top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Microsoft SQL Server

Программирование
и администрирование СУБД

Урок №7

Резервное копирование и восстановление

Содержание

1. Резервное копирование БД и восстановление из резервной копии

1.1. Основные понятия

Создавать базу данных мы научились, а также разобрались с организацией безопасности данных на сервере. Но это еще далеко не все.

После этого стоит позаботиться об оптимизации ее деятельности и безопасность данных в случае возникновения непредвиденных ситуаций, ведь в процессе работы всякое случается: пожар, потоп, землетрясение, разного рода сбои в аппаратной части, ошибки в работе пользователей (например, были случайно удалены или изменены данные) тому подобное.

Для таких случаев очень полезно иметь под рукой копию данных, которую можно восстановить в случае потери информации. Осуществляется это с помощью **механизма резервного копирования**.

Резервное копирование (backup) базы данных и восстановления из резервной копии (restore) – два важных и наиболее частых административных процесса, которые осуществляются разработчиками и администраторами базы данных. В SQL Server 2008 для этого можно воспользоваться операторами T-SQL или использовать набор инструментов SSMS.

Стоит отметить, что регулярное выполнение backup / restore базы данных и журнала транзакций позвляют избежать потери данных.

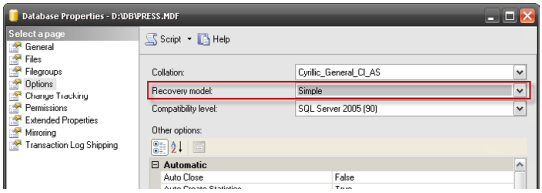
Все типы резервного копирования, которые поддерживаются SQL Server 2008, зависят **от модели восстановления базы данных (recovery model)**. Они также определяют, как SQL Server будет работать с журналом транзакций, будет его регистрировать и очищать (точнее обрезать (truncating) журнал, то есть удалять зафиксированы транзакции и высвобождать места для регистрации новых). SQL Server 2008 поддерживает **три модели восстановления базы данных:**

1. **Модель полного восстановления (Full recovery model)** – в журнале транзакций будут регистрироваться все операции и очистки журнала происходить не будет. Данная модель позволяет полностью восстановить базу данных до состояния на момент аварийного завершения ее работы.
2. **Простая модель восстановления (Simple recovery model)** – регистрирует минимум данных о большинстве транзакций и выполняет очистку журнала транзакций после каждой контрольной точки. Таким образом, формируется удобный и компактный журнал транзакций, который обеспечивает достаточно высокое повышение производительности, но не будет пригодна для восстановления. В связи с этим, данная модель не поддерживает резервное копирование и восстановление журнала транзакций, а также не позволяет восстанавливать отдельные страницы.

3. **Модель с неполным протоколированием (Bulk-Logged recovery model)** – является упрощенной моделью полного восстановления, в которой об обычные операции в журнал транзакции записывается полная информация, а вот регистрация информации о массовых операции (SELECT INTO и BULK INSERT) сводится к минимуму. Итак, если в резервной копии журнала транзакций содержатся какие-либо массовые операции, базу данных можно восстановить только к состоянию, которое соответствует концу резервной копии журнала, то есть нельзя восстановить на определенный период времени. Эта модель восстановления используется только для больших массовых операций.

На практике рекомендуется использовать модель полного восстановления базы данных, поскольку она обладает наибольшими возможностями восстановления. Если же вы используете массовые операции для импорта данных, лучше в такой период менять модель восстановления на модель с неполным протоколированием, поскольку она позволяет повысить производительность массовых операций.

Модель восстановления базы данных можно изменить средствами SSMS на странице **Properties (Свойства)** базы данных.



А также с помощью инструкции **ALTER DATABASE** с последующим синтаксисом:

```
-- синтаксис инструкции ALTER DATABASE для изменения модели
 восстановления базы данных
 ALTER DATABASE имя_базы_данных
 SET RECORERY { FULL | SIMPLE | BULK_LOGGED }
-- например, установить полную модель восстановления базы
 данных
 alter database Press
 set recovery full;
```

Посмотреть информацию о текущей модели восстановления можно с помощью системного представления **sys.databases:**

```
select name, recovery_model_desc
from sys.databases;
```

Результат:

Results | Messages

	name	recovery_model_desc
1	master	SIMPLE
2	tempdb	SIMPLE
3	model	FULL
4	msdb	SIMPLE
5	D:\DB\PRESS.MDF	SIMPLE
6	Test	FULL

1.2. Резервное копирование

1.2.1. Виды резервного копирования

Резервное копирование базы данных – это процесс считывания всех данных из базы данных и сохранения

их в виде одного или нескольких файлов на диске или устройства резервного копирования. Резервные копии можно делать для всей базы данных или ее части, набора файлов или файловых групп.

Важно знать, что все операции резервного копирования выполняются в контексте безопасности учетной записи пользователя, поэтому такому пользователю необходимо предоставить соответствующие права. К необходимым правам следует отнести права на чтение и запись в произвольные каталоги или дисковые накопители, которые будут использоваться для создания резервной копии, а также он должен быть членом фиксированной роли уровня базы данных **db_backupoperator.**

Существуют следующие **виды резервного копирования:**

1. **Полное резервное копирование базы данных (full database backup)** – позволяет сохранить текущее состояние всех данных, которые хранятся в базе. Для этого механизм резервного копирования извлекает все экстенты базы данных, которые выделены для объектов.

(Экстент – это основная единица организации пространства в SQL Server. Экстент представляет собой коллекцию, которая состоит из 8 физических непрерывных страниц (64 Кб). Страница – это основная единица источника данных в SQL Server. Место на диске, где находится файл базы данных, логично делится на непрерывные страницы по 8 Кб с нумерацией от 0 до n. Все дисковые операции ввода-вывода выполняются на уровне страниц. Все страницы сохраняются в экстентах.

Файлы журнала транзакций не содержат страниц, в них размещается последовательность записей журнала (!).)

2. **Дифференциальное резервное копирование базы данных (differential database backup)** – сохраняет все экстенты, которые были изменены с момента последнего полного резервного копирования (!). Оно позволяет уменьшить количество резервных копий журнала транзакций, которые необходимо восстановить в случае потери данных.

