

苏州大学实验报告

院、系	计算机学院	年级专业	19 计科图灵	姓名	张昊	学号	1927405160
课程名称	数据库课程实践					成绩	90
指导教师	赵朋朋	同组实验者	无	实验日期	2021 年 4 月 16 日		

实验名称

SQL 语言实验 12-14

试验十二 恢复技术

一、实验目的

- 1 掌握数据库的备份及恢复的方法。
- 2 了解备份方案的设定

二、实验内容

(一) 完全备份的建立与恢复

1 建立完全备份

```
USE school
GO
BACKUP DATABASE school TO DISK='C:\schooldata.bak'
```

2 查看备份文件中的信息

```
RESTORE FILELISTONLY FROM DISK='c:\schooldata.bak'
RESTORE HEADERONLY FROM DISK='c:\schooldata.bak'
```

3 恢复完全备份

1) 先删除数据库 School

```
USE Master
GO
DROP DATABASE school
```

2) 然后恢复.

```
RESTORE DATABASE school from DISK='c:\schooldata.bak'
```

3): 查看 school 的 student 中的数据

(二) 建立差异备份

1 建立备份

1) 制作数据文件备份 schoolDiff.bak

2) 把学号 7001, 姓名: 王海, 性别: 男, 年龄为 23 的学生加入 student

3) 制作 school 的差异备份, 存入 schoolDiff.bak

```
BACKUP DATABASE school TO DISK='schoolDiff.bak' WITH DIFFERENTIAL
```

4) 把学号 7002, 姓名: 赵燕, 性别: 女, 年龄为 22 的学生加入 student

5) 制作 school 的差异备份, 存入 schoolDiff.bak

```
BACKUP DATABASE school TO DISK='schoolDiff.bak' WITH DIFFERENTIAL
```

2 查看备份文件 schoolDiff.bak 中的信息

3 删除 school 数据库

4 恢复数据库 school 到第 2 步状态

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak' WITH file=1 NORECOVERY
RESTORE DATABASE school from DISK='c:\schoolDiff.bak' WITH file=2
```

```
Select * from student
```

观察 student 数据

5 恢复数据库 school 到最新状态

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=1 NORECOVERY
```

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=3
```

```
Select * from student
```

观察 student 数据

思考: 如果仅执行下述恢复语句,能查看 student 的数据吗?

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=1 NORECOVERY
```

```
Select * from student
```

(三) 利用日志备份

1 设置故障还原模型为: 完全

2 建立备份

1) 制作数据文件备份 schooldata1.bak

2) 把学号 7003, 姓名: 王江, 性别: 男, 年龄为 23 的学生加入 student

3) 制作日志备份存入 schoollog.bak

4) 把学号 7004, 姓名: 赵兰, 性别: 女, 年龄为 22 的学生加入 student

5) 制作日志备份存入 schoollog.bak

3 观察 schoollog 中的信息

4 删除 school 数据库

5 利用 schooldata1.bak 及 schoollog.bak 恢复数据库 school 到最新状态

(四) 使用企业管理器练习备份调度策略

1 对数据库 school 每天上午 8 时进行一次数据库完全备份

2 对数据库 school 的每隔 1 分钟备份进行一次差异备份。

3 手工启动两个备份作业

4 删除 school 数据库

5 利用 1, 2 步的备份进行 school 的恢复。

思考: 如何把备份文件备份到另外一台计算机上。

(五) 使用企业管理器练习数据库的分离及附加

(六) 如何清除日志文件。

(七) 使用企业管理器练习数据库的压缩

(八) 把 school 备份到其他计算机上。

[加上 如果没有 norecover,发生的情况]

[企业管理期恢复 如何实现 norecover]

三、实验结果

(一) 完全备份的建立与恢复

建立完全备份:

```
USE school
```

```
GO
```

```
BACKUP DATABASE school TO DISK='C:\schooldata.bak'
```



已为数据库 'school', 文件 'school' (位于文件 1 上)处理了 176 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 6 页。
BACKUP DATABASE 成功处理了 182 页, 花费 0.212 秒 (7.030 MB/秒)。

查看备份文件中的信息：

```
RESTORE FILELISTONLY FROM DISK='c:\schooldata.bak'  
RESTORE HEADERONLY FROM DISK='c:\schooldata.bak'
```

