www.DBTechNet.org

DBTech EXT Backup and Recovery Labs (RCLabs)

With the support of the EC LLP Transversal programme of the European Union

Disclaimers

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Trademarks of products mentioned are trademarks of the product vendors.

Learning Objectives:

- understanding the need, techniques, procedures, and major options of Database Backups
- understanding basics of Database Recovery procedure of the mainstream DBMS systems
- understanding the need of disaster recovery plan and backup strategy

Prerequisites:

We expect that the participant of the labs is familiar with SQL basics and knows how to

- start the database server in the selected DBMS environment
- use SQL commands by the tools of this selected DBMS

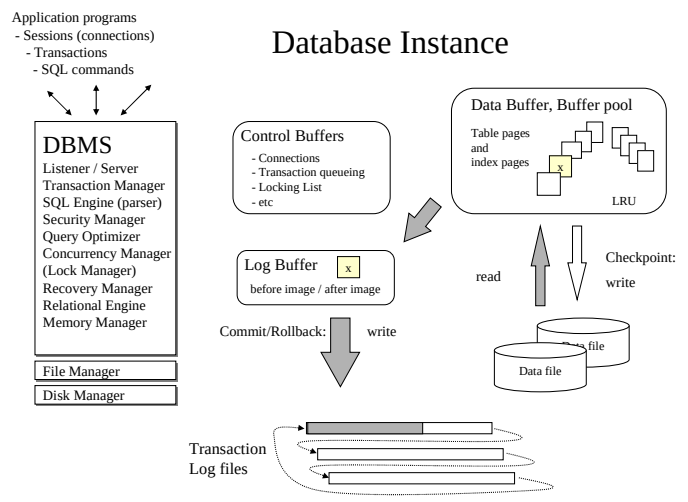
Contents

Part I - Introduction to Database Backups and Recovery 2
Database Server Instance 3
Database data files, pages, and transaction log 3
Managing Transaction Logs 5
Rollback Recovery 7
Managing Backups 9
Roll-forward Recovery 10
Review Exercises: 11
Part II - Recovery Labs 13
Software to be used and the Learning Environment alternatives: 13
RCLabSS - SQL Server Recovery Labs 14
Short introduction to SQL Server Architecture 14
RCLabSS1 - Transaction log 15
RCLabSS2 - Rollback Recovery after a “soft crash” 16
RCLabSS3 – Online Backup 22
RCLabSS4 - Backups and Roll-forward Recovery to a Point-in-Time 25
Review questions and exercises 36
RCLabDB2 – DB2 Express-C Recovery Labs 37
RCLabDB23 - Online Backup 37
RCLabDB24 - Point In Time (PIT) Recovery 39
RCLabOra – Oracle XE Recovery Labs 41
References and Links 41

Part I - Introduction to Database Backups and Recovery

Databases provide a reliable storage for the persistent data of applications. The concepts DBMS system, database instance, and database have no globally accepted standard definition, but for this tutorial we adopt the definitions for these concepts from database administrator's (DBA) point of view to the mainstream DBMS products: DB2, SQL Server, and Oracle, the "big three". A DBMS software product can be installed and used as a database server, or more precisely, one or more configurable database server instances.

Figure 1. Database Server Instance



Database Server Instance

A database server instance is built by installing a DBMS software and configuring a named operational set of DBMS processes which take care of various DBMS services and various memory caches, such as data buffer i.e. buffer pool, log buffers and various control buffers. The configuration is usually controlled in some control files. The instance can be started (processes instantiated) as service and stopped (shutdown). It can be configured to start automatically or manually. An instance manages one or more databases which have data files and transaction log files of their own, but share the buffer pool of the instance. In the following we may use the term DBMS also for the database server instance. The database server instance provides services for the reliable storing and retrieving data.

Database data files, pages, and transaction log

A database is a collection of object structures (tablespaces, tables, indexes, etc) and data which is managed as a "consistent whole". The database contents are stored in one or more data files on discs and these files are managed as file groups called tablespaces. A table or an index on table is created in a tablespace. This means that the pages of the created object will be stored in file pages of some files of that tablespace. The data files are identified internally by file numbers given by the instance, and managed as a sequence of pages (also called as blocks) having blocksize of 4, 8, 16, .. KB and identified accordingly by page numbers in the file.

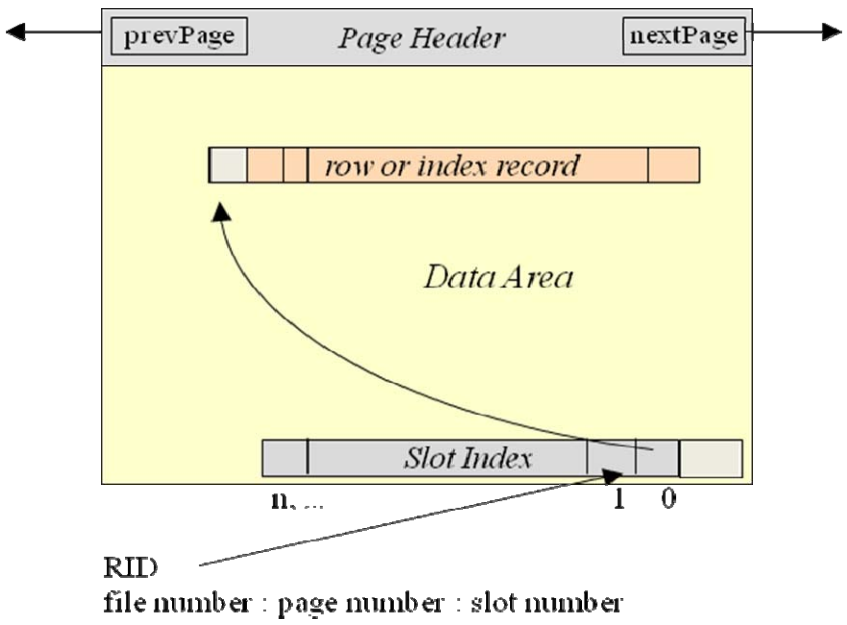


Figure 2. Typical page format

The typical page format of a table or index is presented in Figure 2. Every page has the following 3 parts:

- page header containing various control data used by the DBMS,
- data area for variable length records for storing rows or index entries of the object structure

- slot index (slot directory) containing offset addresses of the records on the data area.

Typically a page registers rows of one table only, and the table is indicated in a field of the page header. Rows are stored on the data area records, which contain also some control information of the row, such as column lengths and offsets on the record. Row addressing is based on the indirect address RID (also called as ROWID, or tuple id TID) which is indicative of the internal file number, page number, and slot number. In case a modified row has grown in size so that it no longer fits in the original page, the RID address remains the same but the record is split into parts which are chained on other pages where there is enough room to accommodate them. For more detailed information of the page header fields and the record structures we refer to textbooks such as [5] and DBMS product manuals such as [9].

Pages of a table (or an index) are double chained, and for faster sequential access of these pages, in case of pre-fetching, grouped in 8 or more pages (called as extents) in the chain stored adjacent to each other on disc. Also for fast finding of pages in which there is room for new records, the database maintains chains of free pages in free lists.

For performance reasons the needed index and data pages are first fetched into buffer pool page frames of the instance allocated in main memory of the server computer. Fetched pages remain in the buffer pool as long as there is room to accommodate new pages, and so if a page is needed again, no disc IO is spent for retrieving the page. This is the main performance benefit and the key for scalability of multi-user databases.

The buffer manager of the DBMS keeps record of the use of the fetched pages in the buffer pool by 2 attributes of the page frames:

- *dirty bit*, which is turned *off* when the page is first loaded to the buffer pool, and it is turned *on* when the page content is modified, for example, a row is inserted, updated, or deleted
- *pin_count*, contains the number of the current users of the page.

The typical *page replacement policy* of pages in the buffer pool is called Least Recently Used (LRU) algorithm, which adds a page to the end of candidate pages to be thrown away from the buffer pool when none of the current transactions has been using it, - and when the buffer manager needs room for new pages in the buffer pool, a database checkpoint operation is executed and the frames at the beginning of this queue will be released for the new pages to be fetched [5].

SQL transactions provide the means for applications to use reliably these services of the DBMS. At the beginning of every transaction, the transaction will get an internal, unique transaction id number.

All data updates on pages in the buffer pool are first written to sequential transaction log files (also called as journals) of the database as log records. Depending on the DBMS, a log record may contain the following fields

- log record sequence number (LSN),
- timestamp
- the transaction id
- type of the log record (start, insert, update, delete, commit, rollback)
- object information
- before image and after image of the maintained row
- chaining information of records of the transaction [4].

The records are first added in the log buffer of the database, and latest at the commit or rollback time of the transaction the records will be written in the transaction log on disc. The latest LSN corresponding to changes on data area of a page is registered in a field on the page header. To avoid unnecessary disc IO, the maintained pages (those having dirty bit on) are written back to the data files at database checkpoint operations (which will write checkpoint log records including the current transaction ids in the transaction log file), or by the lazy writer process as a background operation, and after this the dirty bits of those pages

are turned off. This procedure is called as Write-ahead-Logging (WAL) protocol of the ARIES algorithms [2] [5]. A transaction is a **recoverable** unit of work in terms that when receiving **rollback** command, the DBMS rewrites the before images from the transaction log records (or rollback segments / UndoTablespace in case of Oracle) back to the data pages in the buffer pool.

Managing Transaction Logs

The transaction log files are the most important files in a database instance, since these contain the data modifications made by the latest transactions of the database, even those which have not yet been written to the actual database.

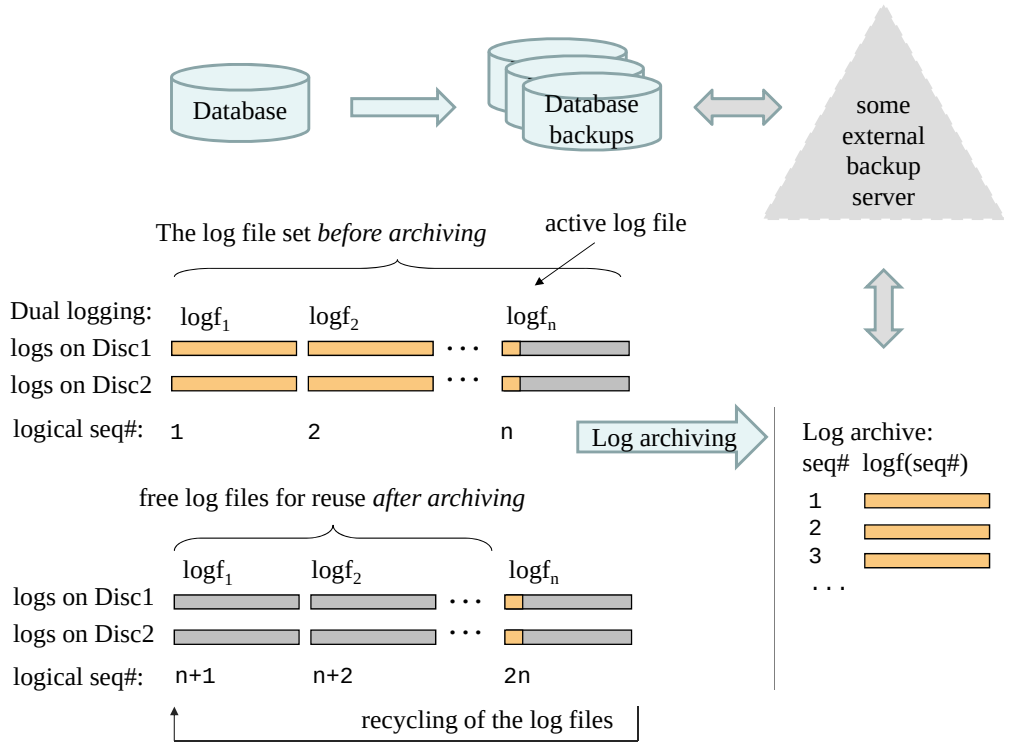


Figure 3. Managing transaction log files of database

Figure 3 presents how transaction log files should be managed in a generic DBMS system (approximately like Oracle DBMS). Instead of a single transaction log file, a **log file set** of some 3 or more transaction files (also called as "redo log files", presented as logf files in the figure) should be allocated by DBA for every database. These will be filled in sequence by the DBMS so that one logf is the **active log file** at any time, and when it gets full, the next logf in the file set will become active. This operation is called **log switching**.

When the last logf of the file set gets full, the physical log files will be reused starting from the first logf. This is called log recycling, and it would overwrite the transaction history written in the log file set, which may be OK for some databases which are not production databases, for example for test databases. However, a production database should be configured to run in archive mode, which means that in the event of log switching, the previous active log will get copied (i.e. archived) into the log archive of the database named according to the ever increasing sequence number (seq#) of the logical transaction log files, starting from seq# value 1. After the archiving operation the physical log file will be free for reusing as shown in Figure 3, and the transaction history older than the last database backup can be recovered from this log archive. We will explain the database backups in the following.

The ACID acronym outlines the properties of an ideal database transaction [1] in terms that an ACID transaction shall

- A be atomic so that the original state can be recovered by rollback operation
- C not violate the consistency of the database
- I operate isolated of concurrent transactions (without side-effects)
- D durable so that committed data can be recovered form transaction logs after any database failures.

However, the mainstream DBMS systems used by ICT industry provide possibilities, and actually as default, the use of non-ACID transactions, especially giving up of strict isolation for better performance of the system. This presents more challenge to applications developers in the art of transaction programming. We cover the isolation requirement in detail in a separate workshop on SQL Concurrency Technologies. On consistency requirement, we refer to textbooks and our workshop on Database Modeling and Design.

In this paper, we focus on the A and D properties of the ACID transactions, which rely on the proper services of the DBMS.

Transaction log files serve as the basis for atomicity of transactions, since in rollback recovery operation the DBMS will restore the before images of all modifications back to the appropriate data pages. These may be available in the log buffer, but this is not guaranteed, so transaction files must be large enough to manage all log records of any mix of concurrently active transactions.