3. **Резервное копирование журнала транзакций (transaction log)** – позволяет сохранить активный журнал (active log), то есть данные журнала, которые начинаются с номера транзакции (LSN), на котором закончилось предыдущее резервное копирование журнала транзакций. После этого сохраняются все последующие транзакции до тех пор, пока не будет обнаружена открытая транзакция.

4. **Резервное копирование файловых групп (filegroup backup)** – позволяет создавать резервное копирование отдельных файловых групп базы данных.

Для выполнения дифференциального резервного копирования и резервного копирования журнала транзакций необходимо создать полную резервную копию. Но работают они отдельно друг от друга.

Подытоживая, отметим также следующие моменты:

- использование полных резервных копий вместе с дифференциальными и резервными копиями журналов транзакций позволяют сохранить всю

базу данных, включая все изменения, которые произошли с момента последнего полного резервного копирования;
- для создания копий фрагментов базы данных лучше всего использовать резервное копирование файловых групп вместе с дифференциальными резервными копиями и копиями журнала транзакций.

1.2.2. Резервное копирование базы данных

Механизм резервного копирования базы данных записывает на устройство резервного копирования страницы базы данных, без учета их порядка. В связи с этим данные могут записываться в несколько потоков, что делает операцию резервного копирования достаточно быстрой. Скорость резервного копирования зависит только от скорости записи данных на устройство.

Чтобы на выходе получить неповрежденные данные (ведь во время резервного копирования к ним могут обращаться другие пользователи и выполнять при этом вставку, модификацию или удаление данных), SQL Server **осуществляет полное резервное копирование следующим образом**:

- Сначала блокируется база данных и все транзакции.
- В журнал транзакций добавляется маркер.
- Снимается блокировка базы данных.
- Выполняется резервное копирование всех страниц базы данных.
- Снова осуществляется блокировка базы данных и всех транзакций.
- В журнал транзакций добавляется маркер.

- Снимается блокировка базы данных.
- Извлекаются все транзакции между двумя маркерами в журнале транзакций и добавляются в резервную копию.

Для резервного копирования используется также карта размещения экстентов. Эта карта используется для определения экстентов, которые должны войти в резервную копию, и представляет собой отдельную страницу данных в базе данных, каждый бит которой представляет один экстент. При полном резервном копировании все биты скидываются на 0. После этого, если экстент меняется, SQL Server изменяет бит, который ему соответствует с 0 на 1. Таким образом, при дифференциальном резервном копировании значение битов карты анализируется и в резервную копию входят только экстенты, которые были изменены (то есть их бит равен 1).

Заметим, что для базы данных master можно осуществлять только полное резервное копирование.

Ну и наконец, резервное копирование базы данных осуществляется с помощью оператора **BACKUP DATABASE:**

```
BACKUP DATABASE { база_данных | @переменная_с_именем_БД
}  -- база данных или ее часть

--(объект)
TO устройство_резервного_копирования [ ,...n ]
[ MIRROR TO устройство_резервного_копирования,...n ]] --
зеркало для основного устройства резервного копирования
- параметры операций резервного копирования
[ WITH [ DIFFERENTIAL ]  -- Резничная резервная копия
[ общие_опции_WITH [ ,...n ] ]
]
```

```
-- устройство_резервного_копирования:
{ имя_логичного_устройства_резервного_копирования
                                | @переменная_имени_
логичного_устройства_резервного_копирования }
| { DISK | TAPE } = { 'имя_физического_устройства_
резервного_копирования'
                    | @переменная_имени_физического_
устройства_резервного_копирования }
```

Согласно вышеописанного синтаксиса, сначала необходимо указать название базы данных (или ее части), резервное копирование которой необходимо сделать. После ключевого слова **TO** указывается устройство резервного копирования, на котором будет храниться резервная копия. Им может быть как имя созданного логического устройства резервного копирования, так и полный путь на диске или ленточном устройстве.

ПРИМЕЧАНИЕ! В следующей версии SQL Server поддержка ленточных устройств будет удалена. Поэтому следует избегать использования параметра TAPE.

Параметр **MIRROR TO** служит для описания устройства (-в) резервного копирования, который будет зеркалом для основного устройства резервного копирования. Максимальное количество устройств для зеркального резервного копирования – три. При этом, указано количество устройств в параметре MIRROR TO должна иметь тот же тип и то же количество, что и в параметре TO.

Параметр **WITH** позволяет задать дополнительные необязательные параметры архивации, а также указать необходимость создания дифференциальной резервной

копии базы данных (по умолчанию создается полная резервная копия). В качестве общих опций параметра WITH могут выступать:

1. Параметры резервного набора данных, который создается текущей операцией резервного копирования.
   - COPY_ONLY – Резервная копия только для копирования. Если параметры DIFFERENTIAL и COPY_ONLY используются вместе, тогда параметр COPY_ONLY игнорируется и создается дифференциальная резервная копия.
   - COMPRESSION|NO_COMPRESSION – используется компрессия (compression) данной резервной копии.
   - DESCRIPTION = { 'текст' | @переменная_текста} – Текстовое описание резервного набора данных, который может содержать до 255 символов.
   - NAME = { 'имя' | @переменная_имени} – имя резервного набора данных, которое не должно превышать 128 символов. По умолчанию имя отсутствует.
   - PASSWORD = { 'пароль' | @переменная _пароля} – Пароль на резервный набор данных. В следующей версии SQL Server эта опция будет удалена.
   - EXPIREDATE = { 'дата' | @переменная _даты} и RETAINDAYS = {к-во_дней | @переменная _к-ва_дней} – Время, которое определяет, когда эту резервную копию можно перезаписать. Эти параметры могут указываться вме-

сте, но в таком случае RETAINDAYS имеет приоритет над EXPIREDATE.

Если ни один из этих параметров не задан, тогда срок хранения определяется параметром конфигурации mediaretention.

Если указанные параметры задаются с помощью переменной, тогда они должны иметь один из следующих типов:

- для EXPIREDATE:
  - Строчной константой;
  - Символьной строкой, кроме типов данных ntext и text;
  - Smalldatetime;
  - Datetime.

  для RETAINDAYS – целым числом.

