит 70 Кбайт.

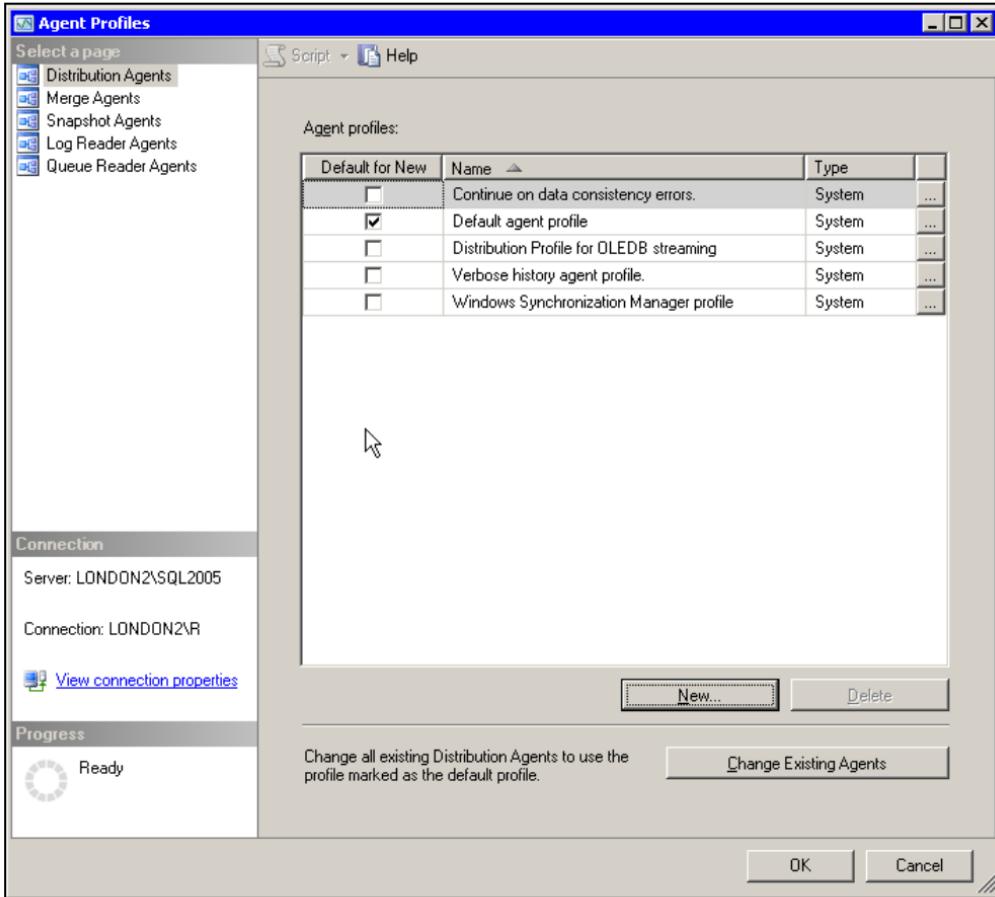


Рис. 12.2. Работа с профилями агентов репликации

Такие скрипты очень удобно использовать для копирования проверенной конфигурации репликации на другой сервер (например, для настройки серверов в филиалах) или для резервного копирования системы репликации.

## 12.7.2. Применение Replication Monitor

Если у вас настроена система репликации, рано или поздно вы столкнетесь с проблемами. Они могут возникнуть, например, из-за разрыва сетевых соединений, при нехватке свободного места на диске, при аутентификации, если контроллер домена был недоступен, и т. п. Поэтому важно уметь получать информацию о работе системы репликации и обнаруживать причины возникающих проблем.

Главное средство мониторинга и диагностики репликации — это Replication Monitor. По сравнению с Replication Monitor в SQL Server 2000 новая версия этой программы существенно изменилась. Открыть Replication Monitor можно при помощи команды **Launch Replication Monitor** (Запустить Replication Monitor), которая есть в контекстном меню практически любого объекта в контейнере **Replication** в Management Studio. Откроется окно **Replication Monitor**, аналогичное представленному на рис. 12.3.

