**Министерство Образования Российской Федерации**

Название Вашего института (полное)

Название Вашей кафедры (полное)

# Курсовая работа

по специальности ""

на тему:

**"Восстановление базы данных"**

Москва, 2010

**Содержание**

Введение

1. Причины повреждений баз данных

2. Восстановление баз данных на примере SQL Server 2005

2.1 Основные возможности восстановления баз данных SQL Server 2005

2.2 Подготовка к восстановлению

2.3 Проведение восстановления

2.4 Специальные ситуации восстановления

Заключение

Список литературы

# Введение

Установлено, что большинство предприятий, переживших крупную необратимую потерю корпоративных данных, прекращают свое существование в течение трех лет после такого инцидента. Мысль о возможной катастрофе неприятна для ИТ- и бизнес-менеджеров, поэтому они часто не принимают серьезных предупредительных мер. К сожалению, это грубая правда жизни. И здесь под "предупредительными мерами" ни в коем случае нельзя понимать "покупку техники brand-name", так как "брэнды" тоже ломаются, иногда даже чаще чем "самосборные машины". Поэтому "предупредительные меры" - это не только качественное "железо", но и планирование резервного копирования данных.

Актуальность моего исследования определила цель и задачи работы:

Цель исследования – рассмотреть методы восстановления баз данных

Для достижения цели необходимо решить следующие задачи:

1. На основе анализа зарубежной и отечественной литературы, монографических источников изучить методы восстановления баз данных
2. Выявить причины при которых базу данных нужно восстанавливать
3. Провести анализ основных возможностей восстановления
4. Рассмотреть на примере SQL Server 2005 восстановление базы данных
5. На основе проведенного исследования сделать выводы и дать рекомендации по работе

Для раскрытия поставленной цели и задач определена следующая структура исследования: работа состоит из введения, двух глав, заключения и списка использованной литературы. Названия глав отображают их содержание.

# 1. Причины повреждений баз данных

Существует несколько причин, при которых база данных может оказаться поврежденной. Перечислим наиболее характерные.

1. *Отключение питания сервера.*

Самый частый случай повреждения базы данных это отключение питания на сервере. Такие ситуации нужно пытаться предотвращать, используя аппаратные средства (UPS, RAID-контроллеры с батарейками).

Существует два режима записи страниц - синхронный и асинхронный. Синхронная запись означает, что измененные страницы базы данных не будут кэшироваться операционной системой, а будут записываться непосредственно на диск при выталкивании страниц из кэша на запись (на Windows это в буквальном смысле отсутствие флага lazy write при открытии файла БД). Это ухудшает производительность, поэтому большинство людей выключают forced writes. В этом случае измененные страницы находятся в кэше операционной системы до тех пор, пока операционная система не решит записать их на диск.

В некоторых случаях при непрерывной работе с БД операционная система не сбрасывает измененные страницы на диск до тех пор, пока все пользователи не отсоединятся от базы данных. При выключении питания в этом случае повреждения базы данных могут быть максимальными. Причем, повреждения в данном случае происходят от "недозаписи" информации. Это куда менее печальный случай, чем "перезапись" информации случайными данными. Однако, на Windows было обнаружено, что если у базы данных установлено Forced Write = Off, то при определенных условиях измененные страницы БД могли неделями не попадать в БД, и оставаться в кэше операционной системы. При этом, в случае сбоя сервера (или отключения питания), пропадало огромное количество изменений в БД (а база могла выглядеть вообще неповрежденной). То есть, БД как бы оказывалась "неизменяемой" в течение длительного времени.

1. *Дефекты оборудования*

*Память.* Самый частый дефект - сбои памяти (RAM). Очевидно, при использовании серверов "своей сборки" приобретается память подешевле, что приводит к соответствующим результатам. Желательно для сервера приобретать и материнскую плату и память с поддержкой ECC. Сбои памяти могут привести к достаточно тяжелым последствиям. Сервер не только "пропускает" страницы базы данных через память, но и кэширует их в памяти. Контрольные суммы страниц, даже если бы они и были, не помогут когда сервер будет записывать страницу на диск через сбойный участок памяти. В противном случае данные пришлось бы перечитывать, что весьма серьезно ухудшило бы производительность. Сбои памяти еще плохи тем, что в этом случае поврежденными как правило оказываются и база данных и ее копия, если копия используется в качестве "быстрой резервной копии" (т.к. запись на диск идет из одних и тех же участков памяти).