2. Параметры набора носителей, на которые записываются резервные копии:
   - NOINIT | INIT – определяют, будет ли операция резервного копирования перезаписывать (INIT) резервные наборы данных, которые уже существуют на носителе резервной копии, или дописывать (NOINIT) новые наборы данных в конец.
     По умолчанию используется опция NOINIT.
   - NOSKIP | SKIP – определяет, будет ли операция резервного копирования проверять (NOSKIP) дату и время окончания срока резервной копии на носителе перед их перезаписью.
     Опция NOSKIP установлена по умолчанию.
   - NOFORMAT | FORMAT – должен (FORMAT) или нет (NOFORMAT) заголовок носителя записываться (или

перезаписываться) на томах, которые используются текущей операцией резервного копирования.
В связи с этим, опцию FORMAT следует использовать очень осторожно, так как форматирование любого тома из набора носителей повреждает всю резервную копию.

- MEDIADESCRIPTION = { 'текст' | @переменная _текста} – Текстовое описание набора носителей, который не должен превышать 255 символов.
- MEDIANAME = { 'имя_носителя " | @переменная _имени} – имя носителя для всего набора носителей резервных копий, не должно быть более 128 символов. Это имя должно совпадать с уже заданным именем носителя, который существует в томах резервных копий.

Если имя не указано и указан параметр SKIP, тогда проверки имени носителя не будет.

- MEDIAPASSWORD = { 'пароль' | @переменная _пароля} – Пароль на набор носителей. В следующей версии SQL Server данный параметр будет отсутствовать.
- BLOCKSIZE = {размер | @переменная _размера} – Размер физического блока в байтах, который может принимать одно из следующих значений: 512, 1024, 2048, 4096, 8192, 16384, 32768 и 65536 байт. Значение по умолчанию – 65536 для ленточных устройств и 512 для других.

На практике, данный параметр почти не используется, поскольку оператор BACKUP автоматически выбирает размер необходимого блока.

3. Параметры передачи данных:
   - BUFFERCOUNT = {к-dj_буферов | @переменная _к-во_буферов} – общее количество буферов ввода-вывода, которые будут использоваться при резервном копировании. Заметим, что при большом количестве буферов операция резервного копирования может быть очень ресурсоемкой.

   Общее пространство, которое используется буферами определяется по следующей формуле: кол-во_буферов * объем_пакета.
   - MAXTRANSFERSIZE = {объем_пакета | @переменная _объема_пакета} – наибольший объем пакета данных в байтах для передачи между SQL Server и носителем резервного набора. Поддерживаются значения, кратные 65 536 байт (64 Кб) и до 4194304 байт (4 Мб).

4. Параметры управления ошибками:
   - NO_CHECKSUM | CHECKSUM – разрешены (CHECKSUM) или нет (NO_CHECKSUM) контрольные суммы.

   По умолчанию контрольные суммы отсутствуют, за исключением сжатых резервных копий.
   - STOP_ON_ERROR| CONTINUE_AFTER_ERROR – будет ли операция резервного копирования остановлена после обнаружения ошибки в контрольной сумме страницы (STOP_ON_ERROR)

или продолжить работу (CONTINUE_AFTER_ERROR).

По умолчанию установлен STOP_ON_ERROR.

5. Параметры совместимости представлены одним параметром **RESTART**, который ничего не делает и оставлен только для обеспечения обратной совместимости.

6. В параметрах наблюдения существует также один представитель – это параметр **STATS** [= процентный_**интервал**]. Он позволяет выводить сообщения каждый раз по окончании указанного процентного интервала. По умолчанию сообщение выводится после каждых 10 процентов.

7. Параметры ленточных устройств:
   - REWIND | NOREWIND – Освободить и перемотать ленту (REWIND, значения по умолчанию) или же сохранить ее открытой после окончания резервного копирования (NOREWIND), что позволяет существенно улучшить производительность при выполнении нескольких операций резервного копирования на ленту.
   - Параметр NOREWIND включает в себя параметр NOUNLOAD, поэтому вместе их указывать в операторе BACKUP не допускается.
   - UNLOAD | NOUNLOAD – позволяют настроить текущий сеанс резервного копирования:
     - UNLOAD указывает на то, что лента должна автоматически перематываться и выгружаться по завершении операции backup.

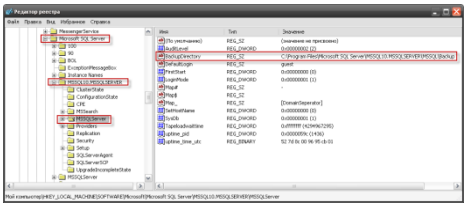
- NOUNLOAD указывает на то, что лента будет храниться в ленточном накопителе после операции backup.

ПРИМЕЧАНИЕ! В символьных параметрам не допускается указывать типы данных ntext и text. Кроме того, оператор BACKUP запрещено использовать в явных и неявных транзакциях.

Приведем пример создания полной резервной копии базы данных Press в файл на диске. При этом все резервные наборы, которые есть на диске, следует перезаписать:

```
backup database Press
to disk = 'D:\Backups\Press.bak'
with init;
```

Файл резервной копии – это обычный файл и поэтому достаточно просто прописать его название с полным путем. При создании резервной копии можно указывать **абсолютный** или **относительный** путь к файлу, а также вообще его **упустить.** В двух последних случаях, будет использоваться каталог по умолчанию, которым является "**C: \ Program Files \ Microsoft SQL Server \ MSSQL.n \ MSSQL \ Backup**", где n – номер экземпляра сервера. Каталог по умолчанию можно изменить в разделе реестра **BackupDirectory**, в ветке **HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Microsoft SQL Server \ MSSQL.n \ MSSQLServer.**



Кроме того, можно обращаться к удаленному дисковому файлу. Для доступа к общему сетевому ресурсу, необходимо, чтобы этот диск был подключен как сетевой. Чтобы указать сетевой ресурс при резервном копировании, необходимо использовать полное имя в формате UNC, которое имеет следующую форму: \\ **Имя_системы** \ **Общая_папка** \ **Путь** \ **имя_файла**. К примеру:

```
backup database Press
to disk = '\\BackupServer\Backups\Press.bak';
```

Что касается логических устройств резервного копирования, то они используются для неявного обращения к существующим физическим устройствам резервного копирования. Это позволяет упростить процесс указания файла резервной копии, поскольку отпадает необходимость прописывать долгий полный путь. Для их создания используется хранимая процедура **sp_addumpdevice**, которая позволяет связать физическое устройство резервного копирования с логическим.

```
sp_addumpdevice [ @devtype = ] 'тип_устройства_
резервного_копирования'      -- disk або tape
        , [ @logicalname = ] 'имя_логического_
устройства_резервного_копирования'
        , [ @physicalname = ] 'имя_физического_
устройства_резервного_копирования '
        -- устаревшие параметры, которые существуют для
совместимости      [, { [ @cntrltype = ] тип_контроллера | [
@devstatus = ] 'статус_устройства' } ]
```