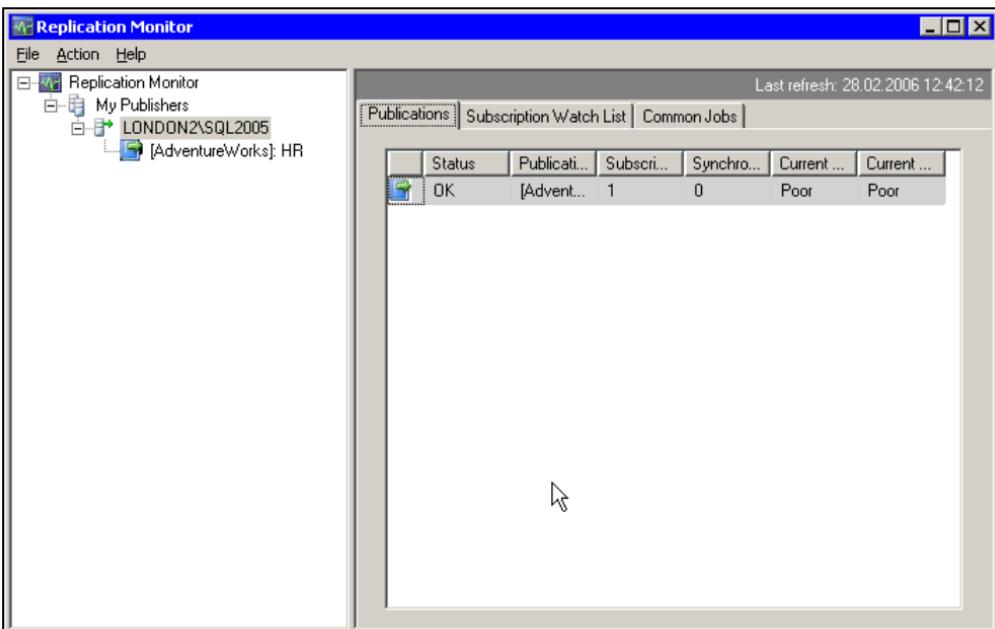


Рис. 12.3. Окно Replication Monitor

В левой части окна **Replication Monitor** находится список групп издателей, сами издатели и публикации. Группы издателей — это просто средство логического объединения разных издателей в окне **Replication Monitor**. Отнесение издателя к определенной группе не оказывает никакого влияния на него, кроме как на отображение в Replication Monitor. В группу вы можете самопод

сторону добавить другие серверы-издатели или серверы-распространители (в этом случае в группу будут добавлены не сами серверы-распространители, а все известные им издатели).

Список издателей в группе уже можно использовать для целей мониторинга репликации. Если в процессе репликации возникли какие-либо ошибки, то соответствующие иконки в этом списке будут помечены специальными символами. Однако намного больше информации вы можете получить из правой части окна **Replication Monitor**.

Вкладки в правой части окна **Replication Monitor** зависят от того, какой именно элемент выбран в списке слева. Если выбрана публикация, то в вашем распоряжении будут три вкладки:

- **Publications** — на ней вы можете просто просмотреть информацию о публикации, ее статус, количество настроенных для нее подписок и общую оценку производительности при работе с данной публикацией;
- **Subscription Watch List** (Список наблюдаемых подписок) — это список подписок для данной публикации. Очень подробную и ценную для диагностики информацию можно получить, если щелкнуть по объекту подписки на этой вкладке правой кнопкой мыши и в контекстном меню выбрать **View Details** (Просмотреть подробности). Откроется окно с информацией о всех сеансах передачи данных от издателя распространителю и от распространителя подписчику (на последней вкладке будет также дана информация о всех командах, которые пока не удалось выполнить для подписчика);
- **Common Jobs** (Обычные задания) — информация о заданиях, настроенных для данной публикации. Из контекстного меню объекта задания можно запустить его на выполнение вручную или остановить. Кроме того, если открыть свойства задания (при помощи команды **Properties** (Свойства) того же контекстного меню), то можно, например, изменить настроенное для него расписание, или, нажав на ссылку **View Job History** (Просмотреть историю заданий) на вкладке **General** (Общие), посмотреть подробную историю выполнения этого задания. Конечно, те же операции можно произвести обычным для заданий образом из контейнера SQL Server Agent в Management Studio.

Если в левой части экрана Replication Monitor перейти на уровень ниже и выбрать в списке объект публикации, то в вашем распоряжении появятся дополнительные возможности. На вкладке **Tracer Tokens** (Трассировочные маркеры) вы можете запустить специальный "пробный шар" — трассировочный маркер и посмотреть, как с его обработкой справится система реплика-

ции. Для этой цели предназначена кнопка **Insert Tracer** (Вставить трассировочный).

Физически трассировочный маркер — это специальный тип транзакции, информация о которой будет помещена в базу данных издателя и оттуда проследует к базе данных подписчика. Изменить данные в вашей системе при помощи этого маркера невозможно. Единственное его назначение — диагностика системы репликации. После того как такой маркер создан, вы можете посмотреть (на этой же вкладке), сколько времени заняла его передача от издателя распространителю, а затем от распространителя подписчику. В последнем столбце указывается общее время, которое потребовалось на обе операции. Если в каком-либо столбце осталось значение **Pending** (В ожидании), то это значит, что передача данных еще не произошла.