*Диски.* Раньше, лет 10-15 назад, bad-блоки появлялись часто, и существовали специальные утилиты для их исправления. Сейчас контроль ошибок не только может исправить данные на поврежденном блоке самостоятельно, но и прозрачно сохранит блок в рабочем месте диска, а плохой блок пометит к исключению из дальнейшего использования. Грубо говоря, нынешние диски либо работают, либо не работают целиком.

*Контроллеры.* Здесь можно отметить только то, что при сбое в работе контроллера некорректная информация может быть записана на все носители, которые подключены к этому контроллеру. Именно поэтому при организации создания копии рекомендуется располагать ее на винчестере, подключенном к другому контроллеру.

*Другие программы.* Приложения в основном не работают с операционной системой на "внутреннем" уровне. Такие приложения никогда не смогут вызвать сбой типа известного "синего экрана" в Windows. Поэтому если подобный сбой ОС произошел, в этом скорее виноваты некорректно работающие драйверы или само оборудование (очень часто в "синем экране" виноваты драйвера видео).

1. *Сбои самого сервера*

Разумеется, серверные программы, как и любое другое ПО, не являются идеальной программой. Идеальной, конечно, в смысле отсутствия ошибок. Если "железо" работает нормально, сервер может сам "сломаться" и либо просто перестать работать, либо испортить базу данных.

Раньше код сервера содержал вызов обычной функции позиционирования по файлу БД (seek), которая не могла адресовать более 4-гигабайт (в те далекие времена просто не было файловых систем, которые поддерживали файлы больше 4-х гигабайт). Когда в функцию передавалось такое большое число, оно обрезалось по старшим разрядам. Происходила такая ситуация при операции расширения файла БД, т.е. при записи новых страниц, а следовательно файл БД оказывался "затертым" новой информации с самого начала, т.е. с нулевой страницы (страница заголовка БД). Если новых страниц к записи было много, то уничтожалась начальная часть БД, где как правило содержатся системные таблицы, страницы информации о транзакциях и т.п. Причем борьба с пресловутым размером файла в 4 гигабайта дольше всего велась на Linux, что связано не только с кодом СУБД, но и с поддержкой файлов таких размеров самой операционной системой и ее файловыми системами. На Windows, Firebird и Yaffil этой проблемы уже нет, т.е. файл БД может иметь размер и 10, и 20 и больше гигабайт.

В любом случае, при работе на Unix или Windows следует внимательно изучить возможности ядра и конкретной (используемой) файловой системы - способны ли они работать с файлами больше 4-х гигабайт, а также проверить версию IB/FB/YA, чтобы быть уверенным в корректной работе с такими файлами, или наоборот, сразу предусмотреть разбиение БД на многофайловую, например "кусками" по 2-3 гигабайта.

В отношении файловых систем Windows известна следующая особенность: на FAT32 поддерживаются файлы размером не более 4 гигабайт (чаще всего указанное повреждение БД и происходит при использовании этой, фактически уже устаревшей, файловой системы). Допустим, размер вашей БД достиг 3-х гигабайт, и вы хотите перенести ее на NTFS, чтобы избежать ограничения в 4 Гб. Проблема в том, что с FAT32 на NTFS скопируется только 2 гигабайта из 3-х, из-за особенностей Windows. Это еще раз подчеркивает необходимость знания ограничений используемых файловых систем и их взаимодействия на одном компьютере.

1. *Остановка во время сборки мусора*

Если во время принудительного завершения работы сервера были активные подключения и сервер занимался сборкой мусора, то база данных может быть повреждена (и чаще всего это так и происходит). Уменьшить вероятность таких повреждений можно только отключив автоматическую сборку мусора, а в случае принудительной сборки мусора предварительно делать "быстрый" backup, чтобы резервная копия базы данных оказалась самой свежей в случае сбоя и повреждения БД.

1. *Повреждения индексов*

Повреждения индексов могут происходить как по всем вышеперечисленным причинам, так и из-за ряда багов сервера при работе с индексами. Поскольку индексы не являются 100% необходимым видом объектов для функционирования базы данных, их повреждения обнаруживаются значительно позже, чем повреждения других объектов БД (например, данных таблиц). И клиентские приложения могут продолжать функционировать в такой ситуации как и прежде.