Например, запишем резервную копию нашей базы данных Press на логическое устройство резервного копирования PressBack:

```
-- создаем логическое устройство резервного копирования
 exec sp_addumpdevice 'disk', 'PressBack', 'D:\
 Backups\Press.bak';

-- создаем резервную копию базы данных на логическое
 устройство резервного копирования
 backup database Press
 to PressBack;
```

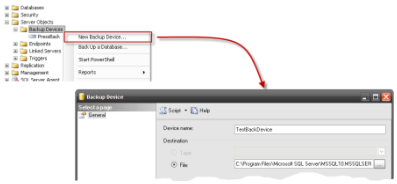
Для удаления существующего логического устройства резервного копирования используется системная хранимая процедура **sp_dropdevice:**

```
-- синтаксис
 sp_dropdevice [ @logicalname = ] 'имя_логического_
 устройства_резервного_копирования'
        [ , [ @delfile = ] 'delfile' ]  -- нужно удалять
- физическое устройство резервного копирования

- например, удалим одно из логических устройств резервного
 копирования,
- вместе с физическим устройством, которое ему
 соответствует
 exec sp_dropdevice 'PressBack', 'delfile'
```

Для просмотра имен существующих логических устройств, можно воспользоваться системным представлением **sys.backup_devices.**

Для создания устройства резервного копирования средствами SSMS необходимо выбрать в необходимом сервере, в папке Server Objects или ее подпапке **Backup Devices** пункт контекстного меню **New Backup Device ... (Новое устройство резервного копирования).**



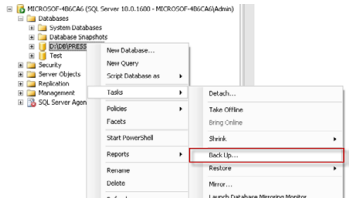
После этого в диалоговом окне Backup Device необходимо указать имя и путь к создаваемому ресурсу.

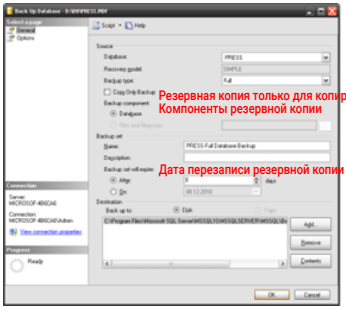
А теперь рассмотрим пример создания зеркальных наборов носителей (media set), которое позволяет создать несколько дополнительных копий данных. Существование нескольких резервных копий данных еще больше снижает риск потери данных в случае выхода из строя одного из устройств резервного копирования. При использовании зеркального резервного копирования SQL Server считывает страницу из файла данных один раз, после чего создает несколько копий при записи страницы на устройство резервного копирования (диск или ленту). Таким образом, одна страница данных записывается одновременно в каждое зеркало.

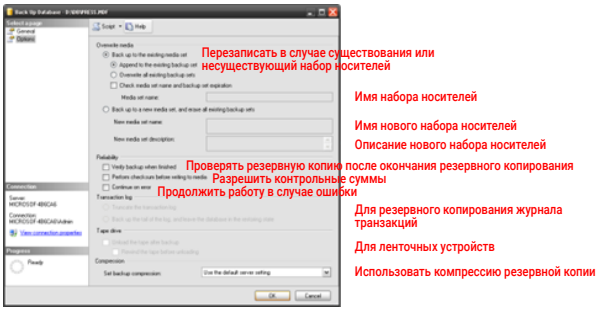
Итак, сохраним резервную копию базы данных Press на носителе, состоящую из двух дисков, а также создадим при этом два зеркала резервной копии. При этом резервная копия сохраняется на локальный диск, а зеркала на сетевые. С помощью опции FORMAT запишем также новые заголовки носителей на все наборы носителей, перезаписывая предыдущие заголовки и форматируя их содержание. Заметим, что без этой опции создание зеркальных наборов невозможно. Это обязательное условие при зеркальном резервном копировании.

```
BACKUP DATABASE Press
TO DISK = 'D:\Backups\Press1B.bak', DISK = 'D:\Backups\
Press2B.bak'
MIRROR TO DISK = '\\BackupServer1\Backups\PressMirror1A.bak',
          DISK = '\\BackupServer\Backups\PressMirror1B.bak'
MIRROR TO DISK = '\\BackupServer2\Backups\PressMirror2A.bak',
          DISK = '\\BackupServer\Backups\PressMirror2B.bak'
WITH
   FORMAT, MEDIANAME = 'PressSet'
```

Резервное копирование базы данных можно сделать и средствами SSMS. Для этого в контекстном меню необходимой базы данных необходимо выбрать пункт **Tasks-> Back Up ...** В новом диалоговом окне **Back Up Database** можно настроить почти все необходимые параметры для осуществления резервного копирования.







1.2.3. Резервное копирование журнала транзакций

Как уже было сказано, резервное копирование журнала транзакций (transaction log) позволяет сохранить активный журнал (active log). После этого сохраняются все последующие транзакции до тех пор, пока не будет обнаружена открытая транзакция.

Резервное копирование журнала транзакций можно осуществлять только для баз данных с моделью полного восстановления или модели восстановления с неполным протоколированием. Кроме того, создание резервной копии журнала транзакций возможно только после выполнения полного резервного копирования базы данных.

Для резервного копирования журнала транзакций используется инструкция **BACKUP LOG**:

```
BACKUP LOG { база_данных | @переменная_имени_БД }
TO устройство_резервного_копирования [ ,...n ] - см.
оператор BACKUP для резервного
                                                - копирование
базы данных [ MIRROR TO устройство_резервного_копирования
[ ,...n ]]
[ WITH { общие_опции_WITH | LOG_опций }  ]
```

Как видно из синтаксиса, инструкция BACKUP LOG почти ничем не отличается от инструкции BACKUP DATABASE для создания резервной копии базы данных. Основное отличие заключается в том, что кроме общих параметров резервного копирования, можно указывать еще ряд параметров, которые относятся только к операции резервного копирования журнала транзакций. Итак, к LOG параметров относят:

- NORECOVERY | STANDBY = имя_резервного_файла – создают резервную копию журнала транзакций и оставляют базу данных в состоянии RESTORING (NORECOVERY) или только для чтения и в состоянии STANDBY (STANDBY).