LogicalName	PhysicalName	Ty...	FileGroupName	Size	MaxSize	Field	CreateLSN	DropLSN	Uniqueld
1 school	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSS...	D	PRIMARY	2293760	35184372080640	1	0	0	1D48052E-2825-4DCD-8C1E-9FEDC3FBF
2 school_log	C:\Program Files\Microsoft SQL Server\MSSQL.1\MSS...	L	NULL	573440	2199023255552	2	0	0	1B6FBEBE-2821-47FD-B75E-536930BAC

BackupName	BackupDescription	BackupTy...	ExpirationDate	Compressed	Position	DeviceTy...	UserName	ServerName	DatabaseName	DatabaseVersion	DatabaseCreationDx
1 NULL	NULL	1	NULL	0	1	2	LAB-243\Administrator	LAB-243	school	611	2021-04-16 08:25:4

查询已成功执行。

(local) (9.0 RTM) LAB-243\Administrator (54) school 00:00:00 3 行

恢复完全备份：

先删除数据库 School

USE Master
GO
DROP DATABASE school

命令已成功完成。

然后恢复。

RESTORE DATABASE school from DISK='c:\schooldata.bak'

已为数据库 'school', 文件 'school' (位于文件 1 上)处理了 176 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 6 页。
RESTORE DATABASE 成功处理了 182 页, 花费 0.278 秒 (5.361 MB/秒)。

查看 school 的 student 中的数据

USE School
GO
SELECT * FROM Student

Sno	Sname	Ssex	Sage	Sdept
1 0001	周志林	男	25	SX
2 0002	李文庆	男	23	JSJ
3 0003	陈小明	男	20	SX
4 0004	杨秀红	女	21	JSJ
5 0009	钱明明	男	20	SX
6 0078	王振	男	21	JSJ
7 0081	刘亭	女	22	SX
8 0091	贺秋雪	女	20	SX
9 0092	赵三	男	22	SX
10 0701	刘欢	男	26	SX
11 8001	张华	男	23	SX
12 8002	赵颖	女	21	SX
13 8003	钱凯	男	22	JSJ
14 8004	王华	男	21	SX
15 8005	张英	女	21	JSJ
16 8006	赵章	女	22	JSJ
17 8007	钱利	男	23	JSJ
18 8008	王铁	男	21	JSJ

(二) 建立差异备份

建立备份:

制作数据文件备份 schoolDiff.bak

```
BACKUP DATABASE school TO DISK='C:\schoolDiff.bak'
```

消息

已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 504 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 1 页。
BACKUP DATABASE 成功处理了 505 页, 花费 0.023 秒(171.259 MB/秒)。

完成时间: 2021-04-16T16:07:35.5102000+08:00

把学号 7001, 姓名: 王海, 性别: 男, 年龄为 23 的学生加入 student

```
INSERT INTO Student (Sno, Sname, Ssex, Sage)  
VALUES ('7001', '王海', '男', 23)
```

制作 school 的差异备份, 存入 schoolDiff.bak

```
BACKUP DATABASE school TO DISK='C:\schoolDiff.bak' WITH DIFFERENTIAL
```

消息

已为数据库 'school', 文件 'school_dat' (位于文件 2 上)处理了 112 页。
已为数据库 'school', 文件 'school_log' (位于文件 2 上)处理了 1 页。
BACKUP DATABASE WITH DIFFERENTIAL 成功处理了 113 页, 花费 0.016 秒(54.779 MB/秒)。

完成时间: 2021-04-16T16:10:37.7236041+08:00

把学号 7002, 姓名: 赵燕, 性别: 女, 年龄为 22 的学生加入 student

```
INSERT INTO Student (Sno, Sname, Ssex, Sage) VALUES ('7002', '赵燕', '女', 22)
```

消息

(1 行受影响)

完成时间: 2021-04-16T16:11:03.6893141+08:00

制作 school 的差异备份, 存入 schoolDiff.bak

```
BACKUP DATABASE school TO DISK='C:\schoolDiff.bak' WITH DIFFERENTIAL
```

消息

已为数据库 'school', 文件 'school_dat' (位于文件 3 上)处理了 112 页。
已为数据库 'school', 文件 'school_log' (位于文件 3 上)处理了 1 页。
BACKUP DATABASE WITH DIFFERENTIAL 成功处理了 113 页, 花费 0.020 秒(43.823 MB/秒)。