Однако, при повреждении индексов возможно искажение данных, получаемых приложениями. Если в индексе повреждено несколько ключей, и сервер не выдал сообщения об ошибке при выполнении запроса, использующего такой индекс, в результат запроса не попадут записи, на которые ссылаются те самые поврежденные ключи. То есть, часть записей может "пропасть". Обнаружить разницу в выдаваемом количестве записей можно только используя запросы с полным перебором записей

SELECT \* FROM TABLE

и с перебором по индексу

SELECT \* FROM TABLE

WHERE FIELD > 0

где FIELD - столбец, по которому есть возможно поврежденный индекс, а > 0 - условие, которое однозначно будет выбирать все записи.

Можно этого и не делать, а при подозрении на "пропадание записей" сразу посмотреть в log-файл, и перестроить те индексы, о повреждениях которых там сообщается.

В log-файл пишутся только порядковые номера индексов (а не их имена) для конкретных таблиц. Процесс backup поврежденные или неповрежденные индексы (за исключением повреждений индексов по системным таблицам) не интересуют, т.к. индексы в backup хранятся только в виде описания в системных таблицах (restore создает индексы по этим описаниям в самом конце процесса restore). Backup считывает записи в натуральном порядке и индексы не использует, поэтому все существующие (committed) записи обязательно попадут в backup. Однако, если поврежден уникальный индекс, то в определенных условиях существует вероятность повторной вставки записи в таблицу с уже существующим (уникальным) значением столбца. Эта ситуация может привести к невосстановимому backup, т.е. процесс restore остановится в момент создания уникального индекса, обнаружив дубликат уникального значения в восстановленных записях. Но такая проблема также не является катастрофической - процесс создания индексов выполняется самым последним, т.е. после того как абсолютно все объекты БД уже восстановлены в базе данных процессом restore. Если вдруг обнаружена проблема неуникальных данных при создании индекса, можно попробовать найти такую запись (и затем удалить лишнюю) запросом

SELECT ID, COUNT(\*) FROM TABLE

GROUP BY ID

HAVING (COUNT(\*)) > 1

где id - столбец, по которому есть не создаваемый уникальный индекс. После этого можно активировать индексы, которые не были восстановлены.

1. *Повреждения таблиц*

Нормальная база данных - это не набор отдельных таблиц. Таблицы между собой могут быть достаточно сильно взаимосвязаны, вплоть до циклических ссылок. Поэтому даже один и тот же тип и объем повреждения может иметь разные последствия, в зависимости от того, с какой таблицей это произошло. Типичный пример: таблица CLIENTS - справочная, а ORDERS - оперативная. Если пропадет часть записей из ORDERS, то после починки БД будет нормально функционировать. Если же будет повреждена CLIENTS, то после починки в ORDERS будут записи, ссылающиеся на несуществующих клиентов. Таким образом БД вроде бы будет отремонтирована, но логическая целостность данных будет нарушена. Бороться с этой ситуацией никак невозможно, разве что чаще делая backup (поскольку справочники меняются реже, чем оперативные данные). Подобная ситуация (с повреждением master-таблицы) может возникнуть даже в том случае, если все записи пакета master-detail вставляются в одной транзакции, а Forced Write выключен (off) - страницы с записями detail могут быть записаны на диск операционной системой из кэша раньше, чем соответствующие им записи таблицы master. Это не приводит к "невосстановимому backup", но после restore придется или добавлять недостающие master-записи, или удалять "осиротевшие" detail-записи (в зависимости от сложности триггеров, обрабатывающих вставку в master или удаление в detail. Возможно, такие триггеры на время ремонта данных нужно будет отключить).

Также, в подобной ситуации, при restore отремонтированной базы данных могут оказаться неактивными индексы по Foreign Key соответствующих связей master-detail. Активировать их можно после упомянутых вставок или удалений master/detail, путем установки rdb$indices.rdb$index\_inactive в 0.

*7. Стихийные и техногенные бедствия*

Шторм, землетрясение, воры, пожар, прорыв водопровода — всё это приводит к потере всех носителей данных, расположенных на определённой территории. Данная причина потери данных бывает очень редко, но она имеет место.

Единственный способ защиты от стихийных бедствий — держать часть резервных копий в другом помещении.

*8. Вредоносные программы*

В эту категорию входит случайно занесённое ПО, которое намеренно портит информацию — (вирусы, черви, «троянские кони»). Иногда факт заражения обнаруживается, когда немалая часть информации искажена или уничтожена.