Последняя вкладка Replication Monitor, которая появляется, если выбрана публикация, — вкладка **Warnings and Alerts** (Предупреждения).

В верхней части этой вкладки вы можете настроить **Warnings** (Предупреждения). Эти предупреждения показываются в окне **Replication Monitor** в виде специальных обозначений.

При помощи кнопки **Configure Alerts** (Настроить предупреждения) можно настроить предупреждения SQL Server Agent, относящиеся к системе репликации. В принципе, все эти предупреждения создаются при настройке репликации автоматически (их можно просмотреть в контейнере **SQL Server Agent | Alerts** в Management Studio). Однако ни одному такому предупреждению по умолчанию не назначены операторы, поэтому реально оповещаться никто не будет. Назначить операторов или внести другие изменения в свойства предупреждений репликации можно при помощи той же кнопки **Configure Alerts** (Настроить предупреждения) в Replication Monitor или стандартными средствами SQL Server Management Studio (*см. разд. 8.1.7*).

В нижней части вкладки **Warnings and Alerts** можно просмотреть информацию о работе агентов, которые принимают участие в данном типе репликации.

По опыту работы автора с репликацией, можно сказать, что пороговые значения для мониторинга системы репликации в Replication Monitor SQL Server 2005 рассчитаны на очень производительную систему, работающую в идеальных условиях. В реальной работе задержки случаются очень часто (из-за скорости сетевых соединений, из-за загруженности сервера SQL Server и т. п.), поэтому паниковать, увидев знаки, сигнализирующие об ошибках, совершенно не нужно. В большинстве случаев все приходит в норму автоматически.

## 12.7.3. Другие средства мониторинга репликации

Replication Monitor — это очень мощное средство для получения информации о работе системы репликации, но не следует забывать и о других возможностях.

Во-первых, текущее состояние и последнее выполненное действие для различных агентов можно просмотреть просто из SQL Server Management Studio. Для этой цели предназначена команда **View Status** (Просмотреть состояние) контекстного меню для объектов публикации и подписки.

Во-вторых, информацию о самых важных проблемах репликации можно получить при просмотре журналов событий для самого SQL Server и SQL Server Agent. Определить, насколько подробно будет протоколироваться информация о репликации, можно самостоятельно. Для самого полного протоколирования достаточно выбрать для интересующих вас агентов профиль **Verbose history agent profile** (Профиль подробной истории агента). Такой профиль можно выбрать из числа готовых для Distribution Agent, Merge Agent и Log Reader Agent.

В-третьих, в вашем распоряжении есть средства SQL Server Agent. Обычно проблемы репликации проявляются в сбоях при выполнении заданий SQL Server Agent. Вы можете обычным способом просматривать историю выполнения этих заданий, анализировать причины проблем при запуске каждого из этапов, и отлаживать каждый этап задания вручную. Кроме того, вы можете настроить свои предупреждения для заданий репликации или использовать готовые предупреждения, которые создаются при настройке репликации автоматически.

Если репликация происходит недопустимо медленно или она оказывает сильное влияние на производительность сервера SQL Server, то для мониторинга и диагностики можно также использовать счетчики производительности в Системном мониторе. В вашем распоряжении есть следующие объекты:

- **SQL Server: Replication Snapshot** — счетчики для мониторинга работы Snapshot Agent;
- **SQL Server: Replication Dist.** — информация о работе Distribution Agent;
- **SQL Server: Replication Logreader** — счетчики Log Reader Agent;
- **SQL Server: Replication Merge** — статистика работы Merge Agent;
- **SQL Server: Replication Agents** — для этого объекта предусмотрен единственный счетчик, который показывает, сколько агентов работает в настоящий момент.

Вместо **SQL Server** в Системном мониторе нужно выбирать `MSSQL$имя_экземпляра` для вашего сервера.

# Задание для самостоятельной работы 12.1. Настройка одноранговой репликации

## Задание:

- Скопируйте базу данных AdventureWorks с экземпляра *имя\_вашего\_сервера\SQL2005* под новым названием AdventureWorksCopy на экземпляре *имя\_вашего\_сервера\Server2*.
- Настройте одноранговую репликацию между экземплярами *имя\_вашего\_сервера\SQL2005* и *имя\_вашего\_сервера\Server2* на вашем компьютере.

При этом:

- репликация должна производиться между базами данных AdventureWorks на экземпляре SQL2005 и базой данных AdventureWorksCopy на экземпляре Server2;
  - в репликации должны участвовать только таблицы из схемы Person;
  - распределителем должен быть сервер *имя\_вашего\_сервера\SQL2005*.
- Убедитесь, что репликация работает нормально.