Transaction log files serve also as the basis of durability of all committed transactions, and we can say that the transaction log files are the most important files of the database, since they register the transaction history up to the last committed transaction. Therefore, when dealing with critical data, the transaction log file sets should be duplicated and these duplicate sets should be allocated on 2 different discs. This is called as dual logging. There shall be no single-point-of-failure, so the discs of these transaction log sets should be in separate buildings connected with different disc controllers and using different uninterruptable power supplies (UPS). Durability, as part of the reliability, can be guaranteed service only in properly managed database environment!

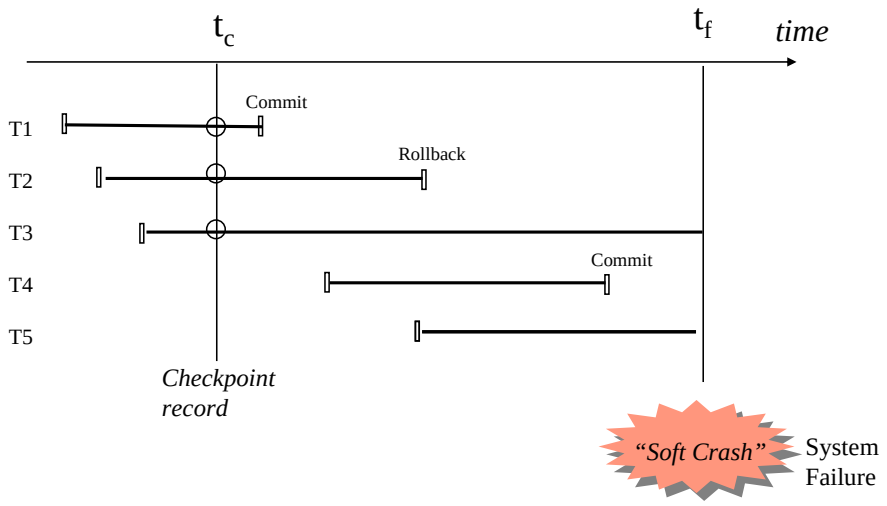
For fault-tolerant services, the database should also be replicated at an other, preferably remote computer acting as stand-by server which will continue the service of applications in case the master server fails.

Rollback Recovery

In case of a "soft crash" (such as power, operating system, or DBMS failure) the contents in the buffer pool (RAM) is lost, so only the current contents in the data files and transactions logs on disks are durable, fortunately up to the last committed transaction.

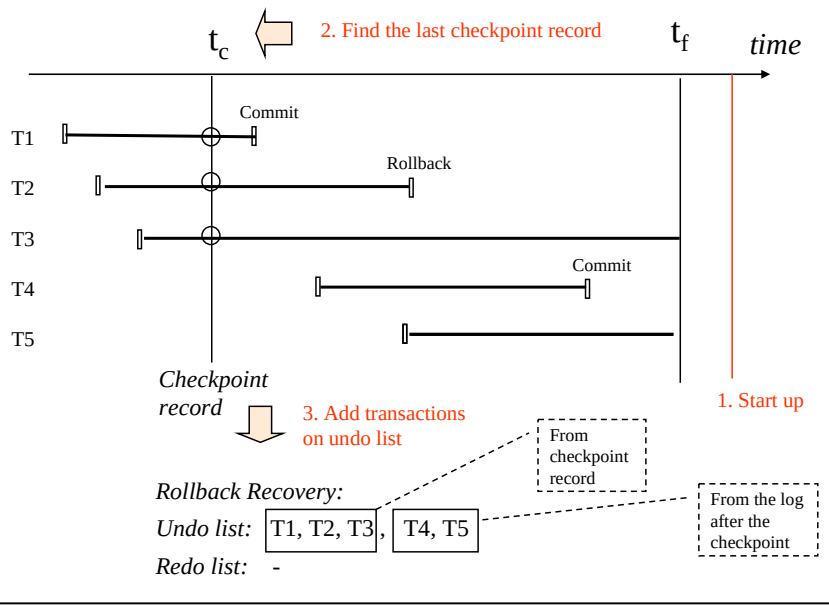
When the failure is over, and database instance restarted, the DBMS will recover database contents up to the last committed transaction based on the transaction history in the transaction log files (see Figure 4).

Figure 4. Transaction log history at a system failure



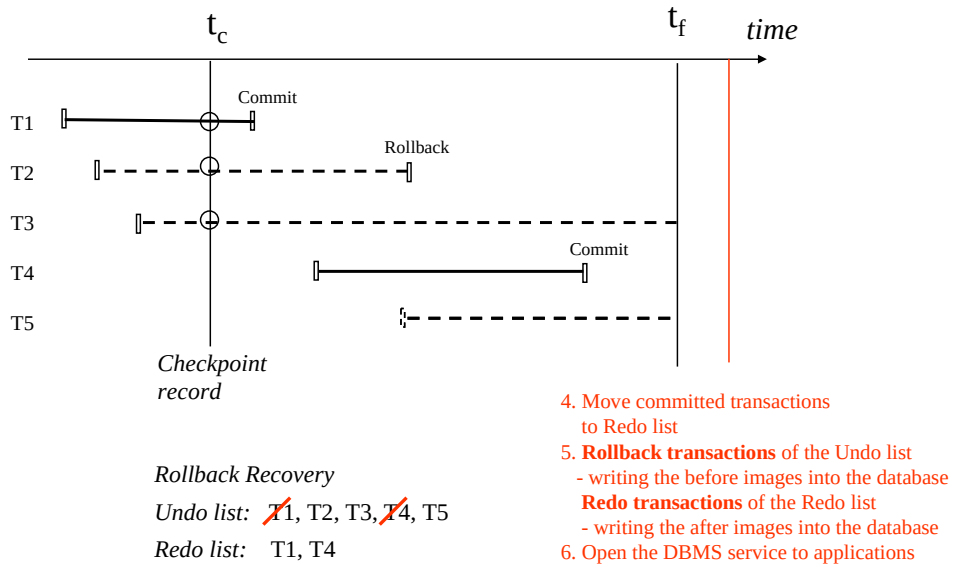
The internal steps of the DBMS of this recovery procedure, called **rollback recovery** (or **system recovery**, or **restart recovery**), are explained well in most database textbooks. When restarted, the DBMS looks first the transaction log to find out the last checkpoint record. The checkpoint record shows which transactions were active at that time, and based on subsequent analysis of the transaction log, DBMS finds the transactions which were started after the checkpoint, as shown in Figure 5.

Figure 5. Start of rollback recovery using the transaction log



All these transactions are first registered in the Undo list for rollback. Then those transactions, which have commit record in the transaction log are moved to the Redo list. For all transactions in the Undo list, the before images of the rows are written to the database, and for all transactions in the Redo list since the checkpoint record the after images of all rows are written in the database, as presented in Figure 6.

Figure 6. Rollback Recovery



According to C. J. Date [11] earlier DBMS versions simply processed the UNDO list first and then the REDO list as presented in Figure 6.

After C. Mohan [2] presented the ARIES algorithms ("Algorithms for Recovery and Isolation Exploiting Semantics") in 1990-92, many systems, for example DB2 and SQL Server, have adopted some variant of the recovery method of ARIES scheme, in which the REDO list is processed first for better performance, and then redoing also the UNDO list with the rollbacks, just repeating the history including the lock operations and logging the redoing again. In redoing the transaction history, the pages in data files are updated only when the LSN in the page header is older than the LSN of the affecting transaction log record [2].

Finally the database is turned into active state available for applications.

Managing Backups

One of the most important responsibilities of DBAs is to organize the backup and recovery routines of the databases to guarantee durability under any any circumstances for the data stored in the databases by transactions.

It is not reliable enough that database data is safely written on discs in the transaction log files and in the database files. The physical files may be lost due to various reasons - some software, hardware, or operation failures, or various accidents. Due to some operation failures, sometimes it may be necessary to restore the database to the state it was before that operation. Therefore, the files have to be backed up according to a well planned backup schedule, which depends on how critical the database is and what kind of disaster recovery plan we have. The lost database should be restored back in the consistent state covering the last committed transaction, or the database contents should be recovered to the last committed transaction at some known point-in-time in case of some application-level problems. This means that a backup of the file system is not enough as a backup of the database! Therefore the current mainstream DBMS systems include sophisticated backup and recovery methods and options of their own.

Database backup can be a full backup including all files of the database, a consistent copy at a moment of time, an incremental backup including only the data which have changed after the last full backup, or a differential backup including only the data (pages?) which have changed after the last full or differential backup, depending on the DBMS implementation. Some database backup products provide also options of backing up selected tablespaces, files, or tables, but remember that these options will risk the consistency of the database contents.

A simple consistent backup can be created when we take the database out of the users into single-user mode for the so called offline backup. In case of 24x7 production database, this not possible any more, and modern DBMS systems can provide online backup which includes also the transaction history of the backup window of time, so that a consistent content of the end moment of the backup can be recovered.

Medias of various tape technologies, such as Digital Audio Tape (DAT), Digital Linear Tape (DLT), etc have been used for backups, but as disk storage is becoming cheaper and cheaper, disk based backups have become the typical solution. Also special backup servers with robotics and dedicated database on history and metadata of the backup files, as presented in Figure 7, are often used for large and business critical databases [7] [10].

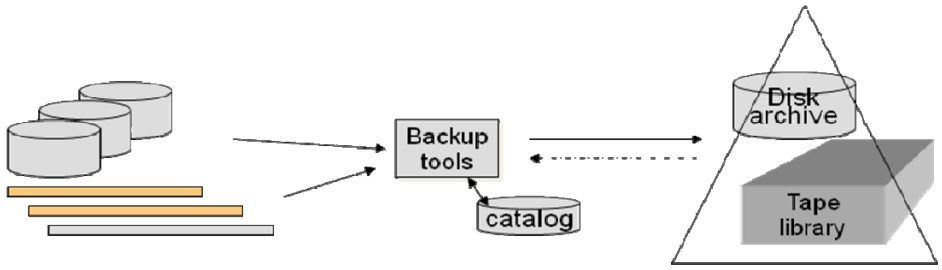


Figure 7. Using a backup server

Roll-forward Recovery

Figure 8 presents how a database can be recovered after a hardware or media failure, or for recovering the database after some serious human operation or programming error to some state before the error, or rebuilding the database in a new environment after some disaster. As the typical case, the database is first restored from the last database full backup followed by roll-forward recovery (also called as restore recovery or media recovery) which writes the after images of all rows in the transaction log into the database for all committed transactions, repeating the transaction history since the database backup.

In case we have incremental backups, the latest incremental backup is restored after the restored full backup, followed by roll-forward recovery of the transaction log history registered after the incremental backup. In case we have differential backups, all the differential backups taken after the full backup will have to be restored after the restoration of the full backup, followed by a roll-forward recovery of the transaction history born after the last differential backup. At the end of roll-forward recovery of transaction history, the DBMS always needs to undergoa rollback recovery based on the last checkpoint of the transaction log.

Note: The terms incremental and differential backup have not been standardized. For example, SQL Server's differential backup actually works like we have explained on incremental backups above. DB2 provides both types, but calls differential backup as "incremental delta".

After a successful roll-forward recovery, it is the right time to make a new full backup!

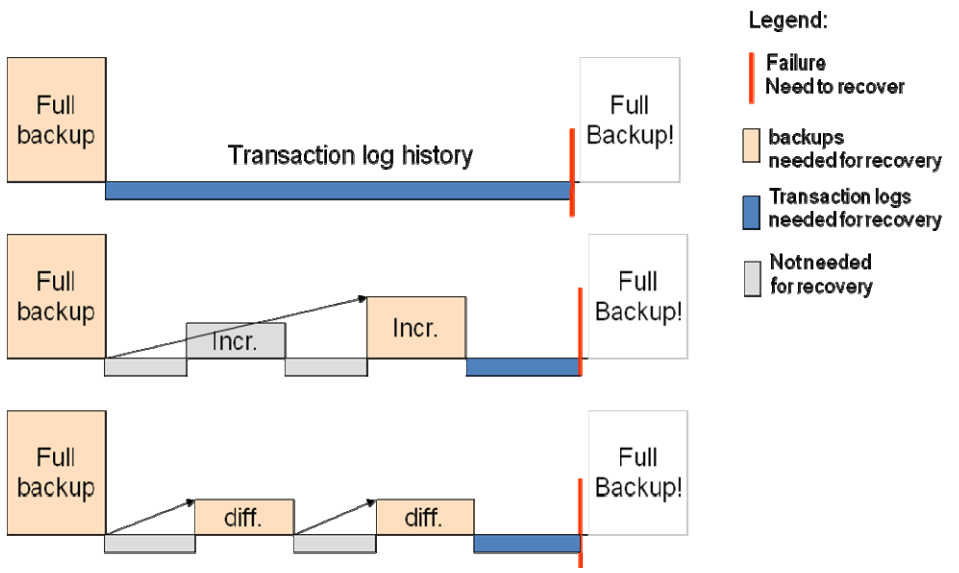


Figure 8. Types of Roll-forward Recovery after a database failure

Speed of recovery counts! The recovery of the database should be practiced beforehand, and it should be implemented in the form of as much an automatic procedure as possible. The whole procedure takes time when it is applied manually, and human thinking time between the recovery steps can be expensive when customers and employees of the company are put on hold, waiting for the system to return to operation. When dealing with business critical data content, a waiting time of hours may not be acceptable at all, and the database is mirrored/replicated to some safe and distant server, a hot standby database, which takes over the production/operation load in the case of a (primary) database server failure.

Accurate definitions of the concepts involved, and detailed system operation and usage descriptions can be found in the user- and reference manuals of the corresponding software products. Luckily, most of the manuals are available for free as PDF documents on the Web.

Review Exercises:

(assuming that you are using for example DB2, SQL Server, or Oracle as your DBMS)

Note: the answers to some of questions that follow may not be present in this tutorial, and you may have to consult the relevant literature/bibliography provided.