Решением этой проблемы будет:

* установка антивирусных программ на рабочие станции. Простейшие антивирусные меры — отключение автозагрузки, изоляция локальной сети от Интернета, и т. д.;
* обеспечение централизованного обновления: первая копия антивируса получает обновления прямо из Интернета, а другие копии настроены на папку, куда первая загружает обновления; также можно настроить прокси-сервер таким образом, чтобы обновления кешировались (это всё меры для уменьшения трафика);
* иметь копии в таком месте, до которого вирус не доберётся — выделенный сервер или съёмные носители;

Если копирование идёт на сервер: обеспечить защиту сервера от вирусов (либо установить антивирус, либо использовать ОС, для которой вероятность заражения мала). Хранить версии достаточной давности, чтобы существовала копия, не контактировавшая с заражённым компьютером.

Если копирование идёт на съёмные носители: часть носителей хранить (без дописывания на них) достаточно долго, чтобы существовала копия, не контактировавшая с заражённым компьютером.

*9. Человеческий фактор*

Намеренное или ненамеренное уничтожение важной информации — человеком, специально написанной вредоносной программой или сбойным ПО.

Тщательно расставляются права на все ресурсы, чтобы другие пользователи не могли модифицировать чужие файлы. Исключение делается для системного администратора, который должен обладать всеми правами на всё, чтобы быть способным исправить ошибки пользователей, программ и т.д.

Обеспечить работающую систему резервного копирования — то есть, систему, которой люди реально пользуются и которая достаточно устойчива к ошибкам оператора. Если пользователь не пользуется системой резервного копирования, вся ответственность за сохранность ложится на него.

Хранить версии достаточной давности, чтобы при обнаружении испорченных данных файл можно было восстановить.

Перед переустановкой ОС следует обязательно копировать всё содержимое раздела, на которой будет установлена ОС, на сервер, на другой раздел или на CD / DVD.

Оперативно обновлять ПО, которое заподозрено в потере данных.

# 2. Восстановление баз данных на примере SQL Server 2005

##

## 2.1 Основные возможности восстановления баз данных SQL Server 2005

Прежде чем рассматривать процедуру восстановления баз данных SQL Server, уточним значение нескольких терминов используемых в SQL Server.

*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. Нередко ею пользуются для создания тестовой базы для проверки критических обновлений или учебы. Во время этого процесса производится перенос данных из резервной копии на сервер базы данных.

*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 был завершена некорректно (например, пропало питание), то эта процедура вернет все базы данных в работоспособное состояние.

*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.

*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.

##

## 2.2 Подготовка к восстановлению

Перед восстановлением 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 Datаbases 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 время у вас есть, то лучше, конечно, подстраховаться, 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.

##

## 2.3 Проведение восстановления

После того, как подготовка завершена, можно приступать 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:

1. 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;

2. 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, которые 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 флажку 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:

1. 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;
2. WITH NORECOVERY — после окончания процесса восстановления с носителя процедура RECOVERY не начнется. Базы данных останется 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;
3. 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, в предыдущих версиях ее не было.

*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 каждых 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.

## 2.4 Специальные ситуации восстановления

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 страничном 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';

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.

Порядок действий восстановления отдельных страниц базы данных 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.

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 перестроения 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 обнаружится 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 перестроение базы данных master, то после завершения восстановления этой базы данных обязательно нужно произвести восстановление 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.

# Заключение

Процесс восстановления данных - важнейшая операции в SQL Server. Восстановить данные не так сложно, как представлялось раньше, особенно с помощью графического интерфейса утилиты Enterprise Manager. Затруднение вызывает только восстановление поврежденной базы данных master.

Делая резервные копии и тестируя каждую создаваемую резервную копию и восстановленную базу данных, мы избегаем частых потерь данных и сбоев.

Резервное копирование необходимо для возможности быстрого и недорогого восстановления информации в случае поломки рабочей копии базы по какой-либо причине.

В заключении своей работы хочется привести первый закон Чизхолма, который доказывает актуальность данной работы:

*Все, что может испортиться, портится.*

*Все, что не может испортиться, портится тоже.*

# Список литературы

1. Ковязин А.Н., Востриков С.М. "Мир Interbase", М.: Кудиц-Образ, 2002г.
2. Крис Касперски «Восстановление данных. Практическое руководство» Спб.: БХВ-Петербург, 2007г.
3. Леонов Василий. «Восстановление данных». М.: Эксмо, 2009 г.
4. Михеев Ростислав «MS SQL Server 2005 для администраторов». Спб.: БХВ-Петербург, 2007г.
5. Уильям Р. Станек «Microsoft SQL Server 2005. Справочник администратора». М.: Русская Редакция, 2008 г.