## Решение:

К пункту 1 задания — копирование базы данных:

- В окне **Object Browser** в Management Studio подключитесь к серверу *имя\_вашего\_сервера\SQL2005*, раскройте контейнер **Databases** (Базы данных), щелкните правой кнопкой мыши по объекту базы данных AdventureWorks и в контекстном меню выберите **Tasks | Copy Database** (Задачи | Скопировать базу данных). Откроется окно мастера Copy Database Wizard. На первом экране этого мастера нажмите на кнопку **Next** (Дальше).
- На экране **Select a source server** (Выберите сервер-источник) выберите *имя\_вашего\_сервера\SQL2005*.
- На экране **Select a destination server** (Выберите сервер назначения) выберите *имя\_вашего\_сервера\Server2*.
- На экране **Select the Transfer Method** (Выберите метод передачи) оставьте переключатель в положении **Use the detach and attach method** (Использовать метод отсоединения и присоединения) и флажок **If a failure occurs, reattach the source database** (Если произошел сбой, присоединить исходную базу данных заново).
- На экране **Select Databases** (Выберите базы данных) убедитесь, что единственный флажок установлен в столбце **Copy** (Копировать) напротив базы данных AdventureWorks.

6. На экране **Configure Destination Database** (Настроить базу данных назначения) в поле **Destination Database** (База данных назначения) введите имя AdventureWorksCopy. В таблице в нижней части экрана измените значения в столбце **Destination Folder** (Каталог назначения). Вместо папки, предлагаемой по умолчанию, введите путь к другой папке, например, C:\.
7. На экране **Select Database Objects** (Выберите объекты базы данных) и на всех остальных экранах оставьте для всех параметров значения, предлагаемые по умолчанию, и на экране **Complete the Wizard** (Завершить работу мастера) нажмите кнопку **Finish**. Убедитесь, что копирование произведено без ошибок.

К пункту 2 задания — настройка одноранговой репликации:

#### Шаг 1. Настройка распределителя:

1. В SQL Server Management Studio щелкните правой кнопкой мыши по контейнеру **Replication** для сервера *имя\_вашего\_сервера\SQL2005* и в контекстном меню выберите **Configure Distribution**. Откроется мастер настройки распределения Configure Distribution Wizard. На его первом экране нажмите кнопку **Next**.
2. На экране **Distributor** убедитесь, что переключатель установлен в положение *имя\_вашего\_сервера\SQL2005 will act as its own Distributor* (*имя\_вашего\_сервера\SQL2005* будет работать как распределитель для самого себя).
3. На экранах **Snapshot Folder** и **Distribution Database** оставьте значения, предлагаемые по умолчанию.
4. На экране **Publisher** нажмите кнопку **Add** (Добавить) и в открывшемся списке выберите **Add SQL Server Publisher** (Добавить издателя SQL Server). В открывшемся окне подключитесь к серверу *имя\_вашего\_сервера\Server2* и убедитесь, что он появился в списке подписчиков вместе с сервером *имя\_вашего\_сервера\SQL2005*.
5. На экране **Distributor Password** (Пароль дистрибутора) введите два раза пароль P@ssw0rd.
6. На экране **Wizard Actions** убедитесь, что переключатель установлен в положение **Configure distribution** и нажмите кнопку **Next**, а затем **Finish**. Убедитесь, что работа мастера завершена без ошибок.

#### Шаг 2. Назначение распределителя второму серверу:

1. Из SQL Server Management Studio подключитесь к серверу *имя\_вашего\_сервера\Server2*, щелкните правой кнопкой мыши по контейнеру **Replication** для этого сервера и в контекстном меню выберите **Configure Distribution**. На первом экране мастера настройки распределения нажмите кнопку **Next**.

2. На экране **Distributor** переставьте переключатель в положение **Use the following server as the Distributor** (Использовать следующий сервер как распределитель), нажмите кнопку **Add** и подключитесь к серверу *имя\_вашего\_сервера\SQL2005*. Убедитесь, что этот сервер добавлен в список распределителей, и нажмите кнопку **Next**.
3. В ответ на приглашение введите два раза пароль *P@ssw0rd*. На экране **Wizard Actions** убедитесь, что переключатель установлен в положение **Configure distribution** и нажмите кнопку **Next**, а затем **Finish**. Убедитесь, что работа мастера завершена без ошибок.