完成时间: 2021-04-16T16:11:34.3180995+08:00

查看备份文件 schoolDiff.bak 中的信息:

```
RESTORE FILELISTONLY FROM DISK='C:\schoolDiff.bak'  
RESTORE HEADERONLY FROM DISK='C:\schoolDiff.bak'
```

结果

消息

	LogicalName	PhysicalName	Type	FileGroupName	Size	MaxSize	FileId	CreateLSN	DropLSN	UniqueId
1	school_dat	C:\data\schooll.mdf	D	PRIMARY	8388608	8388608	1	0	0	88078EE3-1D06-4E30-B149-C0E57BAFE8EE
2	school_log	C:\data\schooll.ldf	L	NULL	2097152	3145728	2	0	0	F7F06D55-55FC-4207-64D3-1CB3D72EAE8C

BackupName	BackupDescription	BackupType	ExpirationDate	Compressed	Position	DeviceType	UserName	ServerName	DatabaseName
NULL	NULL	1	NULL	0	1	2	HONOR-LAPTOP\Holger	HONOR-LAPTOP	school
NULL	NULL	5	NULL	0	2	2	HONOR-LAPTOP\Holger	HONOR-LAPTOP	school
NULL	NULL	5	NULL	0	3	2	HONOR-LAPTOP\Holger	HONOR-LAPTOP	school

删除 school 数据库

```
USE Master
GO
DROP DATABASE school
```

100 %

消息

命令已成功完成。

完成时间: 2021-04-16T16:14:34.4533439+08:00

恢复数据库 school 到第 2 步状态

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=1, NORECOVERY
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=2
```

消息

已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 504 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 1 页。
RESTORE DATABASE 成功处理了 505 页, 花费 0.014 秒(281.354 MB/秒)。
已为数据库 'school', 文件 'school_dat' (位于文件 2 上)处理了 112 页。
已为数据库 'school', 文件 'school_log' (位于文件 2 上)处理了 1 页。
RESTORE DATABASE 成功处理了 113 页, 花费 0.007 秒(125.209 MB/秒)。

完成时间: 2021-04-16T16:21:42.9434753+08:00

观察 student 数据

```
Select * from student
```

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX
2	0002	李文庆	男	23	JSJ
3	7001	王海	男	23	JSJ

5 恢复数据库 school 到最新状态

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=1, NORECOVERY
RESTORE DATABASE school from DISK='c:\schoolDiff.bak'WITH file=3
```

消息

已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 504 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 1 页。
RESTORE DATABASE 成功处理了 505 页, 花费 0.021 秒(187.569 MB/秒)。
已为数据库 'school', 文件 'school_dat' (位于文件 3 上)处理了 112 页。
已为数据库 'school', 文件 'school_log' (位于文件 3 上)处理了 1 页。
RESTORE DATABASE 成功处理了 113 页, 花费 0.013 秒(67.420 MB/秒)。

完成时间: 2021-04-16T16:27:10.4184353+08:00

观察 student 数据

```
Select * from student
```

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX
2	0002	李文庆	男	23	JSJ
3	7001	王海	男	23	JSJ
4	7002	赵燕	女	22	JSJ

思考: 如果仅执行下述恢复语句,能查看 student 的数据吗?

```
RESTORE DATABASE school from DISK='c:\schoolDiff.bak' WITH file=1, NORECOVERY
```

消息

已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 504 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 1 页。
RESTORE DATABASE 成功处理了 505 页, 花费 0.016 秒(246.185 MB/秒)。

完成时间: 2021-04-16T16:28:49.8857464+08:00

```
Select * from student
```

100 %

消息

消息 208, 级别 16, 状态 1, 第 1 行
对象名 'student' 无效。

完成时间: 2021-04-16T16:29:37.8350980+08:00

不能。NORECOVERY 表示还原未恢复, 创建了差异备份需要指定还原的差异备份文件。

(三) 利用日志备份

设置故障还原模型为: 完全

```
USE master;  
GO  
ALTER DATABASE school SET RECOVERY FULL;
```

建立备份

制作数据文件备份 schooldata1.bak

```
BACKUP DATABASE school TO DISK='C:\schooldata1.bak';
```