1. Explain the concept of a database server instance of your DBMS and how it can be configured. Does it support a single database or multiple databases?

2. How does the buffer pool improve the performance of the database server?

3. Explain the concepts of dirty pages, and LRU page replacement policy.

4. What are the three possible types of status for a page in the buffer pool?

5. Explain the concept of Lazy Writer.

6. Find out from the documentation of your DBMS system which events activate a checkpoint operation in your DBMS.

7. Find out from the documentation of your DBMS system which SQL operations are not be logged in the transaction log files by your DBMS.

8. Based on the documentation of your DBMS, list the steps that comprise a checkpoint operation, and their order of execution.

9. Sort out the syntax and the input parameters of the BACKUP DATABASE command of your DBMS.

10. Sort out how to configure and activate your database so that it operates in the "Archive Mode".

11. Does your DBMS support dual logging?

12. Why standby databases, replication, or disk mirroring cannot be used to replace traditional backup and recovery? (source: [3])

13. Describe three types of database failures that may require point-in-time recovery.

14. Describe two ways to recover an index. (source: [3])

15. Name four factors that have an influence on the efficiency of a database recovery procedure. (source: [3])

16. Describe what is a disaster recovery plan. Why does it have to be tested beforehand?

17. What does the single-user mode of database mean? Describe situations when it is needed.

18. What does the quiesced mode of a database or a tablespace mean?

Part II - Recovery Labs

Software to be used and the Learning Environment alternatives:

a) DBTLab on Linux (ref 19)

To make at least part of the materials of the popular DBTech Pro live workshops available to public, to schools, individual students and database professionals in their own Life-Long Learning programmes, we provide downloadable private labs built of free or open source software and free professional, mainstream database systems, such as DB2 Express-C of IBM and Oracle XE of Oracle, available on multiple operating system platforms, including free Linux platforms. SQL Server Express of Microsoft is also freely available, but since it is available only on the proprietary Windows platforms of Microsoft, we cover its features and examples in our tutorials, but users need to build the SQL Server labs of their own.

These are the current downloadable Database Labs:

- `www.DBTechNet.org/download/VMware_Ubuntu_DB2Lab.zip`
  DB2 Express-C V9.5 on Ubuntu 7.10 Linux in VMware virtual machine image prepared for us by IBM Finland. In the ZIP archive we have included also the free VMware Player for Windows, installing notes, learning materials provided by IBM, and the excellent SQL Cookbook of Graeme Birchall.
  Note:
  The latest version of DB2 Express-C is available at IBM's web site
  `http://www-01.ibm.com/software/data/db2/express/download.html`
  The latest version of Ubuntu is available at Canonical Ltd.'s web site
  `http://www.ubuntu.com/`

- `www.DBTechNet/download/Ubuntu_OracleXE.zip`
  Oracle XE 10g on Ubuntu 8.4 Linux in Microsoft Virtual PC 2007 image. The main user in this lab is `dbtech` and the initial password of the user is `dbtech`.
  Microsoft Virtual PC 2007 is freely available at Microsoft's download center

Both DBTLabs can be used as private labs on experimenting with the appropriate examples and exercises of the following tutorials.

b) Own environment and DBMS instances

We will focus on the mainstream DBMS systems: DB2, Microsoft SQL Server, and Oracle. You may already have some of these available, but if not, you may download and install the latest free "Express" editions of these on the own computers, for example

- DB2 Express-C you may download from http://www-01.ibm.com/software/data/db2/express/ (or look for the current site)
- Microsoft SQL Server Express you will find starting from page http://www.microsoft.com/sql
- Oracle XE you will find starting from page http://otn.oracle.com

For all these systems you will also find the installing and "start to use" instructions from the corresponding download sites.

RCLabSS - SQL Server Recovery Labs

Short introduction to SQL Server Architecture

Of the Big Three DBMS, Microsoft SQL Server is widely used and often also in companies which are using DB2 or Oracle as their main DBMS systems. The architecture of Microsoft SQL Server is a bit different from what we have described above, so it is worth of a closer look. Also, due to the easy user interface of SQL Server Management Studio, we have selected it as the DBMS to be used in this first version of our Recovery Virtual Lab workshop.

Beside the application databases, a SQL Server instance has always the following system databases:
- Master containing configuration, security, and metadata of the instance and registry of the databases
- Msdb containing among other things backup histories of databases of the instance
- Tempdb providing temporary storage for databases of the instance
- Model template database from which the application databases will be cloned.

Every database of the instance has a physical transaction log file of its own, but this file consists of a sequence of virtual log files (VLF), which are used like the log files we have described in our generic architecture description above. Archiving of the VLF files is done using BACKUP LOG commands appending the backups into a backup file on a disk which can be local or may reside on a remote file server.

Checkpoint operations occur per database and as default every minute, or on certain activities which are listed in Books Online documentation [9]. The time interval between checkpoints can also be configured longer than one minute, but on instance level. The option is called recovery interval, since frequency of checkpoint records in transaction log affects the duration of the rollback recovery on databases when the instance is restarted.

Page size in SQL server is fixed 8 KB.

SQL Server does not use the term Archiving Mode, but a database can be configured to use one of the following recovery models:
- simple which is equal to a non-archive mode and suits for test databases only
- full this corresponds the archiving mode, and should be configured for all production databases
- bulk loggeg uses minimal logging for bulk operations. For example, this model does not log CREATE INDEX and bulk copy operations. This recovery model can be used temporarily, but after the bulk operations, full backup of the database should be taken and database configured to the full recovery model.

For more details of the SQL Server architecture, we refer to SQL Server Books Online documentation.
Due to the easy database management by the SQL Server Management Studio (SSMS), in this version of the Recovery Lab we experiment with Microsoft SQL Server, in environment type "c)" which you need to install yourself. However, with help of the backup tutorials / manuals of DB2 or Oracle you can apply our examples in your virtual labs as well.

To keep this tutorial simple, we assume that you are the local administrator in the system.

RCLabSS1 - Transaction log

In this lab we experiment with the transaction log of SQL Server using the Transact-SQL script file from
http://www.DBTechNet.org/labs/ccr_lab/ViewTransactionLog.txt
in which we first create a new database LogTest and table TestTable in which we insert some rows transaction.
We inspect the contents of the transaction log using the undocumented SQL Server function

```
::fn_dblog(@StartingLSN, @EndingLSN)
```

which provides various technical information about the transaction log records in derived table format from the selected range of LSN values, and using NULL values as arguments, the whole log. Selecting all the fields from that derived table shows that transaction logs have many details which are made public, and, for example, in the column [Log Record] we get the whole record in a hexa string. However, for our purposes, we can restrict to those columns used in the script.

Transactions

It is especially interesting to verify which operations are logged in the transaction log for transactions and how the log is used in the **autocommit** mode, which is the default transactional behavior of SQL Server.

We can also use the undocumented DBCC command

```
DBCC LOG (<database name>,<output mode>)
```

which gives some more information, but with fixed columns. You may try the command

```
DBCC LOG (LogTest, 2)
```

and see that SQL Server 2008 logs also locking, which will help in REDOing the committed transactions .

Checkpoint test

To explore the checkpoint records of SQL Server we start three simultaneous SSMS Query sessions connected to our LogTest database and enter the following commands in the order presented below:

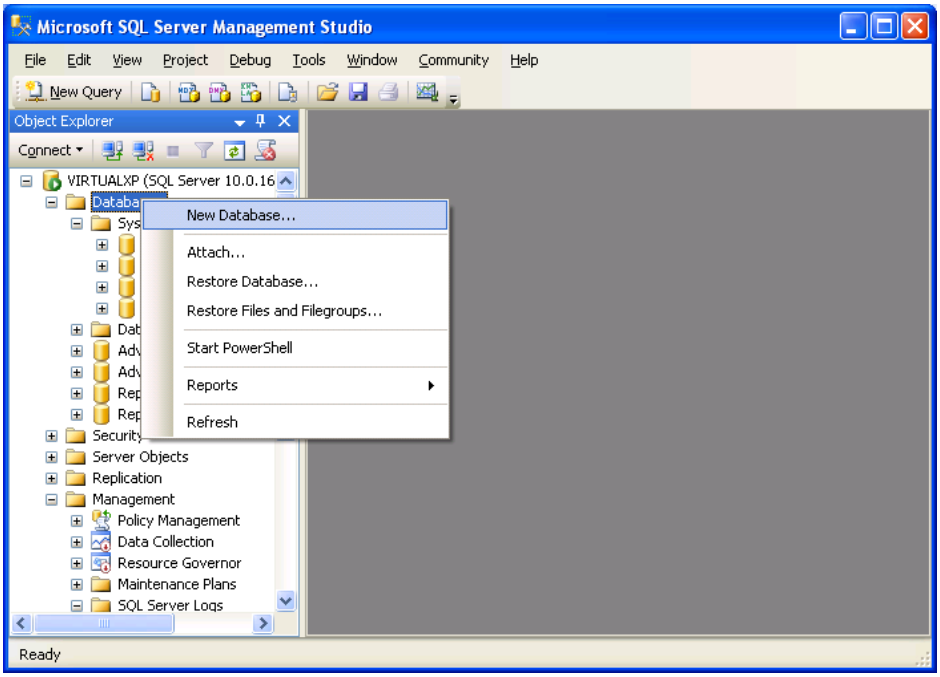
```
/* Session 1 */
BEGIN TRANSACTION
  INSERT INTO TestTable (number) values (51)

/* Session 2 */
BEGIN TRANSACTION
  INSERT INTO TestTable (number) values (52)

/* Session 3 */
CHECKPOINT
SELECT [Current LSN], Operation, [Transaction ID],
       [Num Transactions], Description
  FROM fn_dblog(NULL,NULL)
```

and the three last records in the transaction log contain the information captured from the checkpoint operation. The Description column of the second checkpoint record contains the list of the current transaction ids.

RCLabSS2 - Rollback Recovery after a “soft crash”

In this lab, we will simulate database recovery of SQL Server for 5 concurrent transactions after a system failure presented in Fig 3.

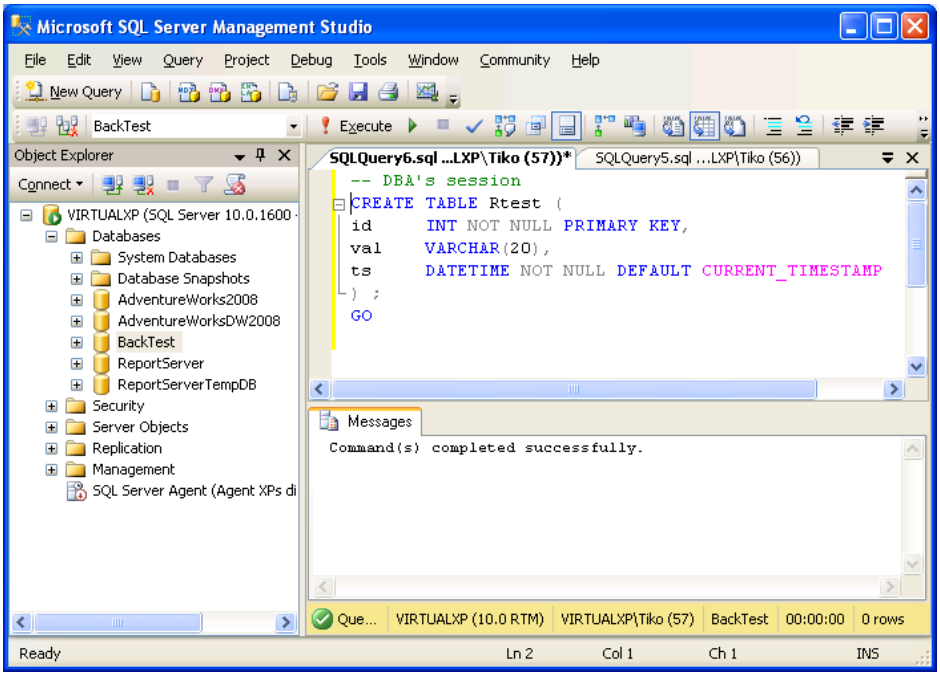
First we will create a test database starting SQL Server Management Studio, then selecting Databases in the Object Explorer, and by right mouse button selecting "New Database..."



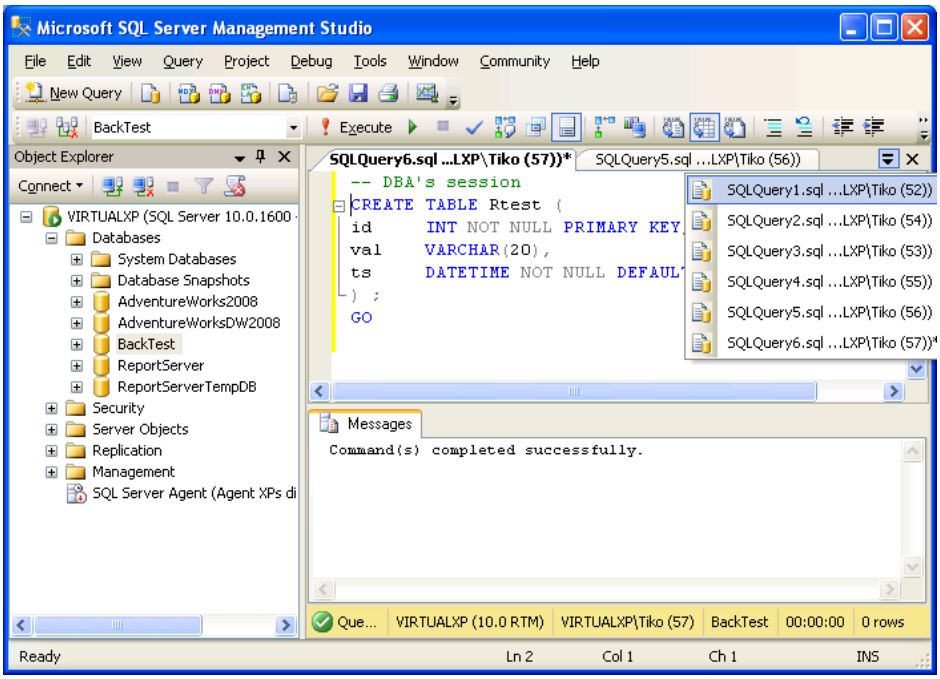
We enter "BackTest" as name of the database, and press OK.