Для наиболее эффективного резервного копирования журналов, при котором не происходит его урезания и база данных автоматически переходит в состояние RESTORING, используйте одновременно параметр NORECOVERY с NO_TRUNCATE.

Использование параметра STANDBY равносильно параметру BACKUP LOG WITH NORECOVERY, за которым следует RESTORE WITH STANDBY.

При использовании режима ожидания необходимо указать резервный файл, в котором содержатся все изменения, которые прошли откат и должны быть восстановлены в случае необходимости.

- NO_TRUNCATE – журнал транзакций не урезается и резервное копирование осуществляется независимо от состояния базы данных. В связи с этим, резервная копия может иметь неполные метаданные. Как

правило, данный параметр используется при повреждении базы данных.

Параметр NO_TRUNCATE равносилен одновременному использованию COPY_ONLY и CONTINUE_AFTER_ERROR.

Стоит отметить, что резервные копии базы данных и журнала транзакций рекомендуется хранить на разных дисках. Это необходимо для обеспечения безопасности резервных копий в случае повреждения одного из дисков.

Приведем небольшой пример создания полной резервной копии базы данных и журнала транзакций. В качестве устройств резервного копирования используем логические устройства.

```
-- создаем логические устройства резервного копирования
- для сохранения резервной копии базы данных
 exec sp_addumpdevice 'disk', 'PressData', 'D:\
 Backups\PressData.bak';
 go
-- для сохранения резервной копии журнала транзакций
 exec sp_addumpdevice 'disk', 'PressLog', 'D:\Backups\
 PressLog.trn'; -- расширение произвольное
 go
-- делаем полную резервную копию базы данных
 backup database Press
 to PressData;
 go
-- делаем резервную копию журнала транзакций
 backup log Press
 to PressLog;
 go
```

1.2.4. Резервное копирование файловых групп и создания частичной резервной копии

В SQL Server 2008, кроме создания полной резервной копии базы данных, существует возможность резервного копирования отдельных файловых групп (filegroup backup). Такой подход очень полезен, если из-за размера базы данных ее полное сохранение и становится неоправданным.

Поскольку резервное копирование файловых групп сохраняет лишь фрагменты базы данных, для его выполнения необходимо, чтобы для базы данных была установлена модель полного восстановления или модель восстановления с неполным протоколированием. Кроме того, если необходимо восстановить файловые группы копий, сделанных в разное время, для их согласованости на определенный период времени, необходимо наличие журнала транзакций.

Для резервного копирования файловых групп базы данных используется следующая форма инструкции BACKUP DATABASE, которая практически ничем не отличается от создания резервной копии базы данных:

```
BACKUP DATABASE { ім'я_БД | @переменная_имени_БД }
{ FILE = { логическое_имя_файла | @ переменная _
логического_имени_файла }
   | FILEGROUP = { логическое_имя_файловой _группы | @
переменная_логического_имени_файловой_группы}
}  [ ,...n ]
TO устройство_резервного_копирования [ ,...n ] -- см.
оператор BACKUP для резервного
                                                  - Копирование
базы данных [ MIRROR TO устройство_резервного_копирования
```

```
[ ,...n ]]
[ WITH { DIFFERENTIAL | общие_опции_WITH [ ,...n ] } ]
```

Например, создадим дифференциальную резервную копию каждого файла, которые размещаются в двух вторичных файловых группах.

```
BACKUP DATABASE Press
   FILEGROUP = 'PressFilegroup1',
   FILEGROUP = 'PressFilegroup2'
TO DISK = 'D:\Backups\PressFilegroups.bak'
WITH DIFFERENTIAL
```

Но существуют случаи, когда в базе данных несколько файловых групп доступны только для чтения. В SQL Server 2008 такие файловые группы не будут входить в резервную копию базы данных, если при ее создании установить параметр **READ_WRITE_FILEGROUPS**. Но можно указать и исключение. Для этого после параметра READ_WRITE_FILEGROUPS необходимо перечислить необходимые файловые группы для чтения, которые планируется сохранить.

Если сама база данных доступна только для чтения, READ_WRITE_FILEGROUPS включает в резервную копию только первичную файловую группу.

Такая резервная копия будет частичной резервной копией базы данных, и синтаксис ее создания приведен ниже.

```
BACKUP DATABASE { имя_БД | @переменная_имени_БД }
READ_WRITE_FILEGROUPS
[, FILEGROUP = { логическое_имя_файловой_группы
          | @переменная _логического_имени_файловой_
группы [ ,...n ] ] -- файловая группа
```

```
- только для чтения
TO устройство_резервного_копирования [ ,...n ] -- см.
оператор BACKUP для резервного
                                                - копирование
базы данных
[ MIRROR TO устройство_резервного_копирования [ ,...n ]]
[ WITH { DIFFERENTIAL |  общие_опции_WITH [ ,...n ] }
]
```

Например:

```
BACKUP DATABASE Press
READ_WRITE_FILEGROUPS
TO DISK = 'D:\Backups\PressRW.bak'
```

1.3. Восстановление из резервной копии

Процесс восстановления базы данных происходит согласно выбранной вами стратегии восстановления, но в большинстве случаев она начинается с полного восстановления базы данных, которая восстанавливает базу данных на определенный период времени. Ведь полная резервная копия содержит все данные базы. После этого используются частичные (дифференциальные) последовательные резервные копии, которые позволяют привести ее в состояние на более поздний период.

Сам процесс восстановления из резервной копии длительный, ведь, если база данных восстанавливается с нуля, операция восстановления сначала создает для базы данных все файлы и файловые группы, после чего по порядку восстанавливаются все страницы.

Поэтому советуем по возможности перед восстановлением не удалять базу данных, а перезаписать ее.

Восстановление резервной копии выполняется с помощью оператора RESTORE, который позволяет осуществлять следующие **сценарии восстановления:**

- **Полное восстановление** – восстанавливает базу данных из полной резервной копии.
- **Частичное восстановление** – восстанавливает часть базы данных из дифференциальной резервной копии.
- **Восстановление файлов** – восстанавливает в базе данных указаные файлы или файловые группы.
- **Восстановление страниц** – восстанавливает указаные страницы базы данных.
- **Восстановление журнала транзакций** – восстанавливает журнал транзакций базы данных.
- **Восстановление по снимку** – позволяет вернуть базу данных к моменту создания снимке (snapshot) базы данных.