已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 504 页。
已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 1 页。
BACKUP DATABASE 成功处理了 505 页, 花费 0.021 秒(187.569 MB/秒)。

完成时间: 2021-05-03T20:06:40.8829234+08:00

把学号 7003, 姓名: 王江, 性别: 男, 年龄为 23 的学生加入 student

```
INSERT INTO Student (Sno, Sname, Ssex, Sage) VALUES ('7003', '王江', '男', 23);
```

制作日志备份存入 schoollog.bak

```
BACKUP LOG school TO DISK = 'C:\schoollog.bak';
```

已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 20 页。
BACKUP LOG 成功处理了 20 页, 花费 0.005 秒(30.664 MB/秒)。

完成时间: 2021-05-03T20:21:10.8839857+08:00

把学号 7004, 姓名: 赵兰, 性别: 女, 年龄为 22 的学生加入 student

```
INSERT INTO Student (Sno, Sname, Ssex, Sage) VALUES ('7004', '赵兰', '女', 22);
```

制作日志备份存入 schoollog.bak

```
BACKUP LOG school TO DISK = 'C:\schoollog.bak';
```

已为数据库 'school', 文件 'school_log' (位于文件 2 上)处理了 2 页。
BACKUP LOG 成功处理了 2 页, 花费 0.010 秒(1.269 MB/秒)。

完成时间: 2021-05-03T20:22:34.4988809+08:00

观察 schoollog 中的信息

```
RESTORE FILELISTONLY FROM DISK='C:\schoollog.bak'
RESTORE HEADERONLY FROM DISK='C:\schoollog.bak'
```

	LogicalName	PhysicalName	Type	FileGroupName	Size	MaxSize	FileId	CreateLSN	DropLSN	UniqueId
1	school_dat	C:\data\school1.mdf	D	PRIMARY	8388608	8388608	1	0	0	88078EE3-1D06-4E30-B3
2	school_log	C:\data\school1.ldf	L	NULL	2097152	3145728	2	0	0	F7F06D55-55FC-4207-84

	BackupName	BackupDescription	BackupType	ExpirationDate	Compressed	Position	DeviceType	UserName	ServerName
1	NULL	NULL	2	NULL	0	1	2	HONOR-LAPTOP\Holger	HONOR-LA
2	NULL	NULL	2	NULL	0	2	2	HONOR-LAPTOP\Holger	HONOR-LA

删除 school 数据库

```
USE master;
GO
DROP DATABASE school;
```

100 %

消息

命令已成功完成。

完成时间: 2021-05-03T20:34:34.3477081+08:00

利用 schooldata1.bak 及 schoollog.bak 恢复数据库 school 到最新状态

```
RESTORE DATABASE school FROM DISK='C:\schooldata1.bak' WITH NORECOVERY;
RESTORE LOG school FROM DISK='C:\schoollog.bak' WITH FILE=1, NORECOVERY;
RESTORE LOG school FROM DISK='C:\schoollog.bak' WITH FILE=2;
```

已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 504 页。
 已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 1 页。
 RESTORE DATABASE 成功处理了 505 页, 花费 0.020 秒(196.948 MB/秒)。
 已为数据库 'school', 文件 'school_dat' (位于文件 1 上)处理了 0 页。
 已为数据库 'school', 文件 'school_log' (位于文件 1 上)处理了 20 页。
 RESTORE LOG 成功处理了 20 页, 花费 0.005 秒(30.664 MB/秒)。
 已为数据库 'school', 文件 'school_dat' (位于文件 2 上)处理了 0 页。
 已为数据库 'school', 文件 'school_log' (位于文件 2 上)处理了 2 页。
 RESTORE LOG 成功处理了 2 页, 花费 0.004 秒(3.173 MB/秒)。

完成时间: 2021-05-03T21:15:33.8652718+08:00

```
SELECT * FROM Student;
```

100 %

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX
2	0002	李文庆	男	23	JSJ
3	7001	王海	男	23	JSJ
4	7002	赵燕	女	22	JSJ
5	7003	王江	男	23	JSJ
6	7004	赵兰	女	22	JSJ

(四) 使用企业管理器练习备份调度策略

首先启动 SQL Server 代理服务:

```
sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO
sp_configure 'Agent XPs', 1;
GO
RECONFIGURE
```

