Репликация, дублирование и восстановление.

- **Репликация** - это процесс, посредством которого данные копируются между базами данных, находящимися на том же самом сервере или на других серверах, связанных через LAN, WAN или Internet
- **Репликация** Microsoft SQL Server использует метафоры:

  publisher

  distributor

  subscriber.

- Publisher - сервер или база данных, которая посылает данные на другой сервер или в другую базу данных.

- Subscriber - сервер или база данных, которая получает данные от другого сервера или другой базы данных.

- Distributor - сервер, который управляет потоком данных через систему **репликации**. Этот сервер содержит специализированную базу данных: Distribution database.

- Publisher содержит публикацию/публикации. Публикация - это совокупность одной или более статей, которые посылаются серверу подписчику (subscriber) или базе данных.

- Статья (Article) - основной модуль **репликации** и это может быть таблица или подмножество таблицы.

- Подписка (subscriptions) - это группа данных, которые сервер или база данных получает.

- Существуют виды подписки:

push и pull subscriptions

- Push subscription - это подписка, при которой сервер издатель периодически помещает транзакции на подписавшиеся сервера или базы данных.
- Pull subscription - это подписка, при которой подписавшийся сервер будет периодически соединяться с тиражируемой информацией и перемещать её из Distribution database.

- Distribution database - это системная база данных, которая хранится на дистрибуторе (distributor) и не содержит никаких пользовательских таблиц. Эта база данных используется для хранения снимков заданий и всех транзакций, ожидающих распределения подписчикам.

Топология репликации

Microsoft SQL Server поддерживает следующие топологии репликации

- Центральный publisher

- Центральный subscriber

- Центральный publisher с отдаленным distributor

- Центральный distributor

- Издающий subscriber

Центральный publisher

Это одна из наиболее используемых топологий **репликации**. В этом сценарии, один сервер исполняет роли publisher и distributor, а другой сервер/серверы определяется, как подписчик/подписчики.

Центральный subscriber

Это обычная топология складирования данных. Несколько серверов или баз данных копируют свои данные на центральный сервер в одну или более базы данных

Центральный publisher с отдаленным distributor

В этой топологии база Distribution постоянно находится на сервере, отличном от сервера, где располагается publisher. Эта топология используется для повышения эффективности, когда объём **репликации** увеличивается, а также, если сервер или сетевые ресурсы ограничены. Это уменьшает загрузку publisher, но увеличивает сетевой трафик. Эта топология требует отдельных инсталляций Microsoft SQL Server для publisher и для distributor.

Центральный distributor

- В этой топологии, несколько издателей используют только один distributor, который постоянно находится на отличном от издателей сервере. Это одна из наиболее редко используемой топологии **репликации**, потому что имеет уязвимую точку (на сервере с центральным distributor), и если сервер distributor потерпит неудачу, сценарий **репликации** будет разрушен полностью.

Издающий subscriber

Это топология двойственной роли. В ней, два сервера издают те же самые данные. Сервер издатель посылает данные на subscriber, и затем subscriber издает данные на любое число подписчиков. Это полезно когда publisher должен послать данные подписчикам по медленной или дорогой линии связи.

Типы репликации

Microsoft SQL Server 7.0/2000 поддерживает следующие виды **репликации:**

- Snapshot
- Transactional
- Merge

Snapshot репликация (снимок)

Является самой простой. При этом, все копируемые данные (точная копия) будут копироваться из базы данных publisher в базу(ы) данных subscriber/subscribers на периодической основе. Snapshot **репликация** является лучшим методом копирования данных, которые нечасто изменяются и когда размер копируемых данных не очень большой.

Transactional репликация

SQL Server фиксирует (делает моментальные снимки) все изменения, которые были сделаны в статье, и сохраняет, как: INSERT, UPDATE и DELETE инструкции в базе Distribution. Эти изменения посылаются подписчикам от Distribution и применяются к расположенным в них данным.

Transactional **репликации** лучше использовать, когда копируемые данные часто изменяются или когда размер копируемых данных достаточно велик и нет необходимости поддерживать автономные изменения реплицируемых данных относительно publisher и относительно subscriber.

Merge репликация

Является наиболее трудным типом **репликации**. Она предоставляет возможность автономных изменений реплицируемых данных и на publisher и на subscriber. При Merge **репликации**, SQL Server фиксирует все накопившиеся изменения не только в источнике данных, но и целевых базах данных, и урегулирует конфликты согласно правилам, которые Вы предварительно конфигурируете, или посредством определённого Вами блока принятия решений - resolver-ра.