При этом, если восстановление резервных копий журнала транзакций не требуется, следует использовать простую модель восстановления.

Приведем синтаксис оператора RESTORE для перечисленных выше случаев использования

```
-- для полного восстановления базы данных
 RESTORE DATABASE { im'я_БД | @переменная_имени_БД }
 [ FROM устройство_резервного_копирования [ ,...n ] ] --
 см.оператор BACKUP для резервного
                                                    _
 копирование БД [ WITH
  {
       [ RECOVERY | NORECOVERY
     | STANDBY = {standby_file_name | @standby_file_
```

```
 name_var } ] -- залишає БД
-- в режиме восстановления, то есть разрешается
- осуществлять только выборку данных
 |, общие_опции_WITH [ ,...n ]
   |, KEEP_REPLICATION    -- опция_репликации_WITH
   |, KEEP_CDC          -- опции WITH отслеживания изменений
 данных  |, опции_WITH_service_broker
   |, опции_WITH_на_заданный_момент_времени
  } [ ,...n ]
 ]

-- для частичного восстановления базы данных
 RESTORE DATABASE { ім'я_БД | @переменная_имени_БД }
 {
    FILE = { логическое_имя_файла | @переменная_
 логического_имени_файла }
    | FILEGROUP = { логическое_имя _файловой_группы | @
 переменная_логического_имени_ файловой_группы }
    | READ_WRITE_FILEGROUPS
 } [ ,...n ]
 [ FROM устройство_резервного_копирования [ ,...n ] ]
 WITH
   PARTIAL,  -- восстановить первичную и указаны вторичные
 файловые группы
   NORECOVERY
    [, общие_опции_WITH | опции_WITH_на_заданный_момент_
 времени ] [ ,...n ]

-- для восстановления файлов и файловых групп
 RESTORE DATABASE { ім'я_БД | @переменная_имени_БД }
 {
    FILE = { логическое_имя_файла | @переменная_
 логического_имени_файла }
    | FILEGROUP = { логическое_имя_файловой_группы | @
 переменная_логического_имени_файловой _группы}
```

```
    | READ_WRITE_FILEGROUPS
} [ ,...n ]
[ FROM устройство_резервного_копирования [ ,...n ] ]
WITH { [ RECOVERY | NORECOVERY ]
       [, общие_опции_WITH [ ,...n ] ]
   } [ ,...n ]

-- для восстановления страниц базы данных
RESTORE DATABASE { имя_БД | @переменная_имени_БД }
PAGE = 'Id_файла:Id_страницы [ ,...n ]' -- список
страниц для восстановления
                                              - (Максимум
1000 страниц) [, { FILE = { логическое_имя_файла | @
переменная _логического_имени_файла }
    | FILEGROUP = { логическое _им_файловой_группы | @
переменная _ логического_имени_файловой_группы}
    | READ_WRITE_FILEGROUPS
    }
] [ ,...n ]
[ FROM устройство_резервного_копирования [ ,...n ] ]
WITH NORECOVERY
     [, общие_опции_WITH [ ,...n ] ]

-- для восстановления журналов транзакций
RESTORE LOG { имя_БД | @переменная _имени_БД }
[ { FILE = { логическое_имя_файла | @переменная _
логического_имени_файла }
    | FILEGROUP = { логическое_имя_файловой_группы | @
переменная _логического_имени_файловой_группы}
    | READ_WRITE_FILEGROUPS
    | PAGE = 'Id_файла:Id_страницы [ ,...n ]'
   } [ ,...n ]
]
[ FROM устройство_резервного_копирования [ ,...n ] ]
[ WITH
   {
     [ RECOVERY | NORECOVERY | STANDBY = {имя_standby_файла
                                          | @переменная _
```

```
 имени_standby_файла } ]
     |, общие_опции_WITH [ ,...n ]
     |, KEEP_REPLICATION  -- опция_репліикации_WITH
     |, опции_WITH_на_заданный_момент_времени
   } [ ,...n ]
]

-- вернуть базу данных на момент создания снимка (snapshot)
 RESTORE DATABASE { имя_БД | @переменная_имени_БД }
 FROM DATABASE_SNAPSHOT = имя_снимка_БД
```

Параметр **WITH** позволяет задать дополнительные необязательные параметры восстановления из резервной копии. Ряд общих параметров WITH совпадает с параметрами WITH для операции BACKUP (см. П.6.2.2. "Резервное копирование базы данных"). К ним относят:

- Параметры набора носителей (Media Set Options);
- Параметры передачи данных (Data Transfer Options);
- Параметры управления ошибками (Error Management Options);
- Параметры наблюдения (Monitoring Options);
- Параметры ленточных устройств (Tape Options).

В SQL Server 2008 параметр **PARTIAL** запускает начальный этап поэтапного восстановления, которое позволяет отложить восстановление файловых групп, которые остались, то есть не были указаны при восстановлении.

Еще несколько общих параметров WITH приведены ниже:

- MOVE "логическое_имя_в_backup";
- TO "имя_файла_с_путем" [... n] – указывает на то, что файл данных или журнал транзакций, который

указан после MOVE, необходимо восстановить из копии и переместить по адресу, заданной в параметре TO.

- REPLACE – если при восстановлении из резервной копии база данных (или журнал транзакций) с аналогичным именем уже существует, тогда необходимо ее перезаписать.

Если данный параметр не указан и при восстановлении из резервной копии на сервере уже существует база данных (или журнал транзакций) с аналогичным именем, тогда операция восстановления отменена.

- RESTART – перезапустить прерванную операцию восстановления с точки прерывания.
- RESTRICTED_USER – ограничить доступ к созданной базе данных для членов ролей db_owner, dbcreator и sysadmin. Параметр RESTRICTED_USER является альтернативой параметра DBO_ONLY ранних версий SQL Server.

Кроме того, данный параметр следует использовать с параметром RECOVERY.

Параметры резервного набора данных (Backup Set Options).

- FILE = {номер_резервного_набора | @переменная_номера_резервного_набора} – указывает на резервный набор данных для восстановления. Например, значение 1 указывает на то, что будет использоваться первый резервный набор данных на носителе данных резервных копий, значение 2 указывает на второй набор данных и т.д.

Перечень резервных наборов данных можно получить с помощью инструкции RESTORE HEADERONLY. По умолчанию принимается значение –1.