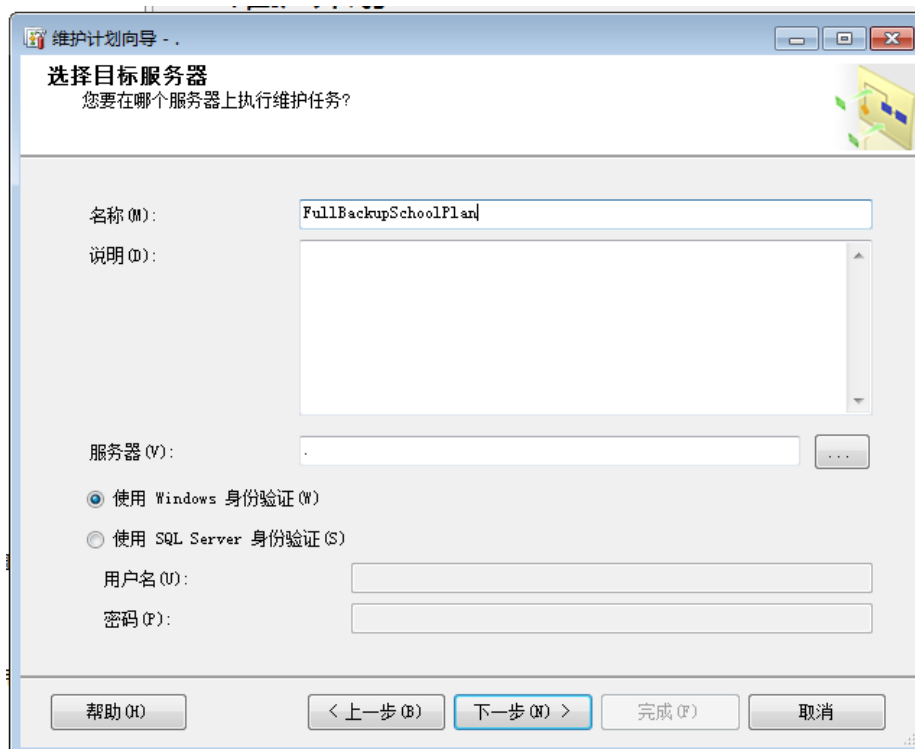



配置选项 'show advanced options' 已从 0 更改为 1。请运行 RECONFIGURE 语句进行安装。
配置选项 'Agent XPs' 已从 0 更改为 1。请运行 RECONFIGURE 语句进行安装。