Now we will select the database BackTest and by the right mouse button select "New Query" 6 times to start 6 concurrent SQL connections to the database. We will use the SQL session in Query window "SQLQuer6.sql" as DBA's session and create table Rtest using the following commands

```
-- DBA's session in SQLQuery6
CREATE TABLE Rtest (
id      INT NOT NULL PRIMARY KEY,
val     VARCHAR(20),
ts      DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
) ;
GO
```

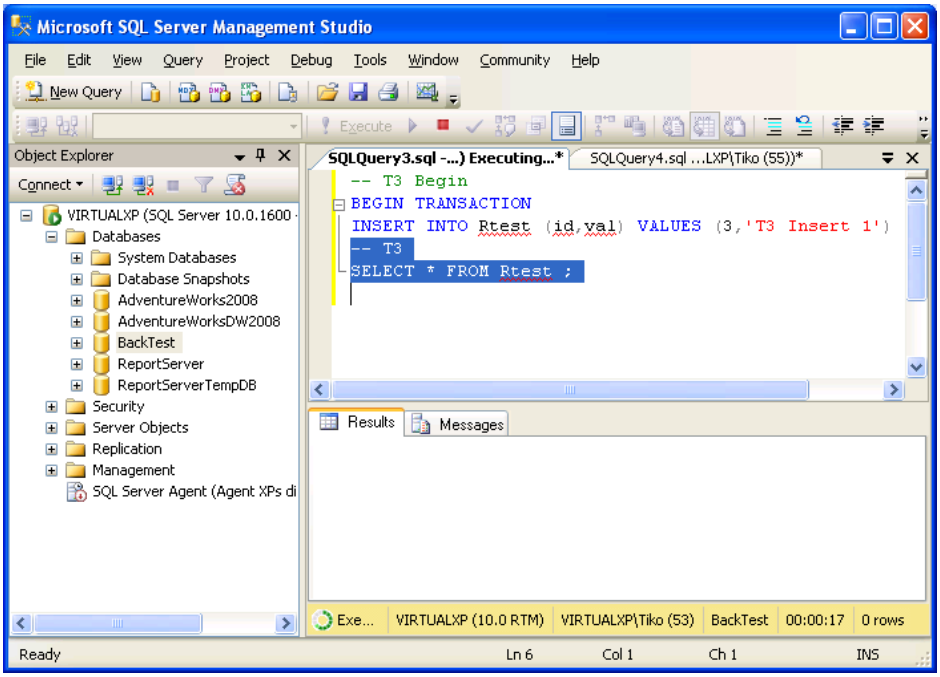


After this, we will proceed with sessions for concurrent transactions T1 .. T5 selecting always the corresponding Query window from the selector button on the top right corner as follows pressing always the "Execute" button after copying the appropriate commands into the Query window




```
-- T1 Begin
BEGIN TRANSACTION
INSERT INTO Rtest (id,val) VALUES (1,'T1 Insert 1') ;
```

```
-- T2 Begin
BEGIN TRANSACTION
INSERT INTO Rtest (id,val) VALUES (2,'T2 Insert 1') ;

-- T3 Begin
BEGIN TRANSACTION
INSERT INTO Rtest (id,val) VALUES (3,'T3 Insert 1') ;

-- DBA's session in SQLQuery6
CHECKPOINT

-- T1
INSERT INTO Rtest (id,val) VALUES (4,'T1 Insert 2') ;
COMMIT WORK ;

-- T2
INSERT INTO Rtest (id,val) VALUES (5,'T2 Insert 2') ;

-- T4 Begin
BEGIN TRANSACTION
INSERT INTO Rtest (id,val) VALUES (6,'T4 Insert 1') ;

-- T5 Begin
BEGIN TRANSACTION
INSERT INTO Rtest (id,val) VALUES (7,'T5 Insert 1') ;

-- T2
UPDATE Rtest SET val = 'T2 Update of 2' WHERE id = 5 ;
ROLLBACK WORK ;

-- T4
INSERT INTO Rtest (id,val) VALUES (8,'T4 Insert 2') ;
COMMIT;

-- T3
SELECT * FROM Rtest ;
```

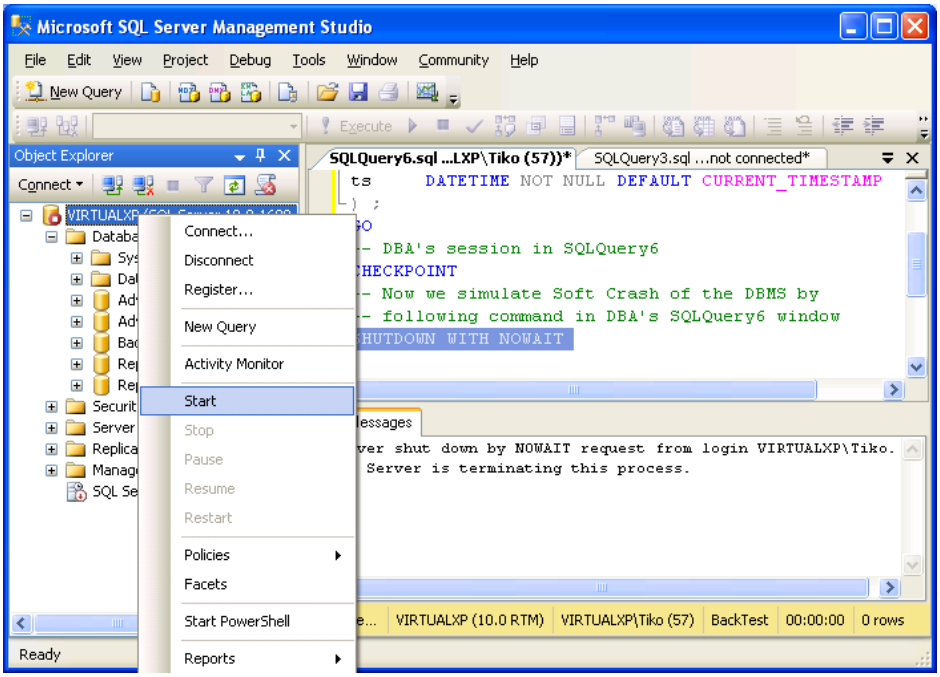



```
-- Now we simulate Soft Crash of the DBMS by
-- following command in DBA's SQLQuery6 window
SHUTDOWN WITH NOWAIT
```

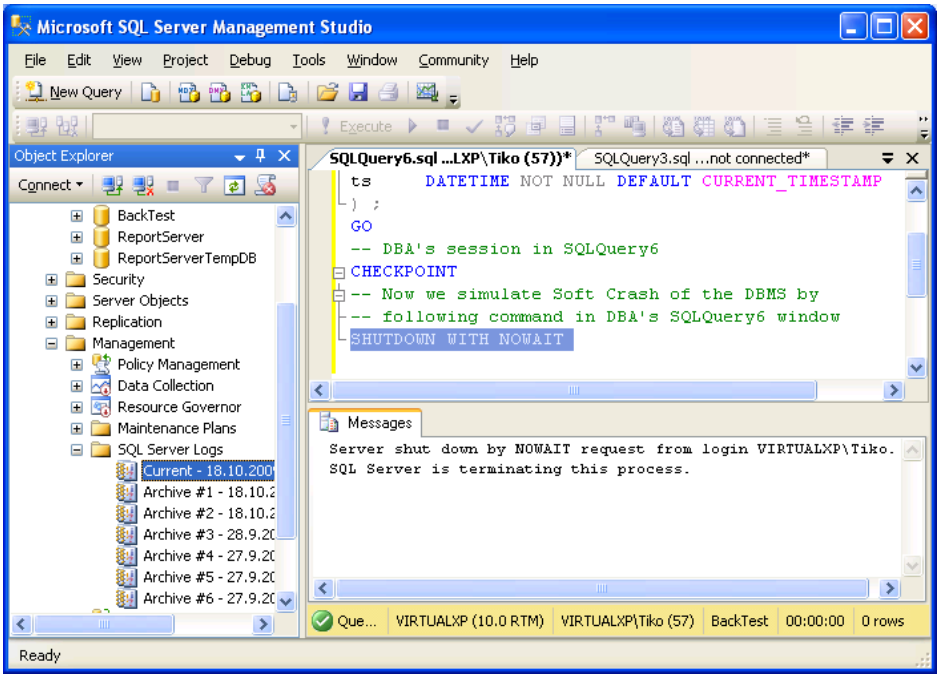
Checkpoints are periodically done by the DBMS after some short intervals (which can be configured) or amount of data written into the transaction log, but just for our experiment, we executed above an explicit checkpoint using DBA's CHECKPOINT command.

`SHUTDOWN WITH NOWAIT` command shuts all databases of the instance without executing checkpoint - which means that the transaction logs don't have a checkpoint record in the end, and therefore, when the instance will be restarted, it has to start with the restart recovery operation before opening the service to the clients.

Note that the instance node in the Object Explorer window has a red spot indicating that the server has stopped working.

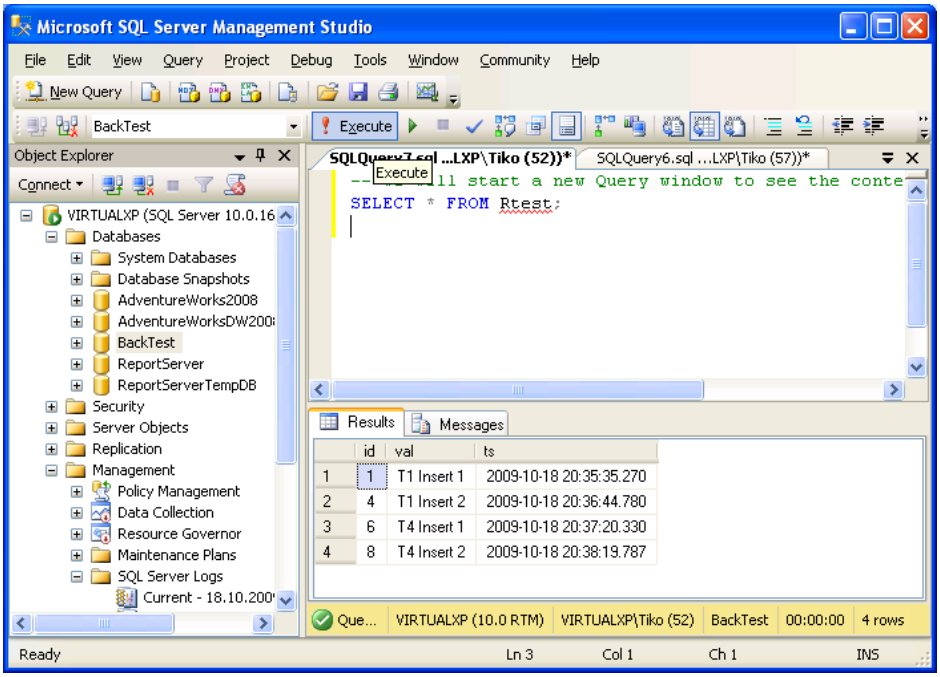


We will now restart the server selecting the instance node by right mouse button and selecting the "Start" option. We will then wait until the red spot in the node turns green again, indicating that the recovery operation has finished.



-- and we will start a new Query window to see the contents of Rtest table

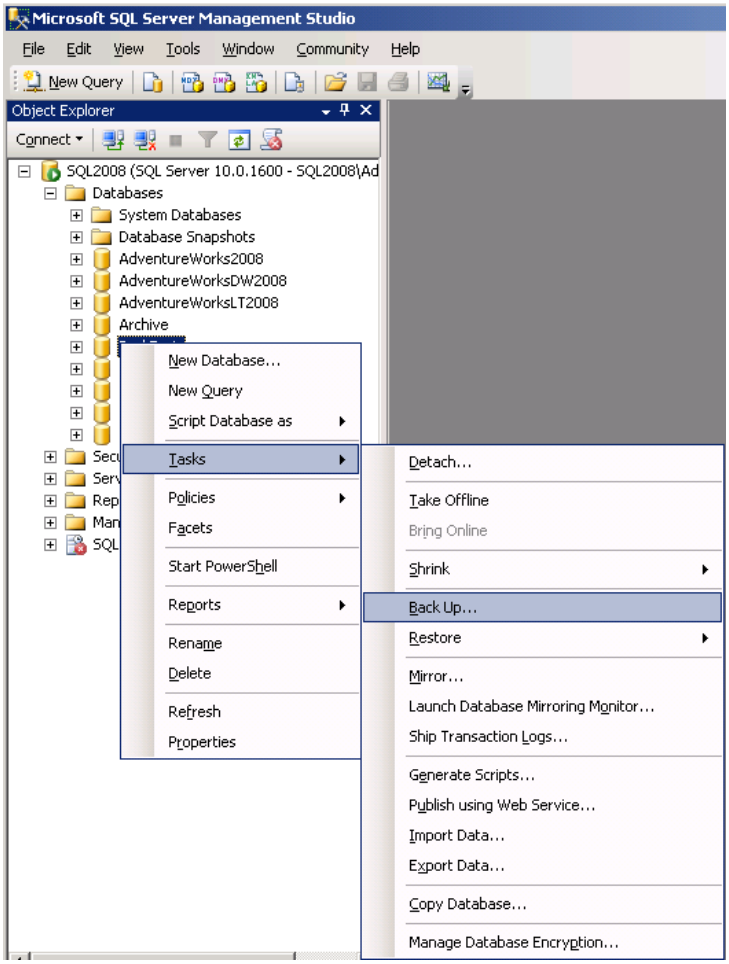
```
SELECT * FROM Rtest;
```



.. and we will see that only the results of the committed transactions T1 and T4 are found in the database, whereas the rows of transactions T2, T3, and T5 are rolled back.