#### Шаг 3. Создание публикации:

1. В окне SQL Server Management Studio раскройте контейнер **Replication | Local Publications** для сервера *имя\_вашего\_сервера\SQL2005* и щелкните по нему правой кнопкой мыши, а затем в контекстном меню выберите **New Publication**. На первом экране мастера New Publication Wizard нажмите кнопку **Next**.
2. На экране **Publication Database** выберите базу данных AdventureWorks.
3. На экране **Publication Type** выберите тип **Transactional Publication**.
4. На экране **Articles** раскройте контейнер **Tables** и установите флажки напротив таблиц Address (Person), AddressType (Person), Contact (Person), ContactType (Person), CountryRegion (Person) и StateProvince (Person).
5. На экране **Filter Table Rows** нажмите кнопку **Next**.
6. На экране **Snapshot Agent** убедитесь, что оба флажка сняты, и нажмите кнопку **Next**.
7. На экране **Agent Security** нажмите кнопку **Security Settings** для агента Snapshot Agent и установите переключатель в положение **Run under the SQL Server Agent account**. Убедитесь, что значение SQL Server Agent Account появилось для Snapshot Agent и Log Reader Agent и нажмите кнопку **Next**.
8. На экране **Wizard Actions** убедитесь, что переключатель находится в положении **Create the publication** (Создать публикацию) и нажмите кнопку **Next**.
9. На экране **Complete the Wizard** введите имя для публикации (например, PersonPublication) и нажмите кнопку **Finish**. Убедитесь, что создание публикации прошло без ошибок.

#### Шаг 4. Настройка свойств публикации:

1. Раскройте в SQL Server Management Studio для вашего сервера контейнер **Replication | Local Publications** и щелкните правой кнопкой мыши по соз-

данной вами публикации PersonPublication. Выберите в контекстном меню команду **Properties** и в окне свойств перейдите на вкладку **Subscription Options** (Параметры подписки). На этой вкладке для параметра **Allow peer-to-peer publication** (Разрешить одноранговую публикацию) установите значение **True** и нажмите кнопку **OK**.

#### Шаг 5. Настройка топологии одноранговой репликации:

- Еще раз щелкните правой кнопкой мыши на публикации PersonPublication и в контекстном меню выберите команду **Configure Peer-to-Peer Topology** (Настроить одноранговую топологию). На первом экране мастера настройки одноранговой топологии Configure Peer-to-Peer Topology Wizard нажмите кнопку **Next**.
- На экране **Publication** выберите предлагаемую по умолчанию публикацию PersonPublication и нажмите кнопку **Next**.
- На экране **Peer Server Instance** (Экземпляр однорангового сервера) нажмите кнопку **Add SQL Server** (Добавить SQL Server) и подключитесь к серверу *имя\_вашего\_сервера\Server2*. Затем в списке **Databases** для этого сервера выберите базу данных AdventureWorksCopy. Убедитесь, что напротив сервера Server2 установлен флажок, и нажмите кнопку **Next**.
- На экране **Log Reader Agent Security** (Безопасность Log Reader Agent) нажмите кнопку напротив каждой из строк таблицы и на обоих экранах установите переключатель в положение **Run under the SQL Server Agent service account**. Такие же настройки нужно произвести на вкладке **Distribution Agent Security** (Безопасность Distributor Agent).
- На экране **New Peer Initialization** (Инициализация нового однорангового сервера) установите переключатель в положение **I created the peer database manually** (Я создал базу данных вручную) и нажмите кнопку **Next**, а затем кнопку **Finish**. Убедитесь, что работа мастера завершена без ошибок. Если при этом возникнет предупреждение **Publication is already exists** (Публикация уже существует), проигнорируйте его.

#### Шаг 6. Проверка репликации.

Для проверки репликации можно:

- запустить Replication Monitor на каждом сервере (команда **Launch Replication Monitor** контекстного меню для контейнера **Replication**), выделить нужную публикацию, перейти на вкладку **Tracer Tokens** и нажать кнопку **Insert Tracer**. Маркер в обоих направлениях должен проходить за несколько секунд;
- внести изменения в таблицы в схеме Person на любом из серверов, принимающих участие в репликации, и убедиться, что в течение нескольких секунд эти изменения отобразятся на втором сервере.