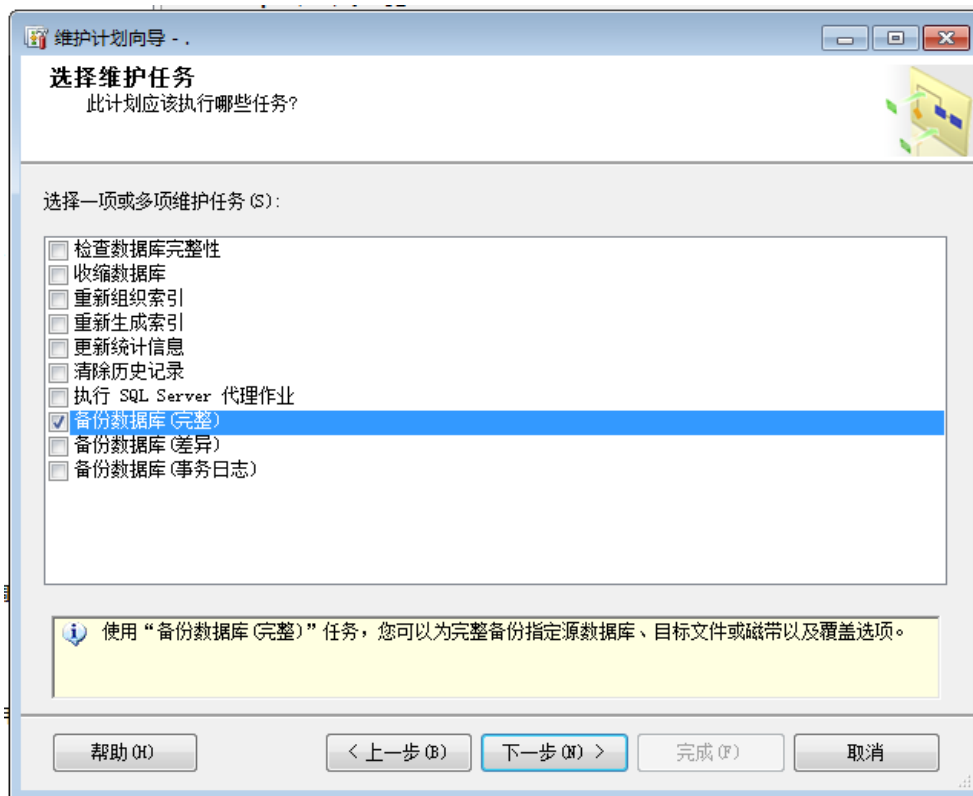
之后需要运行 **RECONFIGURE**

对数据库 school 每天上午 8 时进行一次数据库完全备份。

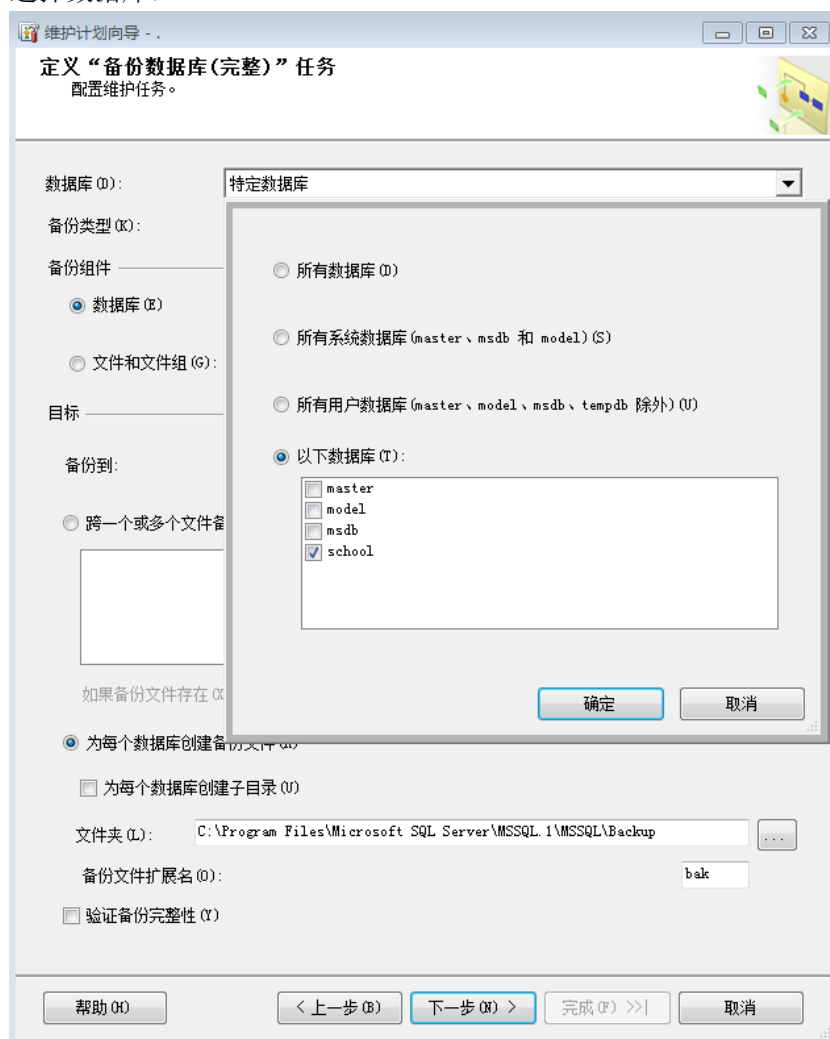
“对象资源管理器”->“管理”->“维护计划”，右击选择“维护计划向导”



选择：数据库完全备份



选择数据库:



创建计划：

维护计划向导 - .

选择计划属性
定义维护计划的计划。

计划:
未计划 (设置) 更改 (C)...