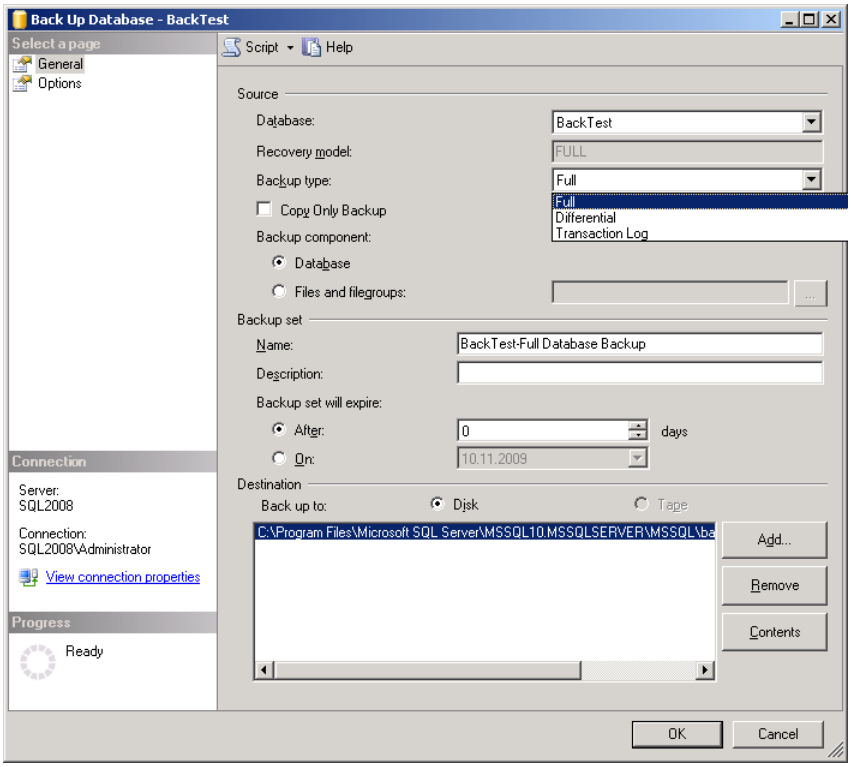
RCLabSS3 – Online Backup

SQL Server database backups can be done using the Transact-SQL BACKUP command of which the syntax, parameters, and options are explained in Books Online documentation [9].
An easier possibility is to use the GUI interface of SQL Server Management Studio selecting first the database by right mouse button from the Object Explorer pane, then "Tasks" and "Backup...".

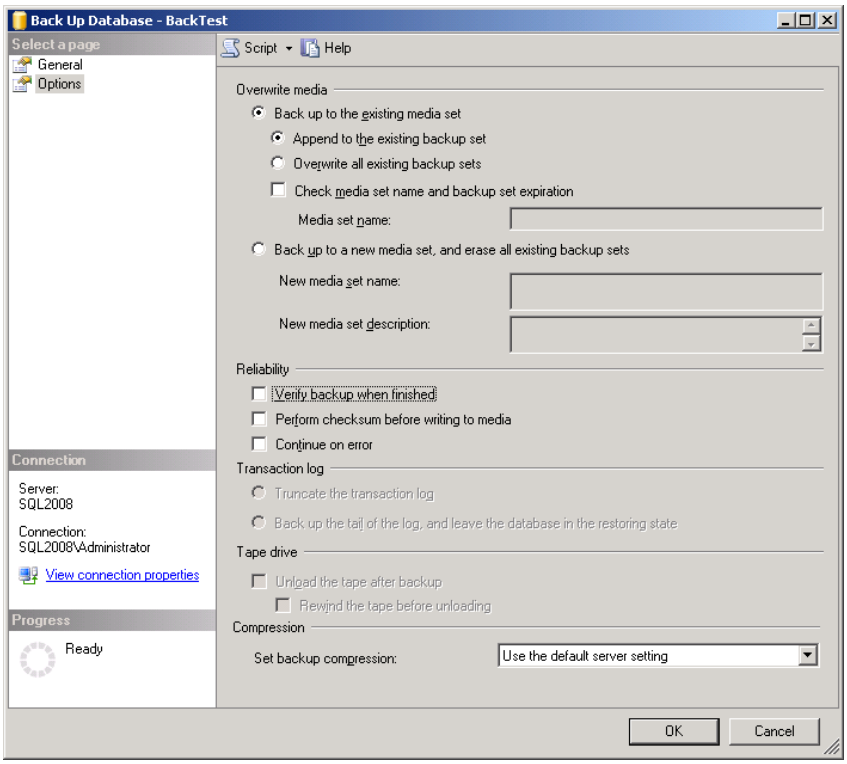


This brings us to the General page of Backup Database wizard, on which we can select
- the backup type : Full , Differential, or Transaction Log
- path for the backup file
and define name and description of the backup.

We can also set expiry time of the backup after which the backup can be overwritten.



On Option page we can fine tune the backup operation, for example if overwrite or append the backup file. Our snapshot shows the default options, but for backups of production databases it would be important to select at least the "Verify backup when finished".



The graphical user interface presents the major options to the user, but, in the end, our tool will generate and use the Transact-SQL BACKUP command for the operation when we press OK button.

Note: The knowledge of BACKUP commands is important, since backups should be automated operations, and usually be defined in Transact-SQL scripts. Also, it is always possible that our tool does not always generate proper commands.

Review Exercises:

1. Using SSMS, make a backup of the database BackTest, then on a Query session, first list all rows of the Rtest table of our previous lab RCLab1, and then delete all those rows.
   Using SSMS, restore the backup over the existing database BackTest, and using a new Query session, verify that you have got back the original contents of the table.

2. Using Books Online documentation, study the meanings of
   - "Copy Only Backup" on the General page, and
   - "Perform checksum before writing to media" on Options page.

RCLabSS4 - Backups and Roll-forward Recovery to a Point-in-Time

ISO SQL standard does not even mention the concept of database, and so it does not cover database administration operations such as start, shutdown, backup, restore, etc. However, SQL implementations include quite similar looking SQL command extensions for these purposes. These operations are also available as wizards in DBA tools, such as SQL Server Management Studio, showing the major options for the user and using appropriate SQL commands to communicate with the server.

In this lab we will play with a sequence of transactions inserting rows in our Backtest database making BACKUP of the database and some backups of the transaction log file. We will take note of the timestamps of the inserted rows, and finally we will try to RESTORE the database to selected point-in-time based on these backups.

You may use the database BackTest of RCLab1, or drop it and create a new one as follows

Note: In the following the home folder of our SQL Server instance is
'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL',
but you may use any folder you consider appropriate.

```
USE Master;
DROP DATABASE BackTest;
GO

CREATE DATABASE BackTest ON PRIMARY
 ( NAME = 'BackTest', FILENAME =
  'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\BackTest.mdf',
 SIZE = 2048KB , FILEGROWTH = 1024KB )
 LOG ON ( NAME = 'BackTest_log', FILENAME =
 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\BackTest_log.ldf',
 SIZE = 1024KB , FILEGROWTH = 10%) ;
GO
```

Note: For our test purposes we can have the data file and transaction log on the same disc, but for production environment, the transaction log should be allocated on a separate, fast disc.

-- Now we will configure the database to use the full recovery model (for explanations see [9])
-- and create there an easy-to-use, minimalistic table T for our test, and we don't try to save space

```
ALTER DATABASE BackTest SET RECOVERY FULL ;
EXEC SP_DBOPTION BackTest,'trunc. log on chkpt','FALSE'
GO
--
USE BackTest;
IF EXISTS(SELECT name FROM sysobjects WHERE  name = N'T' AND type = 'U')
    DROP TABLE T
GO
CREATE TABLE T (
id  INT PRIMARY KEY IDENTITY(1, 1) NOT NULL,
var CHAR(4000) NOT NULL DEFAULT 'some text',
ts  DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
)
GO
```

-- In table T we will insert 100 rows as follows:

```
USE BackTest;
```

```
SET NOCOUNT ON;
DECLARE @counter int;
SET @counter=1;
WHILE (@counter <= 100)
BEGIN
    BEGIN TRANSACTION;
    INSERT INTO T DEFAULT VALUES;
    COMMIT WORK;
    SET @counter=@counter+1;
END;
GO
```

-- Using the DBCC LOGINFO command, we get some statistics of the VLF files as follows:

```
DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ ---------
2       253952              8192                 22          2           64     0
2       253952              262144               23          2           64     0
2       253952              516096               24          2           64     0
2       278528              770048               0           0           0      0

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

A `FileId` is the logical file number of the transaction log in the set of files of the database. The `FSeqNo` indicates the logical seq# of the VLF. A value 2 of `Status` column indicates a filled or an active VLF, which has not been backed up, and value 0 indicates an unused or reusable VLF.

We will now backup databases Master, BackTest, and Msdb

```
USE Master;
BACKUP DATABASE Master TO DISK =
 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\backup\master.bak'
   WITH INIT, NAME = 'Master backup';
BACKUP DATABASE BackTest to disk =
 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\backup\BackTest.bak'
   WITH INIT, NAME = 'BackTest backup';
BACKUP DATABASE Msdb TO DISK =
 'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\backup\Msdb.bak'
   WITH INIT, NAME = 'Msdb backup';
GO

Processed 376 pages for database 'Master', file 'master' on file 1.
Processed 4 pages for database 'Master', file 'mastlog' on file 1.
BACKUP DATABASE successfully processed 380 pages in 0.653 seconds (4.540
MB/sec).
Processed 208 pages for database 'BackTest', file 'BackTest' on file 1.
Processed 2 pages for database 'BackTest', file 'BackTest_log' on file 1.
BACKUP DATABASE successfully processed 210 pages in 0.373 seconds (4.384
MB/sec).
Processed 1288 pages for database 'Msdb', file 'MSDBData' on file 1.
Processed 5 pages for database 'Msdb', file 'MSDBLog' on file 1.
BACKUP DATABASE successfully processed 1293 pages in 1.357 seconds (7.441
MB/sec).
```

-- We continue to insert 100 more rows as follows

```
USE BackTest;
SET NOCOUNT ON;
DECLARE @counter int;
```

```
SET @counter=1;
WHILE (@counter <= 100)
BEGIN
    BEGIN TRANSACTION;
    INSERT INTO T DEFAULT VALUES;
    COMMIT WORK;
    SET @counter=@counter+1;
END;
GO
```

```
DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ --------
2       253952              8192                 26          2           128    0
2       253952              262144               23          0           64     0
2       253952              516096               24          2           64     0
2       278528              770048               25          2           64     0

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

-- Now we will backup the transaction log and Msdb

```
USE Master;
BACKUP LOG BackTest to disk =
'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\backup\BackTest_log.bak'
  WITH INIT, NAME = 'BackTest log backup';
BACKUP DATABASE Msdb TO DISK =
'C:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\backup\Msdb.bak'
  WITH INIT, NAME = 'Msdb backup';
GO

Processed 62 pages for database 'BackTest', file 'BackTest_log' on file 1.
BACKUP LOG successfully processed 62 pages in 0.142 seconds (3.400 MB/sec).
Processed 1288 pages for database 'Msdb', file 'MSDBData' on file 1.
Processed 5 pages for database 'Msdb', file 'MSDBLog' on file 1.
BACKUP DATABASE successfully processed 1293 pages in 1.017 seconds (9.928
MB/sec).
```

use BackTest

```
DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ ---------
------------------
2       253952              8192                 213         2           128    0
2       262144              262144               212         2           64     0
2       262144              524288               210         0           64
185000000012800007
2       262144              786432               211         0           64
186000000014400008

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

-- We will insert 100 more rows in table T

```
USE BackTest;
SET NOCOUNT ON;
DECLARE @counter int;
SET @counter=1;
WHILE (@counter <= 100)
BEGIN
    BEGIN TRANSACTION;
    INSERT INTO T DEFAULT VALUES;
    COMMIT WORK;
    SET @counter=@counter+1;
```

```
END;

DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ ---------
------------------------------
2       253952              8192                 26          2           128    0
2       253952              262144               27          2           128    0
2       253952              516096               28          2           128    0
2       278528              770048               25          0           64     0

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

-- and we will backup the transaction log with Msdb again

```
USE Master;
BACKUP LOG BackTest to disk =
'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\backup\BackTest_log.bak'
  WITH NOINIT, NAME = 'BackTest log backup';
BACKUP DATABASE Msdb TO DISK =
'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\backup\Msdb.bak'
  WITH INIT, NAME = 'Msdb backup';
GO

Processed 62 pages for database 'BackTest', file 'BackTest_log' on file 2.
BACKUP LOG successfully processed 62 pages in 0.130 seconds (3.710 MB/sec).
Processed 1288 pages for database 'Msdb', file 'MSDBData' on file 1.
Processed 5 pages for database 'Msdb', file 'MSDBLog' on file 1.
BACKUP DATABASE successfully processed 1293 pages in 1.047 seconds (9.644
MB/sec).

DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ ---------
-----------------
2       253952              8192                 213         2           128    0
2       262144              262144               212         2           64     0
2       262144              524288               210         0           64
185000000012800007
2       262144              786432               211         0           64
186000000014400008

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

-- We continue to insert 20 more rows in table T

```
USE BackTest;
SET NOCOUNT ON;
DECLARE @counter int,
        @maxid int;
SET @counter=1;
WHILE (@counter <= 20)
BEGIN
    BEGIN TRANSACTION;
    INSERT INTO T DEFAULT VALUES;
    COMMIT WORK;
    WAITFOR DELAY '00:00:01';      <<- using this we get different timestamps on the inserted rows
    SET @counter=@counter+1;
END;
```

-- Let's see the 10 last rows in the table