Merge **репликацию** лучше использовать, когда Вы хотите обеспечить поддержку автономных изменений реплицируемых данных относительно publisher и относительно subscriber.

Агенты Репликации

Microsoft SQL Server 7.0/2000 поддерживает следующих агентов **репликации:**

- Snapshot Agent
- Log Reader Agent
- Distribution Agent
- Merge Agent

Snapshot Agent

Агент **репликации**, который создаёт файлы снимков, хранит снимки на distributor и производит запись информации о состоянии синхронизации в Distribution database. Snapshot Agent используется во всех типах **репликации** (Snapshot, Transactional и Merge) и может управляться из SQL Server Enterprise Manager.

Log Reader Agent

Агент **репликации**, который перемещает транзакции, отмеченные для **репликации** из transaction log, находящегося на publisher, в Distribution database. Этот агент **репликации** не используется в Snapshot **репликации**.

Distribution Agent

Агент **репликации**, который перемещает обрабатывающие снимки задания из Distribution database к подписчикам и перемещает все транзакции, ожидающие распределения на подписчиков. Distribution Agent используется в Snapshot и Transactional **репликациях** и может управляться с помощью SQL Server Enterprise Manager.

Merge Agent

Агент **репликации**, который применяет первоначальные, обрабатывающие снимки задания по таблицам базы данных publication на подписчиках, и потом объединяет возможные последующие изменения данных, которые произошли после создания первоначального снимка. Merge Agent используется только в Merge **репликации**

Резервное копирование

MS SQL поддерживает 3 типа backup'а данных

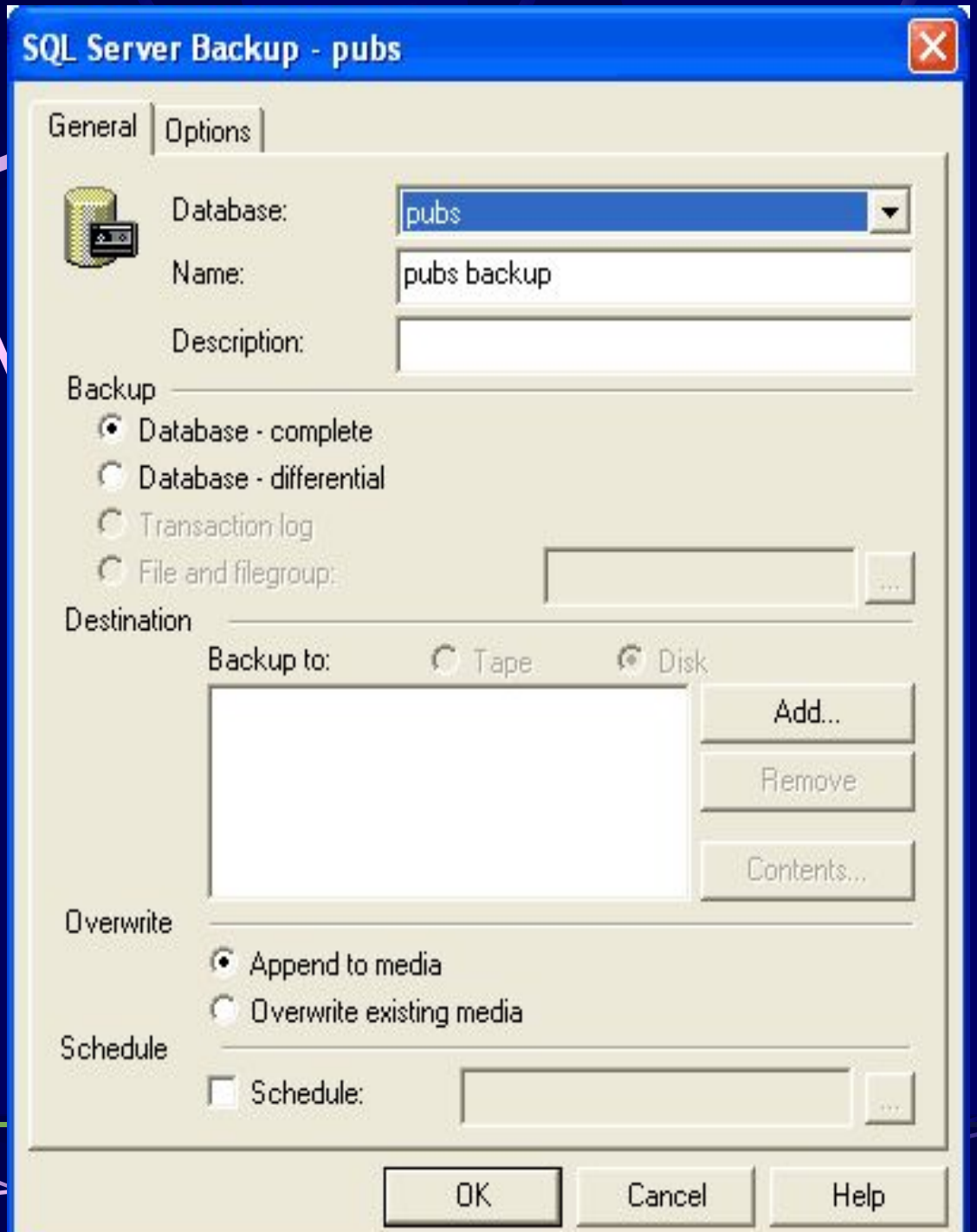
- Full backup
- Differential backup
- Transaction-log backup