- PASSWORD = {'пароль' | @переменная_пароля} – Пароль на резервный набор данных. В следующей версии SQL Server эта опция будет удалена (!).

Если при создании резервной копии проводилась репликация базы данных, тогда при восстановлении следует указать параметр репликации **KEEP_REPLICATION**. Он предотвращает удаление параметров репликации, если резервная копия базы данных или журнала транзакций восстанавливается на сервере "горячего" резервирования и база данных воспроизводится. При этом, запрещается использовать данный параметр вместе с параметром NORECOVERY.

Если при создании резервной копии в базе данных была включена система отслеживания изменений данных, тогда при восстановлении следует указать ключевое слово **KEEP_CDC**. В таком случае SQL Server предотвращает удалению настроек системы отслеживания изменений данных. Данный параметр также запрещается использовать вместе с параметром NORECOVERY.

Если при создании резервной копии для базы данных был активирован компонент Service Broker, при восстановлении можно воспользоваться набором опций WITH для данного компонента.

- ENABLE_BROKER – После окончания процесса восстановления включить доставку сообщений компонента Service Broker. По умолчанию при

восстановлении доставка сообщений отключена. При этом существующий идентификатор компонента Service Broker сохраняется в базе данных.

- ERROR_BROKER_CONVERSATIONS – Завершает все диалоги, которые находятся в состоянии ошибки и были или присоединены к базе данных, или восстановлены. Это позволяет приложениям выполнять регулярную очистку существующих сеансов связи.

Доставка сообщений компонента Service Broker до завершения данной операции выключается, а затем снова включается. При этом существующий идентификатор компонента Service Broker сохраняется в базе данных.

- NEW_BROKER – назначает базе данных новый идентификатор компонента Service Broker. При этом все существующие сеансы связи в базе данных будут немедленно удалены, без выдачи диалоговых окон при завершении. В связи с этим, все маршруты, которые ссилаються на предыдущий идентификатор компонента Service Broker, нужно пересоздать.

Опции WITH на заданный момент времени позволяют восстановить базу данных на определенный момент времени или в определенной транзакции, указав при этом необходимую точку восстановления в инструкции STOPAT, STOPATMARK или STOPBEFOREMARK.

Для восстановления на определенный момент времени сначала нужно восстановить полную резервную копию базы данных, конечная точка которой находится ранее целевой точки восстановления. Чтобы упростить этот процесс, можно еще при создании резервной копии,

в операторе RESTORE DATABASE указать инструкцию WITH STOPAT, STOPATMARK или STOPBEFOREMARK.

Параметры WITH на заданный момент времени для инструкций RESTORE_DATABASE и RESTORE_LOG одинаковы, за исключением параметра STOPATMARK.

- STOPAT = { 'дата_и_время "
  | @переменная_даты_и_времени} – Восстановить базу данных или журнал транзакций до указанного периода времени. При этом к базе данных применяются только те записи журнала транзакций, которые были сделаны до указанной даты и времени.
  Тип данных переменной должен быть varchar, char, smalldatetime или datetime.
  (!) Если указанное время назначено после создания последней резервной копии журналов, база данных остается в невозобновленном состоянии. Это равносильно инструкции RESTORE LOG с параметром NORECOVERY.
  Для RESTORE DATABASE:
  STOPATMARK = { 'lsn: lsn_номер'}
  [AFTER "дата_и_время ']

  Для RESTORE LOG:
  STOPATMARK = { 'точка_транзакции “
  | "Lsn: lsn_номер '}
  [AFTER "дата_и_время '] – Возврат к указанной точке восстановления. Указанная транзакция включается в восстановление, но фиксируется только в случае, если она была изначально зафиксирована в ходе формирования транзакции.

Параметр lsn_номер определяет регистрационный номер транзакции в журнале.

Если в инструкции RESTORE LOG не задан параметр AFTER, тогда восстановление происходит до первой точки с указанным именем. Иначе восстановление происходит до первой точки с указанным именем перед или после указанной даты и времени.

(!) Если указана точка сохранения транзакции, номер LSN или время назначены после создания последней резервной копии журналов, база данных остается в невозобновленных состоянии. Это равносильно инструкции RESTORE LOG с параметром NORECOVERY.

Для RESTORE DATABASE:

STOPBEFOREMARK =
{ ‹Lsn: lsn_номер›}
[AFTER "дата_и_время ']

Для RESTORE LOG:

STOPBEFOREMARK =
{ ‹Точка_транзакции “
| "Lsn: lsn_номер '}
[AFTER "дата_и_ время ']

Возврат к указанной точке восстановления. Заданная транзакция не включается в восстановление, так как после использования параметра WITH RECOVERY происходит ее откат.

Параметр lsn_номер определяет регистрационный номер транзакции в журнале.

Если в инструкции RESTORE LOG не задан параметр AFTER, тогда восстановление происходит до первой точки с указанным именем. Иначе восстановление происходит до первой точки с указанным именем перед или после указанной даты и времени.

- Восстановление на заданный момент времени не будет выполняться, если последовательность частичного восстановления включает файловые группы FILESTREAM. Но существует возможность принудительно продолжить последовательность восстановления. Для этого необходимо указать параметр CONTINUE_AFTER_ERROR вместе с параметром STOPAT, STOPATMARK или STOPBEFOREMARK. Это позволит выполнить последовательность частичного восстановления, но файловая группа FILESTREAM при этом не восстановится.

В SQL Server 2008 существует возможность вернуть базу данных к сделанному снимку базы данных (snapshot). Для этого необходимо задать имя необходимого снимка в параметре **DATABASE_SNAPSHOT** при восстановлении из резервной копии. Но, следует помнить:

- Аргумент DATABASE_SNAPSHOT доступен только для полного восстановления базы данных;
- При таком восстановлении будут удалены все полнотекстовые каталоги;
- Максимальный размер снимка базы данных равен размеру самой базы данных;
- Расширение файла снимка базы данных может использоваться произвольное.

Приведем несколько примеров восстановления.