# Предметный указатель

**64**

64-битная версия, отличия 27

**A**

Access, перенос баз данных на SQL Server 53  
ActiveX Script Task 366  
Alerts 264  
Analysis Services 35  
Analysis Services Deployment Wizard 90  
Analysis Services Instance Rename 90  
Analysis Services Migration Wizard 90  
APPLY, оператор 16  
AUTO\_UPDATE\_STATISTICS\_ASYNC, параметр 6

**B**

bcp, утилита 92  
Best Practices Analyzer 457  
Bulk Insert Task 368  
Business Intelligence Development Studio  
◊ назначение 74  
◊ типы проектов 75

**C**

CDO, объектная модель 281  
cidump, утилита 92  
CLR Integration 12  
Common Table Expression 16  
Configure Web Synchronization Wizard 91  
Copy Database Wizard 91  
CREATE CERTIFICATE, команда 170  
CREATE DATABASE, команда 100  
CREATE LOGIN, команда 134

CREATE USER, команда 146  
Credential, объект 258

**D**

Data Flow Task 352  
Database Engine Tuning Advisor  
◊ назначение 88  
◊ новые возможности 89  
Database Mail, настройка 273  
Database Master Key 171  
DBStressUtil, утилита 429  
Dedicated Administrator Connection (DAC) 83  
DENY, команда 152  
distmodel, база данных 98  
distribution, база данных 98  
dta, утилита 92  
dtattach, утилита 92  
DTExec, утилита 92  
DTSRun, утилита 92  
DTUtil, утилита 92

**E**

Event Notifications 421  
EXECUTE AS, выражение 155  
Execute DTS 2000 Package Task 374  
Execute Package Task 374  
Execute Package Utility 91  
Execute Process Task 378  
Execute SQL Task 369  
Extended properties 119

**F**

File System Task 377  
For Loop, контейнер SSIS 384

Foreach Loop, контейнер SSIS 384

FTP Task 377

## G

GRANT, команда 152

## I

IMPERSONATE, разрешение 156

## L

Log shipping 213

Logman, утилита 426

lrltest, утилита 92

## M

Management Studio

- ◊ Object Explorer 64

- ◊ Solution Explorer 66

- ◊ Template Explorer 68

- ◊ назначение 61

- ◊ настройка 62

- ◊ регистрация серверов 62

- ◊ редактор кода 66

MARS 19

master, база данных 97

MAXDOP, параметр 5

Message Queue Task 371

Microsoft Loopback Adapter 30

MigrationWizardConsole, утилита 92

model, база данных 98

msdb, база данных 98

Multiple Active Result Set 19

## N

Named Pipes, сетевая библиотека 55

Native XML Web Services 14

Network Monitor 442

Notification Services 36

nscontrol, утилита 93

nsservice, утилита 93

## O

Operators, SQL Server Agent 270

OPTIMIZE FOR, хинт оптимизатора

19

ORCA, компонент 428

osql, утилита 93

Ostress, утилита 428

OUTPUT, выражение для INSERT,

DELETE, UPDATE 16

## P

PARAMETRIZATION, хинт

оптимизатора 19

PERSISTED, параметр 18

PIVOT, оператор 16

Principals, объекты 130

PSSDiag, утилита 93

## R

Read80Trace, утилита 428

RECOMPILE, хинт оптимизатора 19

Relog, утилита 426

Replication Conflict Viewer 90

Replication Monitor 91, 504

Replication Table Diff Tool 93

Reporting Services 35

Reporting Services Configuration 91

resource, база данных 98

RESTORE, команда 198

REVOKE, команда 152

RML, язык 428

rsactivate, утилита 93

rsconfig, утилита 93

RSKeyMgmt, утилита 93

## S

Script Task 366

Securable 130

Send Mail Task 377

Sequence, контейнер SSIS 386

Service Master Key 171

Shared Memory, сетевая библиотека 55

**SMO:**

- ◊ SMO.Database, объект 305
- ◊ SMO.Server, объект 300
- ◊ источники информации 297
- ◊ общие свойства и методы объектов 298

◊ основные возможности 295  
sp\_who и sp\_who2, хранимые процедуры 409

**SQL Native Client 80****SQL Server Configuration Manager:**

- ◊ назначение 75
- ◊ настройка сетевых библиотек 78
- ◊ работа со службами 77

**SQL Server Import and Export Wizard 91****SQL Server Integration Services Migration Wizard 91****SQL Server Management Studio:**

- ◊ Object Explorer 64
- ◊ Solution Explorer 66
- ◊ Template Explorer 68
- ◊ назначение 61
- ◊ настройка 62
- ◊ регистрация серверов 62
- ◊ редактор кода 66

**SQL Server Profiler:**

- ◊ назначение 86
- ◊ новые возможности 87

**SQL Server Surface Area Configuration:**

- ◊ назначение 84
- ◊ настройка Analysis Services 86
- ◊ настройка возможностей сервера 85
- ◊ настройка служб 85

**SQLCmd:**

- ◊ Dedicated Administrator Connection (DAC) 83
- ◊ назначение 81
- ◊ режимы работы 82