Full backup

Сохраняет все объекты вашей базы, включая пользователей и permissions. Full backup может производиться без остановки работы сервера, все транзакции произведенные за время выполнения backup’а добавляются к нему по окончании.

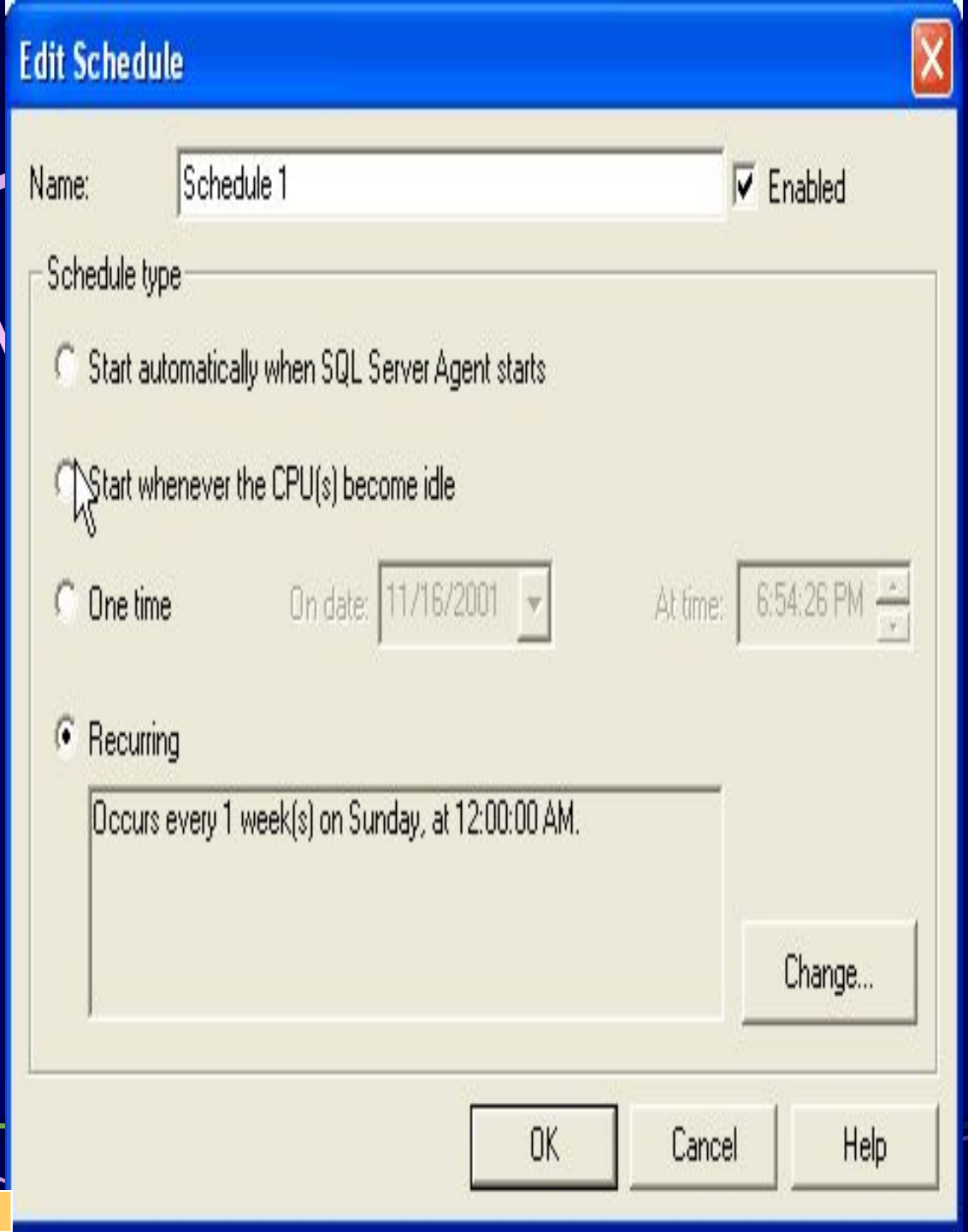
Full backup производит backup пользователей базы, но не производит бэкап логинов. Для того чтобы произвести backup логинов необходимо забэкапить базу данных master. В дальнейшем при восстановлении базы на другом сервере (имеющем свои логины) необходимо использовать процедуру sp_change_users_login для синхронизации имен логинов.