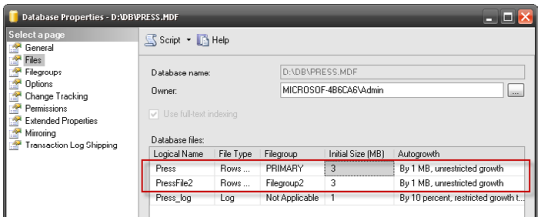
1. Восстановление базы данных с полной резервной копии:

```
restore database Press
from disk = 'D:\Backups\Press.bak'

-- если после восстановления ее нужно переместить, тогда
 используем следующий запрос:
restore database Press
from disk = 'D:\Backups\Press.bak'
with
move 'Press_Data'
to 'C:\Program Files\Microsoft SQL Server\MSSQL10.
MSSQLSERVER\MSSQL\Data\Press.mdf',
move 'Press_Log'
to 'C:\Program Files\Microsoft SQL Server\MSSQL10.
MSSQLSERVER\MSSQL\Data\Press.ldf'
```

2. Создадим снимок базы данных Press с именем Press_sn, после чего попытаемся осуществить полное восстановление базы данных со сделанного снимка.

Напомним, что наша база данных теперь состоит из двух файлов: Press и PressFile2. В соответствии с синтаксисом, при создании снимка необходимо указать каждый файл первичной базы данных. Файловые группы не указываются.




```
-- синтаксис для создания моментального снимка базы данных
CREATE DATABASE имя_снимка_БД
ON
      -- список файлов в основной базе данных
  ( NAME = логическое_имя_файла,
    FILENAME = 'месторасположение_файла'
  ) [ ,...n ]
AS SNAPSHOT OF имя_основной_БД

-- создаем снимок базы данных
create database Press_sn
on
( name = Press, filename = 'C:\Program Files\Microsoft
SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Data\Pressdat1_
sn.ss'),
( name = PressFile2, filename = 'C:\Program Files\
Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Data\
Pressdat2_sn.ss')
as snapshot of Press;
go

-- восстанавливаем базу данных со снимка
restore database Press
from database_snapshot = 'Press_sn';
```

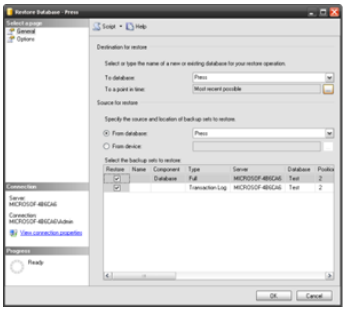
3. Восстановим журнал транзакций до определенной метки, то есть точки сохранения в транзакции PointRestore.

```
-- отмечаем транзакцию, в которой будет восстановлен журнал
транзакций
begin transaction PointRestore
with mark 'Deleting old books'; -- указывает на то,
что транзакция отменена в журнале
go
delete
from sale.Sales
where ID_BOOK in
     (select ID_BOOK
     from book.Books
     where DATEPART(YEAR, DateOfPublish) < 2000)
go
delete
from book.Books
where DATEPART(YEAR, DateOfPublish) < 2000
go
commit transaction PointRestore;
go

-- осуществляем резервное копирование базы данных и журнала
транзакций
use master;
go
backup database Press
to disk = 'D:\Backups\Press.bak'
with init;
go
backup log Press
to disk = 'D:\Backups\PressLog.trn';
go
-- восстанавливаем базу данных и журнал транзакций
restore database Press
from disk = 'D:\Backups\Press.bak'
with norecovery;
go
restore log Press
from disk = 'D:\Backups\Press.bak'
```

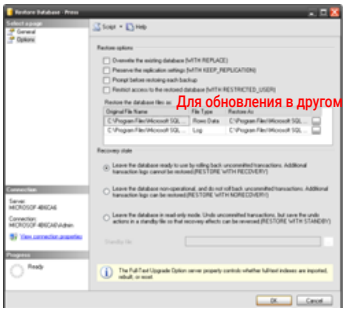
```
with recovery,
     stopatmark = 'PointRestore';
go
```

Средствами SSMS операция восстановления очень схожа с операцией резервного копирования. То есть нужно перейти к базе данных, которую необходимо восстановить, и в ее контекстном меню выбрать пункт **Tasks-> Restore,** после чего указать, что именно требует восстановления база данных, журнал транзакций, файлы или файловые группы. Например, для базы данных, окно Restore будет иметь следующий вид:



Параметры сохранения обновленной копии

Параметры резервной копии, которую необходимо обновить



Опции для обновления

Для обновления в другом месте, то есть где именно обновить

В каком состоянии должна находится БД после обновления

1.4. Проверка резервной копии

Уметь создать резервную копию, к сожалению, мало. Ведь, резервная копия может повредиться или еще в процессе создания или со временем. В таком случае она будет не пригодна для работы. Итак, вы должны также знать, как проверить работоспособность резервной копии. С одной стороны, здесь ничего сложного нет: достаточно просто ее восстановить и проверить все данные. Но этот процесс может быть длительным, а, следовательно, непрактичным. SQL Server 2008 имеет простой способ решения данной проблемы – это инструкция **RESTORE VERIFYONLY**, которая имеет следующий синтаксис:

```
RESTORE VERIFYONLY
FROM устройство_резервного_копирования [ ,...n ] --
см.оператор BACKUP для резервного
                    - копирования БД [ WITH
  {
   LOADHISTORY -- загружать данные в журнальные таблицы
 резервного копирования и
- восстановление БД msdb
- параметры операции восстановления     | MOVE
'логическое_имя_backup_файла' TO 'месторасположение' [
,...n ]
      -- параметры резервного набора данных
   | FILE = { номер_резервного_набора |@переменная_
 номера_резервного_набора }
   | PASSWORD = { пароль | @переменная_пароля }

   -- параметры набора носителей
   | MEDIANAME = { 'имя_носителя' | @переменная _имени }
   | MEDIAPASSWORD = { 'пароль' | @ переменная_пароля }

   -- параметры управления ошибками
```

```
  | { CHECKSUM | NO_CHECKSUM }
  | { STOP_ON_ERROR | CONTINUE_AFTER_ERROR }
  -- параметры наблюдения
  | STATS [ = процентный_интервал ]

  -- параметры ленточных устройств
  | { REWIND | NOREWIND }
  | { UNLOAD | NOUNLOAD }
} [ ,...n ]
]
```

Чтобы убедиться в том, что носитель не поврежден, данная инструкция проверяет сначала заголовок, потом контрольную сумму резервной копии. После этого считывается внутренняя последовательность страниц и вычисляется контрольная сумма резервной копии для сравнения. Также проводятся различные тесты для проверки целостности резервной копии, но структура ее данных при этом не затрагивается.