```
SELECT id, ts FROM T WHERE id > (SELECT MAX(id)-10 FROM T);
id          ts
----------- -----------------------
311         2009-11-08 23:02:36.023
312         2009-11-08 23:02:37.353
313         2009-11-08 23:02:38.357
314         2009-11-08 23:02:39.357    <<- We want to recover up to this transaction
315         2009-11-08 23:02:40.360
316         2009-11-08 23:02:41.360
317         2009-11-08 23:02:42.680
318         2009-11-08 23:02:43.683
319         2009-11-08 23:02:44.733
320         2009-11-08 23:02:45.737

DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ ---------
------------------------------
2       253952              8192                 26          0           128    0
2       253952              262144               27          0           128    0
2       253952              516096               28          2           128    0
2       278528              770048               25          0           64     0

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

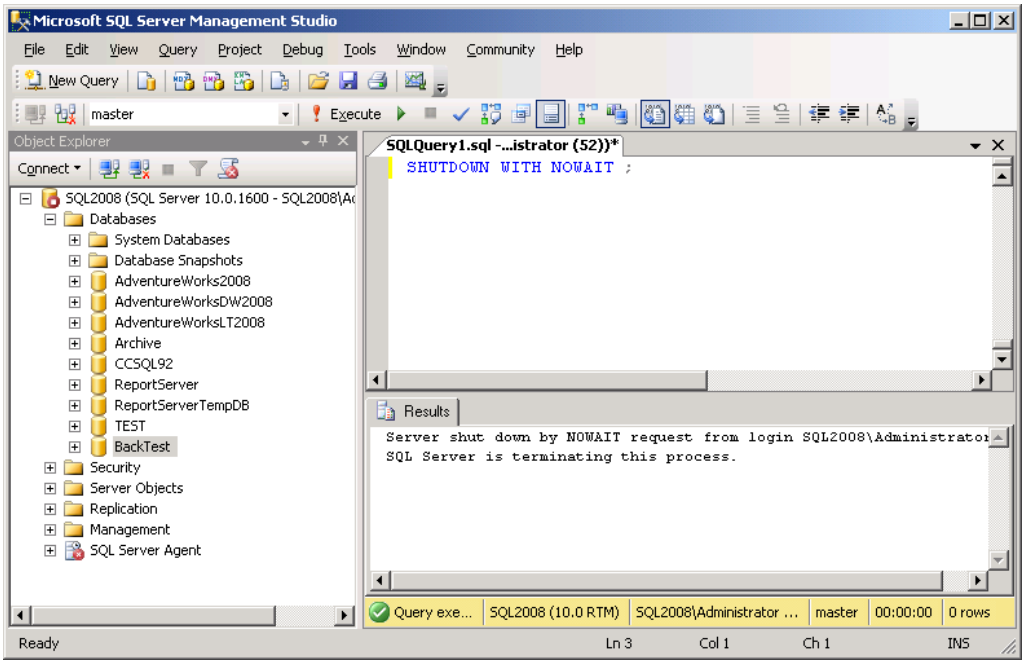
-- Let's save the current log in a separate backup file

```
USE Master;
BACKUP LOG BackTest to disk =
'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\backup\BackTest_cur_log.bak'
  WITH INIT, NO_TRUNCATE, NORECOVERY, NAME = 'BackTest current log backup';
BACKUP DATABASE Msdb TO DISK =
'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\backup\Msdb.bak'
  WITH INIT, NAME = 'Msdb backup';
GO

Processed 15 pages for database 'BackTest', file 'BackTest_log' on file 1.
BACKUP LOG successfully processed 15 pages in 0.060 seconds (1.863 MB/sec).
Processed 1288 pages for database 'Msdb', file 'MSDBData' on file 1.
Processed 3 pages for database 'Msdb', file 'MSDBLog' on file 1.
BACKUP DATABASE successfully processed 1291 pages in 1.222 seconds (8.253
MB/sec).
```

Next we will simulate the system crash by the following command which shuts down the server instance without ARIES checkpoints of the databases

```
SHUTDOWN WITH NOWAIT ;
```

and when the instance node in Object Explorer pane get red mark on it, the instance has stopped.



Note: After the "crash" of the server instance has been sorted out, the instance should first be started in single-user mode using the startup paramter **–m** so that applications cannot access the instance.

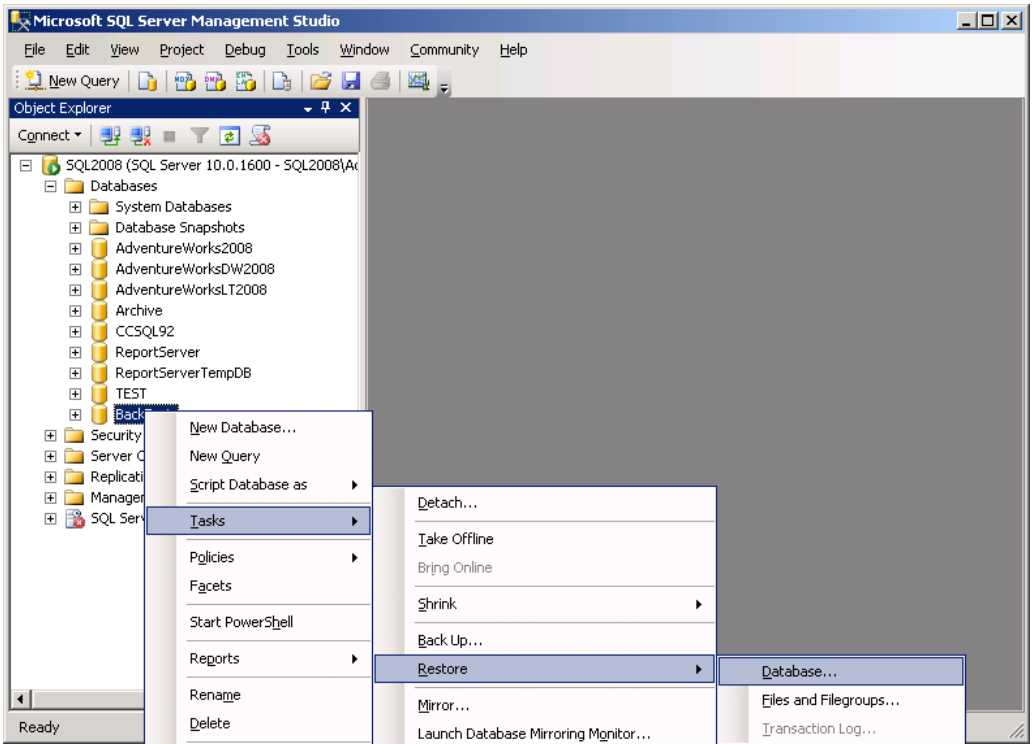
If Master database has been lost, it must be restored first from its last full backup using SQLCMD session. For more details on this, see Books Online [9].

If Msdb database has been lost, it must be restored first from its last full backup using SQLCMD session or SSMS Query window using commands

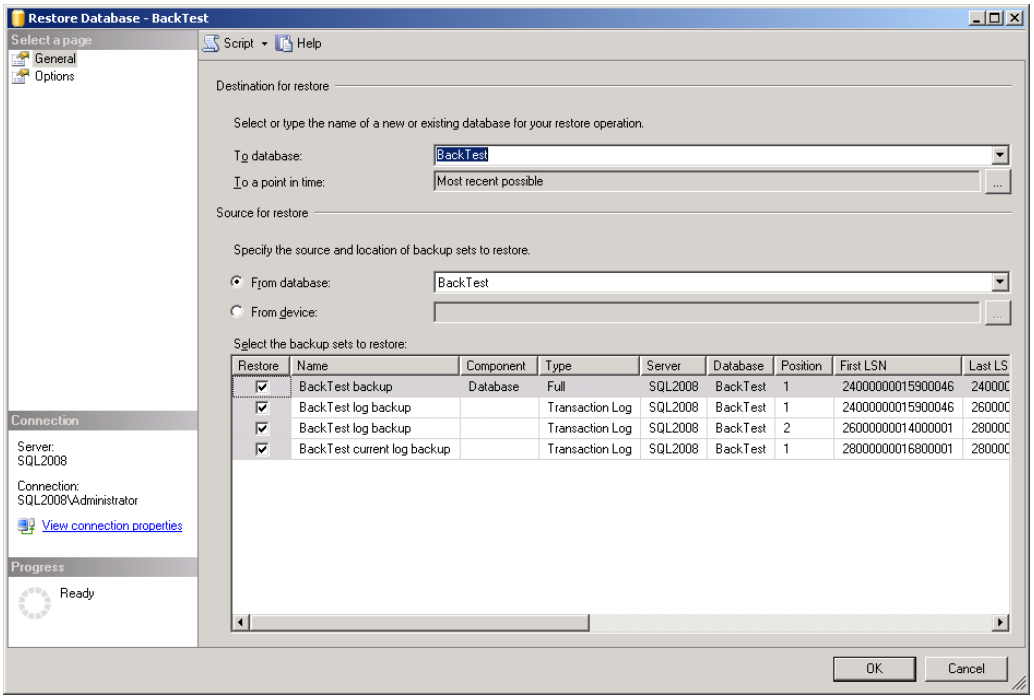
```
USE master;
RESTORE DATABASE Msdb FROM DISK = 'C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\backup\msdb.bak'
    WITH RECOVERY;
```

To restore our application database BackTest, we could use Transact-SQL RESTORE commands, but we need to know its backup files and the logical backup file numbers of the transaction logs there. Since Msdb database contains the backup histories of the databases, it is much more reliable to use SSMS for restoring our database. We start selecting our database with the right mouse button, then

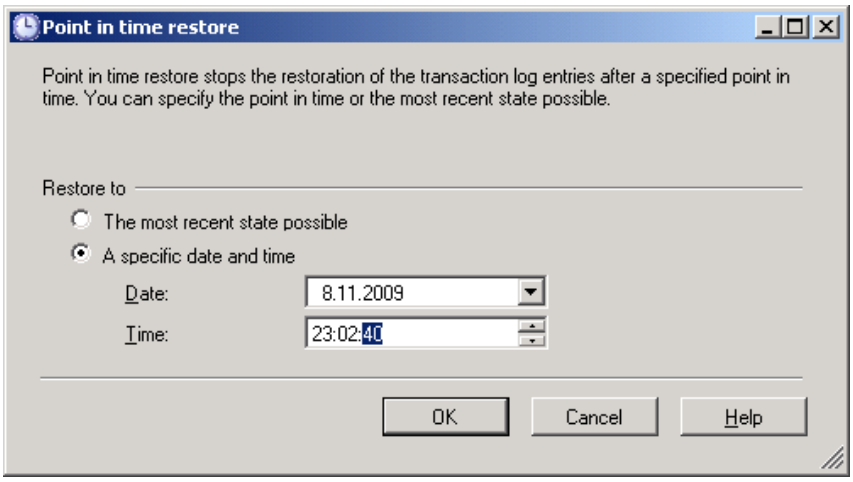
Tasks – Restore – Database…



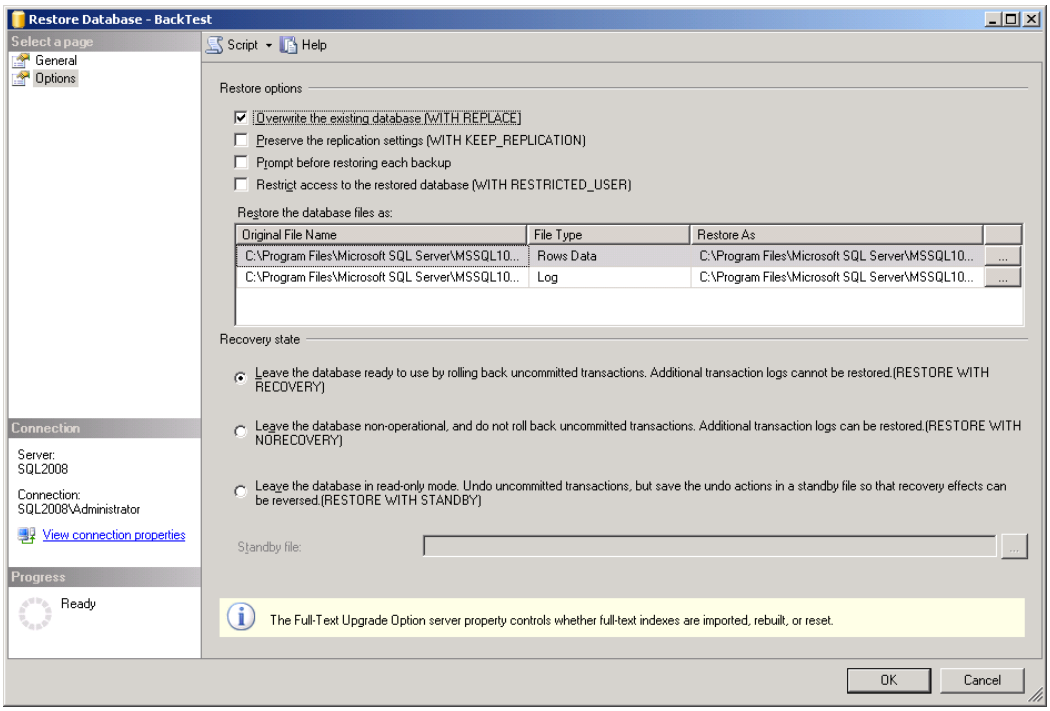
and on the General form we will see catalog of the backups pre-selected for us



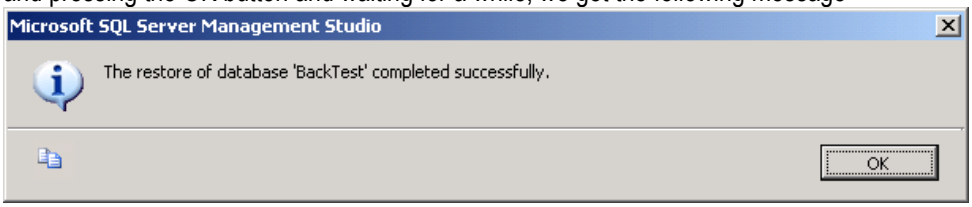
We select the selection button at the end of the field “To a point in time”, and we proceed to “Point in time restore” form where we enter the date and time of our wish of the first moment after our last acceptable transaction should have managed to COMMIT.



on the Options form we select “Overwrite the existing database (WITH REPLACE)” option and for the Recovery state the “RESTORE WITH RECOVERY”



and pressing the OK button and waiting for a while, we get the following message



Now we can check if we succeeded in restoring the database to that selected point in time.