1. **Enterprise Manager**
2. tools
3. backup database



Указание места бэкапа.

Выбор периодичности и времени бэкапа.



Differential backup

Сохраняет только данные, которые изменились со времени последнего full backup’а Благодаря этому занимает гораздо меньше места на диске и выполняется существенно быстрее, что позволяет выполнять его чаще.

Backup можно выполнить также используя конструкцию T-SQL:

```
BACKUP   DATABASE   {   database_name   |   @database_name_var   }
  TO        <        backup_device        >        [        ,...n        ]
  [                                                                  WITH
       [ BLOCKSIZE   =   {  blocksize  |  @blocksize_variable  }  ]
       [ [  ,  ] DESCRIPTION  =  {  'text'  |  @text_variable  }  ]
            [       [       ,       ]       DIFFERENTIAL          ]
       [   [   ,   ]   EXPIREDATE   =   {   date   |   @date_var   }
            |   RETAINDAYS   =   {   days   |   @days_var   }   ]
       [ [  ,  ] PASSWORD  =  {  password  |  @password_variable  }  ]
           [       [       ,       ]       FORMAT       |       NOFORMAT       ]
          [     [     ,     ]     {     INIT     |     NOINIT     }     ]
       [ [  ,  ] MEDIADESCRIPTION = {  'text'  |  @text_variable  }  ]
       [ [  ,  ] MEDIANAME  =  {  media_name  |  @media_name_variable  }
  ]
         [   [   ,   ]   MEDIAPASSWORD   =   {   mediapassword   |
  @mediapassword_variable                      }                    ]
       [ [  ,  ] NAME  =  {  backup_set_name  |  @backup_set_name_var  }
  ]
          [     [     ,     ]     {     NOSKIP     |     SKIP     }     ]
         [     [     ,     ]     {    NOREWIND    |    REWIND    }     ]
         [     [     ,     ]     {    NOUNLOAD    |    UNLOAD    }     ]
              [        [        ,        ]        RESTART              ]
       [ [  ,  ] STATS [ = percentage ]
```

Пример - BACKUP DATABASE

-- Create a logical backup device for the full MyNwind backup.

- USE master
- EXEC sp_addumpdevice 'disk', 'MyNwind_1', DISK ='c:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\MyNwind_1.dat'

-- Back up the full MyNwind database.

- BACKUP DATABASE MyNwind TO MyNwind_1

Transaction-log backup

Включает в себя историю всех транзакций базы. Наличие такого лога позволит вам откатиться на любой момент времени до последнего full backup и привести базу в состояние, в котором она была в этот момент времени

```
BACKUP LOG { database_name | @database_name_var }
   {
        TO < backup_device > [ ,...n ]
        [ WITH
            [ BLOCKSIZE = { blocksize | @blocksize_variable } ]
            [ [ , ] DESCRIPTION = { 'text' | @text_variable } ]
            [ [ ,] EXPIREDATE = { date | @date_var }
                | RETAINDAYS = { days | @days_var } ]
            [ [ , ] PASSWORD = { password | @password_variable } ]
            [ [ , ] FORMAT | NOFORMAT ]
            [ [ , ] { INIT | NOINIT } ]
            [ [ , ] MEDIADESCRIPTION = { 'text' | @text_variable }
   ]
            [ [ , ] MEDIANAME = { media_name |
   @media_name_variable } ]
            [ [ , ] MEDIAPASSWORD = { mediapassword |
   @mediapassword_variable } ]
            [ [ , ] NAME = { backup_set_name |
   @backup_set_name_var } ]
            [ [ , ] NO_TRUNCATE ]
            [ [ , ] { NORECOVERY | STANDBY =undo_file_name } ]
            [ [ , ] { NOREWIND | REWIND } ]
            [ [ , ] { NOSKIP | SKIP } ]
            [ [ , ] { NOUNLOAD | UNLOAD } ]
            [ [ , ] RESTART ]
            [ [ , ] STATS [ =percentage ] ]
```


Бэкап базы и журнала.