**SQL-DMO:**

- ◊ SQLDMO.Application, объект 308
- ◊ SQLDMO.Database2, объект 311
- ◊ SQLDMO.Server2, объект 308
- ◊ возможности 306

SQLMail, настройка 273

**SqlLiMailWizard, утилита 93**

sqlmaint, утилита 93, 194, 288

**SSIS:**

- ◊ ActiveX Script Task 366
- ◊ Bulk Insert Task 368
- ◊ Data Flow Task 352
- ◊ Execute DTS 2000 Package Task 374
- ◊ Execute Package Task 374
- ◊ Execute Process Task 378
- ◊ Execute SQL Task 369
- ◊ File System Task 377
- ◊ For Loop, контейнер 384
- ◊ Foreach Loop, контейнер 384
- ◊ FTP Task 377
- ◊ Message Queue Task 371
- ◊ Script Task 366
- ◊ Send Mail Task 377
- ◊ Sequence, контейнер 386
- ◊ Transfer Database Task 375
- ◊ Web Services Task 378
- ◊ WMI Data Reader Task 380
- ◊ WMI Event Watcher Task 380
- ◊ XML Task 370
- ◊ безопасность пакетов 395
- ◊ источники и назначения 354
- ◊ конфигурации пакетов 391
- ◊ мастер импорта и экспорта данных 339
- ◊ менеджеры подключений 349
- ◊ ограничения предшественников 387
- ◊ основные возможности 335
- ◊ планы обслуживания 386
- ◊ преобразования 356
- ◊ программные средства 336
- ◊ протоколирование выполнения пакетов 389
- ◊ пути потока данных 364
- ◊ работа с SSIS Designer 343
- ◊ хранение пакетов 393

**Support Escalation Services Support Utilities, утилиты нагружочного тестирования 428**

**System Configuration Checker 35**

**T**

- tablediff, утилита 93
- TABLESAMPLE, выражение 18
- TCP/IP, сетевая библиотека 55
- tempdb, база данных 98
- Transfer Database Task 375
- TRY...CATCH, синтаксическая конструкция 15
- Typeperf, утилита 426

**U**

- UNPIVOT, оператор 16
- USE PLAN, хint оптимизатора 19

**V**

- VIA, сетевая библиотека 55

**W**

- WBEMTest, утилита 316
- Web Services Task 378
- WMI:
  - ◊ ODCB-драйвер для WMI 317
  - ◊ Provider for Configuration Management 315, 326
  - ◊ Provider for Server Events 315, 328
  - ◊ запросы WQL 320
  - ◊ компоненты архитектуры 314
  - ◊ основные возможности 313
  - ◊ подключение к службе 319
  - ◊ поставщики для работы с SQL Server 315
  - ◊ программные средства 316
  - ◊ событийные запросы 323
- WMI Data Reader Task 380
- WMI Event Watcher Task 380
- WQL, язык запросов 320

**X**

- XML:
  - ◊ индексирование столбцов XML 19
  - ◊ тип данных 15
- XML Task 370

**A**

- Автоматизированная установка 49

**B**

- Базы данных:
  - ◊ журналы транзакций 108
  - ◊ изменение владельца 124
  - ◊ набор файлов 104
  - ◊ отсоединение 103
  - ◊ параметры 114
  - ◊ перевод в режим:
    - OFFLINE 103
    - ONLINE 103
  - ◊ переименование 124
  - ◊ перенос файлов 123
  - ◊ применение неразмеченные разделов 105
  - ◊ присоединение файлов 102
  - ◊ проверка целостности 125
  - ◊ работа с файловыми группами 108
  - ◊ расширенные свойства 119
  - ◊ режимы:
    - восстановления 110
    - работы 112
  - ◊ скрытые 98
  - ◊ служебные 97
  - ◊ создание 99, 100, 102
  - ◊ увеличение размера 121
  - ◊ удаление 124
  - ◊ уменьшение размера 121
  - ◊ учебные 99
- Блокировки SQL Server 475

**B**

- Восстановление баз данных:

- ◊ оперативный режим 203
- ◊ основы 195
- ◊ параметры 198
- ◊ планирование 196
- ◊ подготовка 197
- ◊ постраничное 204
- ◊ проверка резервных копий 198
- ◊ системных 205
- Время отклика 425

**Г**

- Генерация кода скриптов 70  
 Гиды по планам выполнения 485  
 Главный ключ:  
   ◊ базы данных 171  
   ◊ службы 171

**Д**

- Дефрагментация индексов и таблиц 467  
 Динамические представления 410  
 Доставка журналов:  
   ◊ мониторинг 221  
   ◊ настройка 215  
   ◊ отмена 226  
   ◊ переключение серверов 224  
   ◊ преимущества 213  
   ◊ режимы работы 228  
   ◊ терминология 214