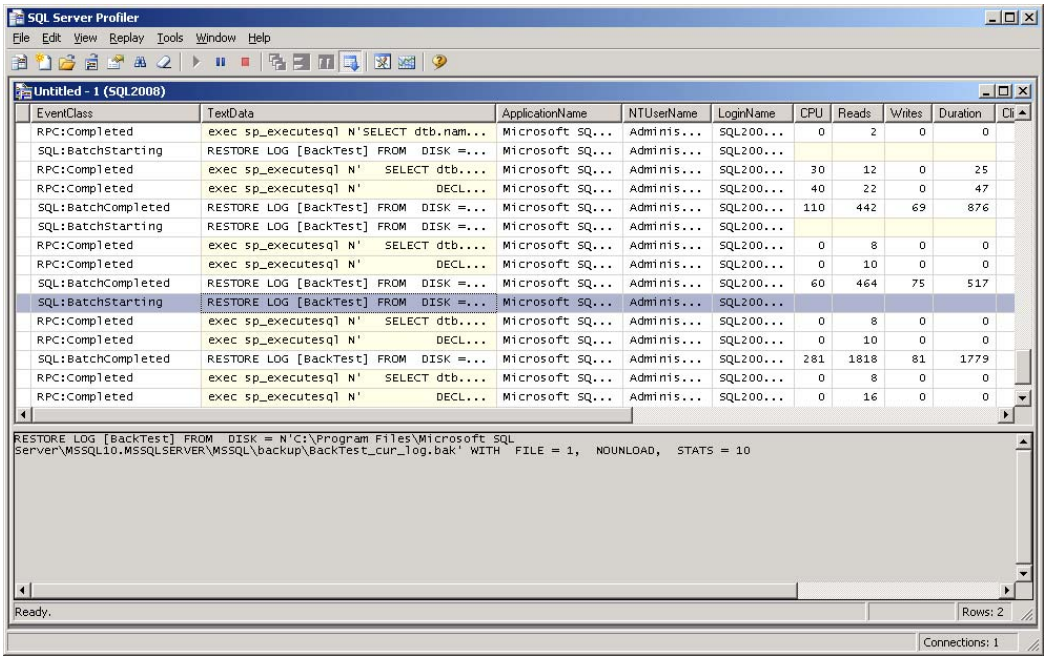
We open a new Query window and enter commands to see the 10 last rows in the table

```
USE Backtest
SELECT id, ts FROM T WHERE id > (SELECT MAX(id)-10 FROM T);

id          ts
----------- -----------------------
311         2009-11-08 23:02:36.023
312         2009-11-08 23:02:37.353
313         2009-11-08 23:02:38.357
314         2009-11-08 23:02:39.357
315         2009-11-08 23:02:40.360
316         2009-11-08 23:02:41.360
317         2009-11-08 23:02:42.680
318         2009-11-08 23:02:43.683
319         2009-11-08 23:02:44.733
320         2009-11-08 23:02:45.737

(10 row(s) affected)
```

… and we have actually restored the database up to the last committed transaction – but not to the point in time we selected. Well, this is a known bug in previous the versions of SSMS, but the developers have not fixed it at least in the 2008 version that we used:

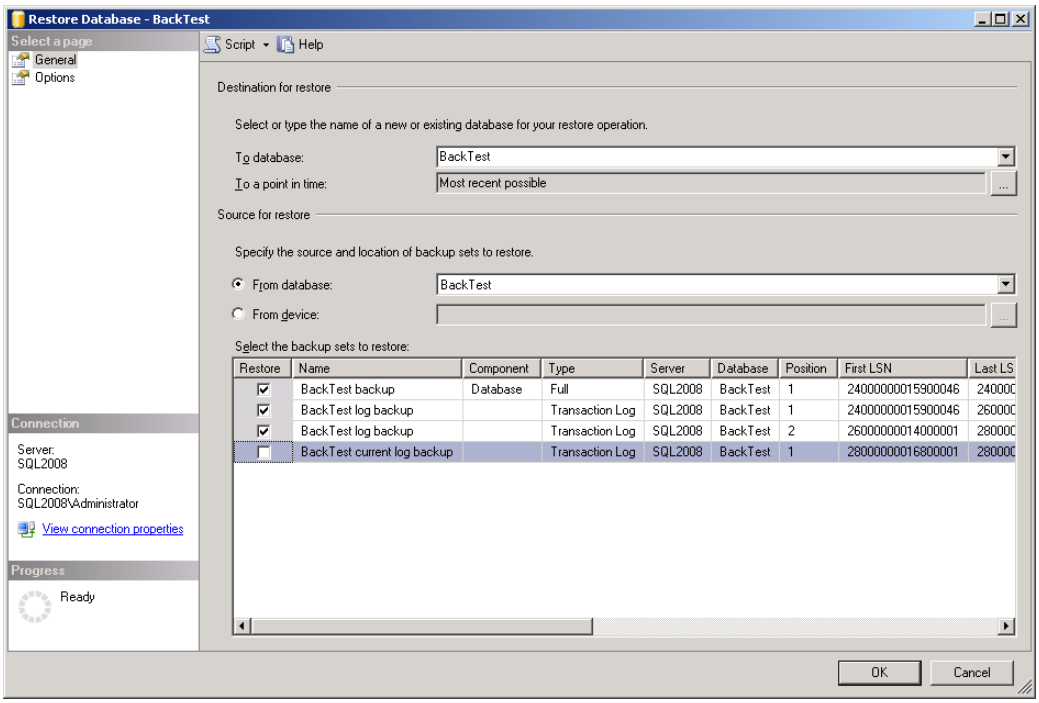
SELECT @@version
Microsoft SQL Server 2008 (RTM) - 10.0.1600.22 (Intel X86)
Jul 9 2008 14:43:34
Copyright (c) 1988-2008 Microsoft Corporation
Enterprise Edition on Windows NT 5.2 <X86> (Build 3790: Service Pack 2)

By tracing the last RESTORE LOG step using the Profiler tool, we can see that SSMS does not generate the needed STOPAT parameter for the point-in-time.

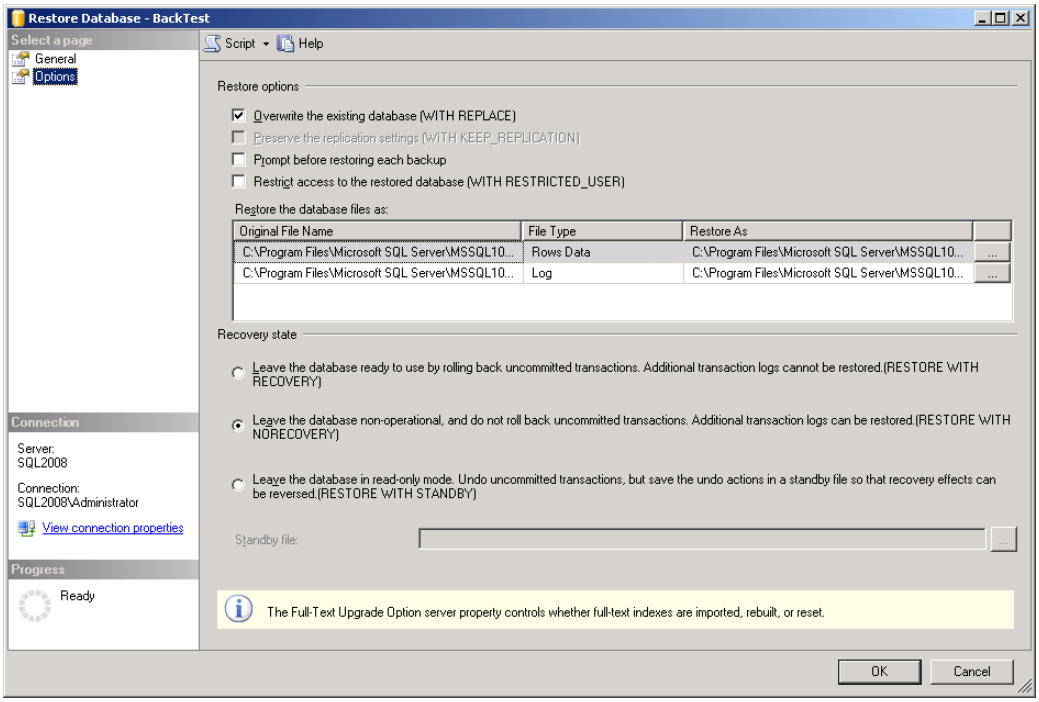


To fix the problem, we need to use the Transact-SQL RESTORE command with the STOPAT parameter by ourselves. However, we want to use the automatic Restore services of SSMS and the Msdb database as far as we can to avoid failures in restoring.

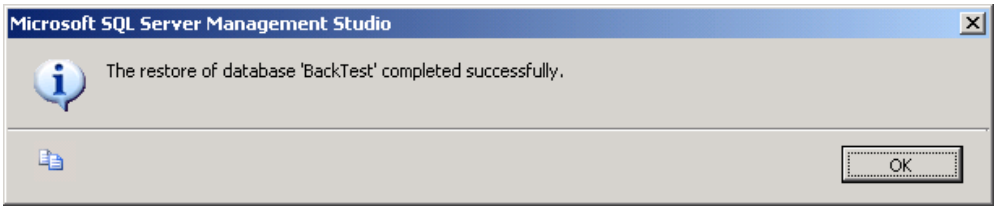
If we look at SSMS – BackTest – Task – Restore - Restore Database – General form page, and the start date column of the backup sets, we can see that the point-in-time we want to use is covered in the current log backup, so we will first use the SSMS Restore leaving the current log backup out of the restore



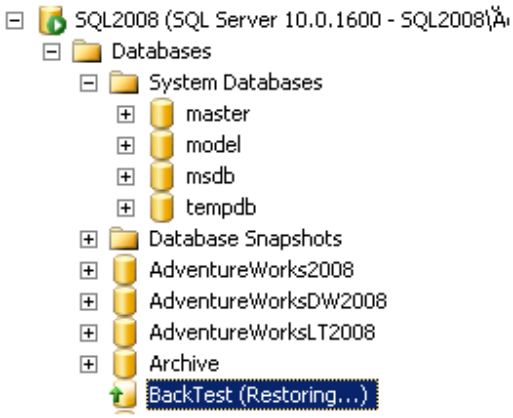
and select Recovery state as WITH NORECOVERY



pressing the OK button, and after the message



we will see that the Backtest is not yet available, but in a Restoring state:




```
USE master
RESTORE LOG [BackTest] FROM  DISK =
'C:\Program Files\Microsoft SQL
Server\MSSQL10.MSSQLSERVER\MSSQL\backup\BackTest_cur_log.bak'
```

```
WITH  FILE = 1, RECOVERY, STOPAT = '2009-11-08 23:02:40'
GO

Processed 0 pages for database 'BackTest', file 'BackTest' on file 1.
Processed 15 pages for database 'BackTest', file 'BackTest_log' on file 1.
RESTORE LOG successfully processed 15 pages in 0.028 seconds (3.993 MB/sec).

USE BackTest
SELECT id,ts FROM T WHERE id = (SELECT MAX(id) FROM T)

id          ts
----------- -----------------------
314         2009-11-08 23:02:39.357

(1 row(s) affected)
```

and this is just the row of the last committed transaction before the select STOPAT time.
We may be also interested in seeing the current status of the VLF files

```
DBCC LOGINFO
FileId  FileSize            StartOffset          FSeqNo      Status      Parity CreateLSN
------- ------------------- -------------------- ----------- ----------- ------ ---------
2       253952              8192                 26          0           128    0
2       253952              262144               27          0           128    0
2       253952              516096               28          2           128    0
2       278528              770048               25          0           64     0

(4 row(s) affected)

DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
```

We could now change the instance into a multi-user mode and so available to applications.

Review questions and exercises

Why we should always take backup of the Msdb database as the last step after database backups or transaction log backups, and why with the INIT option?

Why does SQL Server pack the current transaction log with the full backup of the database?

Advanced exercises

Using SSMS, generate a maintenance plan job for the BackTest database so that database backup will take place once a day, and transaction log backups every hour. How do you include Msdb backup as the last step of every backup run?

If the database file BackTest.mdf or the transaction log file is lost (test by deleting it) and the instance is then restarted, the database can be seen in the Object Explorer pane of SSMS, but its contents cannot be explored in SSMS, and SSMS does not provide you the RESTORE opportunity. What possibilities do you have for recovering the database if you still have the backup files of the database and transaction log history?

Consider the case that you have lost Master.mdf or its transaction log. What options do you have to get your instance up and running? (Search the article “Recover the master database in SQL Server” in Web)

RCLabDB2 – DB2 Express-C Recovery Labs

Now it is time to look for more challenging systems. We start with the DB2 Express-C (see [6]) in our VMware – Ubuntu platform. Almost everything of our RCLabSS can be done using the DB2 SQL commands in Command Line Processor (CLP) session. However, for a novice user, the Control Center is a life-saver presenting steps and available options in proper order with explanations, and providing also the generated SQL commands.
Instead of presenting every step RCLabSS here, we present just some sample screenshots with explanations, trusting that you can sort out the detailed steps yourself.