- USE master EXEC sp_addumpdevice 'disk', 'MyNwind_2', 'c:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\MyNwind_2.dat'

--Create the log backup device.

- USE master EXEC sp_addumpdevice 'disk', 'MyNwindLog1', 'c:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\MyNwindLog1.dat'

-- Back up the full MyNwind database.

- BACKUP DATABASE MyNwind TO MyNwind_2 -- Update activity has occurred since the full database backup.

-- Back up the log of the MyNwind database.

- BACKUP LOG MyNwind TO MyNwindLog1

Для восстановления базы из Backup используется команда restore database, для восстановления лога – restore log, кроме того как всегда можно воспользоваться Enterprise Manager

```
RESTORE DATABASE { database_name | @database_name_var }
    [ FROM < backup_device > [ ,...n ] ]
    [ WITH
        [ RESTRICTED_USER ]
        [ [ , ] FILE = { file_number | @file_number } ]
        [ [ , ] PASSWORD = { password | @password_variable } ]
        [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
        [ [ , ] MEDIAPASSWORD = { mediapassword | @mediapassword_variable
    } ]
        [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ]
                [ ,...n ]
        [ [ , ] KEEP_REPLICATION ]
        [ [ , ] { NORECOVERY | RECOVERY | STANDBY = undo_file_name } ]
        [ [ , ] { NOREWIND | REWIND } ]
        [ [ , ] { NOUNLOAD | UNLOAD } ]
        [ [ , ] REPLACE ]
        [ [ , ] RESTART ]
        [ [ , ] STATS [ = percentage ] ]
```


```
RESTORE LOG { database_name | @database_name_var }
    [ FROM < backup_device > [ ,...n ] ]
    [ WITH
        [ RESTRICTED_USER ]
        [ [ , ] FILE = { file_number | @file_number } ]
        [ [ , ] PASSWORD = { password | @password_variable } ]
        [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ]
                [ ,...n ]
        [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
        [ [ , ] MEDIAPASSWORD = { mediapassword | @mediapassword_variable
    } ]
        [ [ , ] KEEP_REPLICATION ]
        [ [ , ] { NORECOVERY | RECOVERY | STANDBY = undo_file_name } ]
        [ [ , ] { NOREWIND | REWIND } ]
        [ [ , ] { NOUNLOAD | UNLOAD } ]
        [ [ , ] RESTART ]
        [ [ , ] STATS [= percentage ] ]
        [ [ , ] STOPAT = { date_time | @date_time_var } .
            | [ , ] STOPATMARK = 'mark_name' [ AFTER datetime ]
            | [ , ] STOPBEFOREMARK = 'mark_name' [ AFTER datetime ]
        ]
    ]
```


1. RESTORE DATABASE MyNwind FROM MyNwind_1

1. RESTORE DATABASE MyNwind FROM MyNwind_1
   WITH NORECOVERY RESTORE DATABASE MyNwind FROM MyNwind_1 WITH FILE = 2

При настройке бэкапа в EM имеется возможность выбрать одну из 3-х recovery моделей

- Simple recovery – позволяет в дальнейшем восстановить базу в состояние в котором она была при бэкапе, то есть дойти до состояния в котором возник сбой не удастся
- Full recovery – можно дойти до состояния при котором возник сбой (бэкапится transaction log)
- Bulk-Logged Recovery – тоже самое что и Full Recovery, но оптимизировано бэкапит операции insert into и bulk insert – занимает в итоге меньше места, но и гарантия восстановления ниже

This example restores the transaction log to the mark named "RoyaltyUpdate."

1. BEGIN TRANSACTION RoyaltyUpdate WITH MARK 'Update royalty values'
2. GO
3. USE pubs
4. GO
5. UPDATE roysched SET royalty = royalty * 1.10 WHERE title_id LIKE 'PC%'
6. GO
7. COMMIT TRANSACTION RoyaltyUpdate
8. GO --Time passes. Regular database --and log backups are taken. --An error occurs.
9. USE master
10. GO
11. RESTORE DATABASE pubs FROM Pubs1 WITH FILE = 3, NORECOVERY
12. GO
13. RESTORE LOG pubs FROM Pubs1 WITH FILE = 4, STOPATMARK = 'RoyaltyUpdate'