**Ж**

- Журналы SQL Server 422

**З**

- Задания:  
   ◊ мультисерверные 261  
   ◊ просмотр истории выполнения 260  
   ◊ создание 248  
   ◊ типы этапов 250  
 Зеркальное отображение:  
   ◊ мониторинг 232  
   ◊ настройка 229  
   ◊ преимущества 227  
   ◊ приостановка и отмена 235  
   ◊ смена ролей 233  
   ◊ терминология 229

**И**

- Изменение контекста выполнения 155  
 Измерения производительности при рабочей нагрузке 425  
 Именованные экземпляры 38

**К**

- Кластеры:  
   ◊ конфигурация  
     ▫ активный/активный 211  
     ▫ активный/пассивный 211  
   ◊ преимущества 209  
   ◊ терминология 210  
   ◊ установка 212  
 Ключ:  
   ◊ асимметричный 172  
   ◊ симметричный 172  
 Кодировки SQL Server 46  
 Компоненты установки 35

**Л**

- Ленточные библиотеки 181  
 Логин 131  
 Логины, создаваемые по умолчанию 140  
 Логические устройства резервного копирования 182

**М**

- Мониторинг подключений пользователей 408

**Н**

- Нагрузочное тестирование 427

**О**

- Обновление предыдущих версий 50  
 Одноранговая репликация 494  
 Оператор последней надежды 247  
 Операторы SQL Server Agent 270  
 Оптимизация:  
   ◊ архитектуры приложения 453  
   ◊ блокировок 475  
   ◊ запросов 480  
   ◊ индексов 462  
   ◊ операционной системы 453  
   ◊ подключений 459  
 Ошибки, перехват 266

**П**

- Пароль, шифрование данных в таблицах 173
- Парольные политики 134
- Планы:
  - ◊ выполнения запросов 482
  - ◊ обслуживания баз данных 284, 386
- Пользователи баз данных 144
- Пользовательские типы данных .NET 13
- Порты SQL Server 56
- Порядок сортировки 46
- Послеустановочные задачи 57
- Постройтель запросов 70
- Предупреждения SQL Server Agent 264
- Принципалы 130
- Прокси-записи SQL Server Agent 257
- Пропускная способность 425
- Профилязовщик 410
  - ◊ назначение 86
  - ◊ новые возможности 87
- Псевдонимы 81

**Р**

- Разрешения в базах данных 151
- Редакции SQL Server 2005 32
- Режимы:
  - ◊ аутентификации 43
  - ◊ восстановления баз данных 110
- Резервное копирование:
  - ◊ логические устройства 182
  - ◊ отчеты 194
  - ◊ параметры 188
  - ◊ планирование 180
  - ◊ расписание 186
  - ◊ типы 183
- Репликация:
  - ◊ Replication Monitor 504
  - ◊ настройка 496
  - ◊ новые возможности 489
  - ◊ одноранговая 494
  - ◊ определение 487
  - ◊ подготовка к настройке 494

- ◊ средства управления 501
- ◊ терминология 490
- ◊ типы 492
- ◊ трассировочные маркеры 505
- Роли:
  - ◊ баз данных 148
    - встроенные 147
  - ◊ приложений 153
  - ◊ серверов 141
- Руководства по запросам 18

**С**

- Секционирование 4
- Сертификаты:
  - ◊ для защиты сетевого трафика 162
  - ◊ назначение 158
  - ◊ шифрование данных в таблицах 170
- Сетевой трафик SQL Server, защита 160
- Сетевые библиотеки 54
  - ◊ клиентские 79
  - ◊ серверные 78
- Системный монитор:
  - ◊ приемы работы с объектами и счетчиками 433
  - ◊ режимы работы 430
  - ◊ счетчики:
    - SQL Server 444
    - диска 439
    - памяти 437
    - процессора 434
    - сети 441
- Скрипты, генерация кода 70
- Службы SQL Server 2005 77
- Средства мониторинга производительности 426
- Статистика 116
  - ◊ оптимизатора 481
- Схемы в базах данных 145

**Т**

- Трассировочные маркеры 505
- Триггеры DDL 419

## У

- Уведомления:  
◊ об изменениях в базе данных 17  
◊ о событиях 421  
Узкие места системы 425  
Уровень изоляции моментальных снимков 5  
Устаревание пароля 136  
Учетные записи служб SQL Server 40

## Ц

- Центр сертификации 164

## Э

- Эталонный график производительности 424  
Этапы заданий SQL Server Agent 250

## Ф

- Файловые группы 108  
Форсированная параметризация 18