RCLabDB23 - Online Backup

The default mode of DB2 is offline backup, but in the modern 24x7 environment we should be able to take online backups like in the following example for the SAMPLE database which comes with DB2

```
BACKUP DATABASE sample ONLINE TO <destination dir>
INCLUDE LOGS
```

which provides about the same operations as SQL Server's online backup.
The default logging mode of a DB2 database is CIRCULAR, which means that when the log files are full, DB2 starts to overwrite on the log files starting from the first log, and destroying the transaction history. For online backups, the database needs to be configured in ARCHIVE logging mode. This can be done from the Control Center selecting first "Configure Database Logging…" and then "Archive Logging" as follows

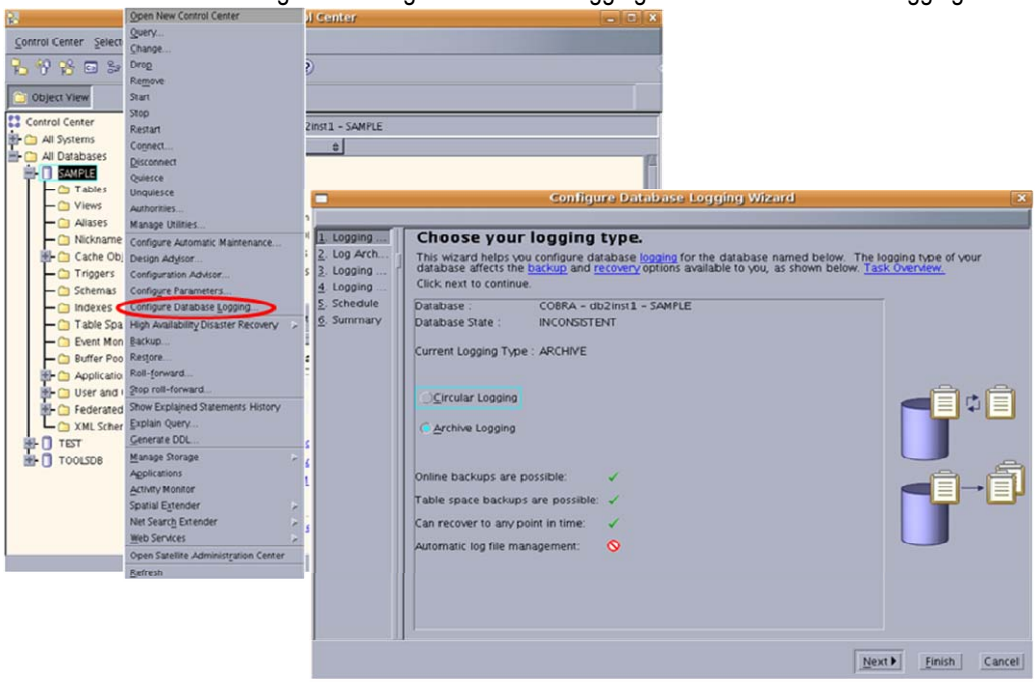
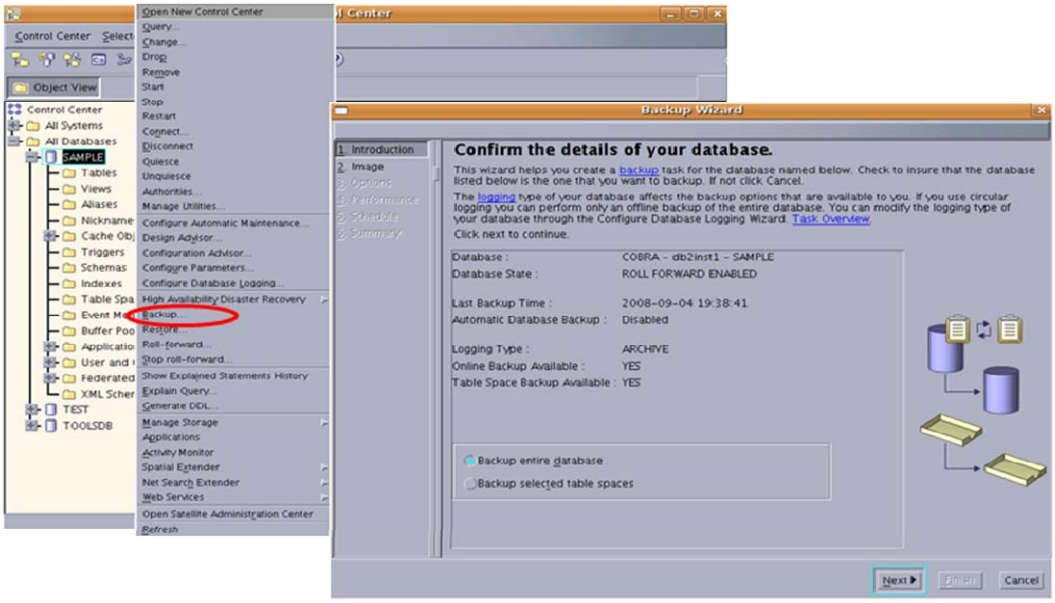
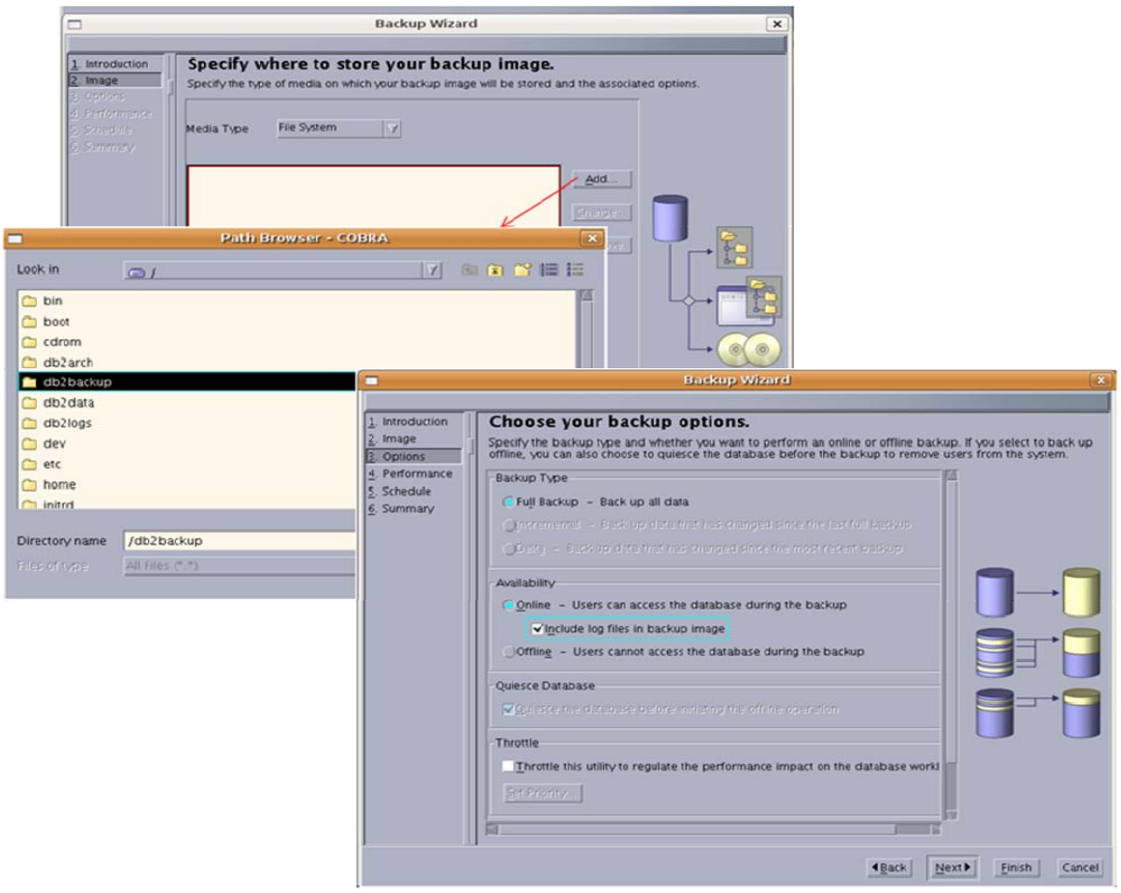


Figure n. Configuring database to ARCHIVE mode

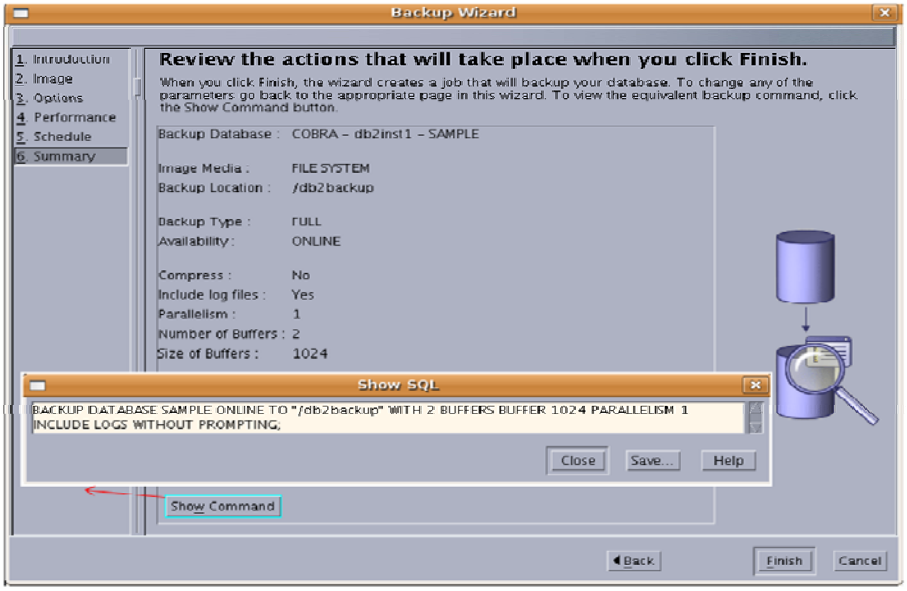
We can now start to define the online backup for the SAMPLE database in the Control Center selecting "Backup…"



then defining the backup directory and including the transaction logs as follows



After performance and scheduling options, we can review the generated BACKUP command and save it to be used in backup scripts or pressing the Finish button starting the defined backup process immediately



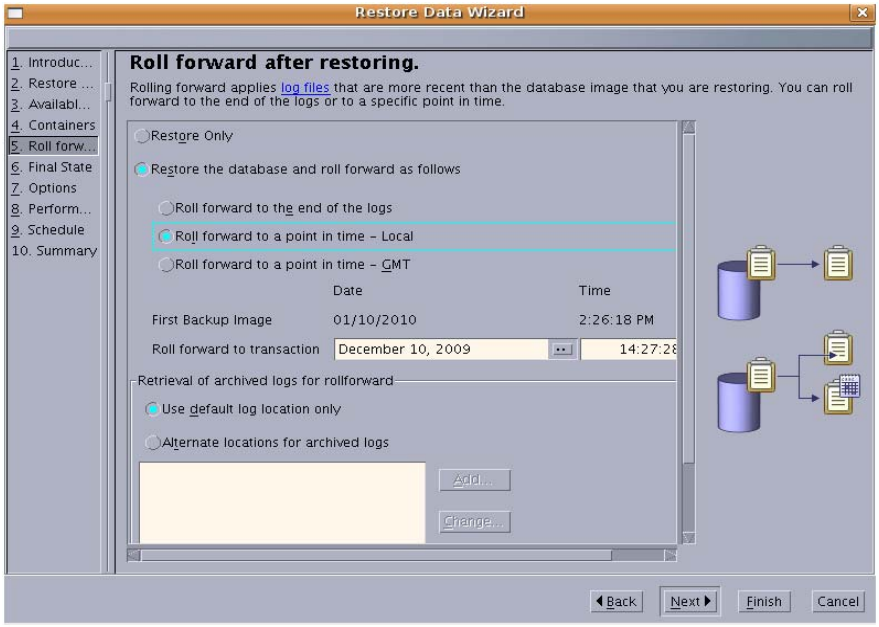
RCLabDB24 - Point In Time (PIT) Recovery

Without going into details, we suggest that you try to apply the Point-in-time recovery procedure we used in RCLabSS4 for DB2 (without the bug fixing procedure ☺)

First, generate a sequence of transactions with timestamp information like we did using SQL Server. Take a note of a timestamp to the level of which you want the database to be recovered.

Then backup the transaction logs after the full backup we just did.

For the recovery, you should start in the Control Center selecting “Restore..” and selecting the proper backup from the list presented by the Restore Wizard and selecting the point-in-time as follows



Finally, verify that you have got the latest committed transactions at the requested PIT.

Note:

The Control Center may be replaced by IBM Data Studio for DB2 in the future. The interface for defining Configuration, Backup and Restore operations is approximately the same as in the Control Center.

RCLabOra – Oracle XE Recovery Labs

For this part of the RC labs should apply the RCLabSS labs for Oracle XE database (see [8]). We plan to write here more notes on the Oracle backup and recovery later, but currently we only provide the virtual computer, and we refer to Oracle tutorials available at http://otn.oracle.com.

References and Links

[1] J. Gray, A. Reuter, "Transaction Processing: Concepts and Techniques", Morgan Kaufmann, 1993

[2] C. Mohan, Repeating History Beyond ARIES, VLDB99 paper, 1999

[3] Craig S. Mullins, Database Administration, The complete Guide to Practices and Procedures, Addison-Wesley, 2002

[4] Thomas Connolly and Carolyn Begg, Database Systems, 5th ed. 2009, Addison-Wesley

[5] Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, 3rd ed. McGraw-Hill, 2003

[6] Introduction to DB2 9 database recovery tutorial http://www.ibm.com/developerworks/edu/dm-dw-dm-0708barsagade-i.html

[7] Oracle database recovery manager RMAN tutorial http://www.oracle.com/technology/deploy/availability/htdocs/rman_overview.htm?_template=/ocom/print

[8] Oracle® Database Backup and Recovery User's Guide, 11g Release 1 (11.1)

[9] Microsoft, SQL Server 2008 Books Online

[10] Tim Mortimer & al, Using ADSM to Back Up Databases, http://www.redbooks.ibm.com

[11] C. J. Date, An Introduction to Database Systems, 8th ed., Addison-Wesley, 2004