帮助 (H) < 上一步 (B) 下一步 (N) > 完成 (F) 取消

新建作业计划

名称 (N): 每天上午8时 计划中的作业 (J)

计划类型 (S): 重复执行 ☒ 已启用 (E)

执行一次

日期 (D): 2021/ 5/ 7 时间 (T): 8:58:49

频率

执行 (R): 每天

执行间隔 (C): 1 天

每天频率

☐ 执行一次, 时间为 (A): 8:00:00

☐ 执行间隔 (I): 1 小时 开始时间 (T): 0:00:00

结束时间 (G): 23:59:59

持续时间

开始日期 (D): 2021/ 5/ 7 ☐ 结束日期 (E): 2021/ 5/ 7

☒ 无结束日期 (E)

摘要

说明 (T): 在每天的 8:00:00 执行。将从 2021/5/7 开始使用计划。

确定 取消 帮助

维护计划向导 - .

完成该向导
验证在向导中选择的选项, 然后单击“完成”。

单击“完成”以执行下列操作:

- 维护计划向导
 - 在“...”上创建维护计划“FullBackupSchoolPlan”
 - 定义“备份数据库 (完整)”任务
 - 在目标服务器连接上备份数据库
 - 数据库: school
 - 类型: 完整
 - 追加现有
 - 目标: 磁盘
 - 定义维护计划属性。
 - 安排运行 SQL Server 代理作业: 在每天的 8:00:00 执行。将从 2021/5/7 开始使用计划。
 - 所选报告选项

帮助 (H) < 上一步 (B) 下一步 (N) > 完成 (F) 取消

对数据库 school 的每隔 1 分钟备份进行一次差异备份。
与上一个的区别在于作业计划不同：

新建作业计划

名称 (N): 每隔1分钟 计划中的作业 (J)

计划类型 (S): 重复执行 ☒ 已启用 (E)

执行一次

日期 (D): 2021/ 5/ 7 时间 (T): 9:03:38

频率

执行 (R): 每天

执行间隔 (C): 1 分钟

每天频率

☐ 执行一次, 时间为 (A): 0:00:00

☒ 执行间隔 (I): 1 分钟 开始时间 (T): 0:00:00

结束时间 (G): 23:59:59

持续时间

开始日期 (D): 2021/ 5/ 7 ☐ 结束日期 (E): 2021/ 5/ 7

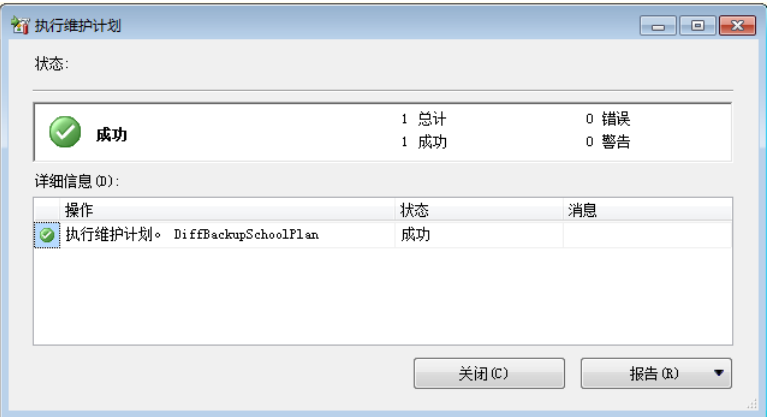
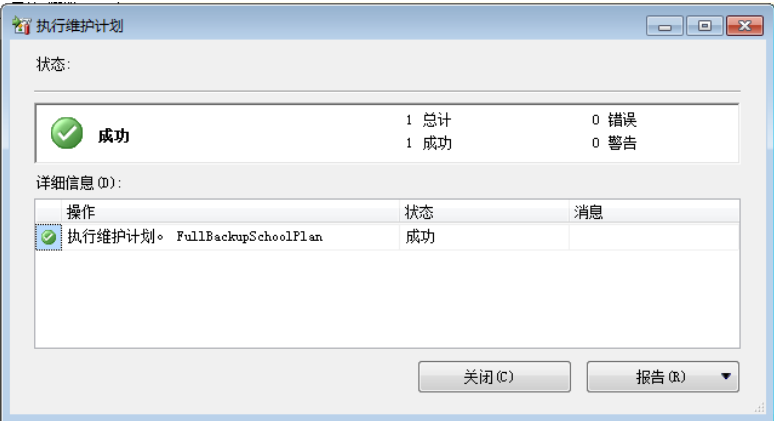
☒ 无结束日期 (E)

摘要

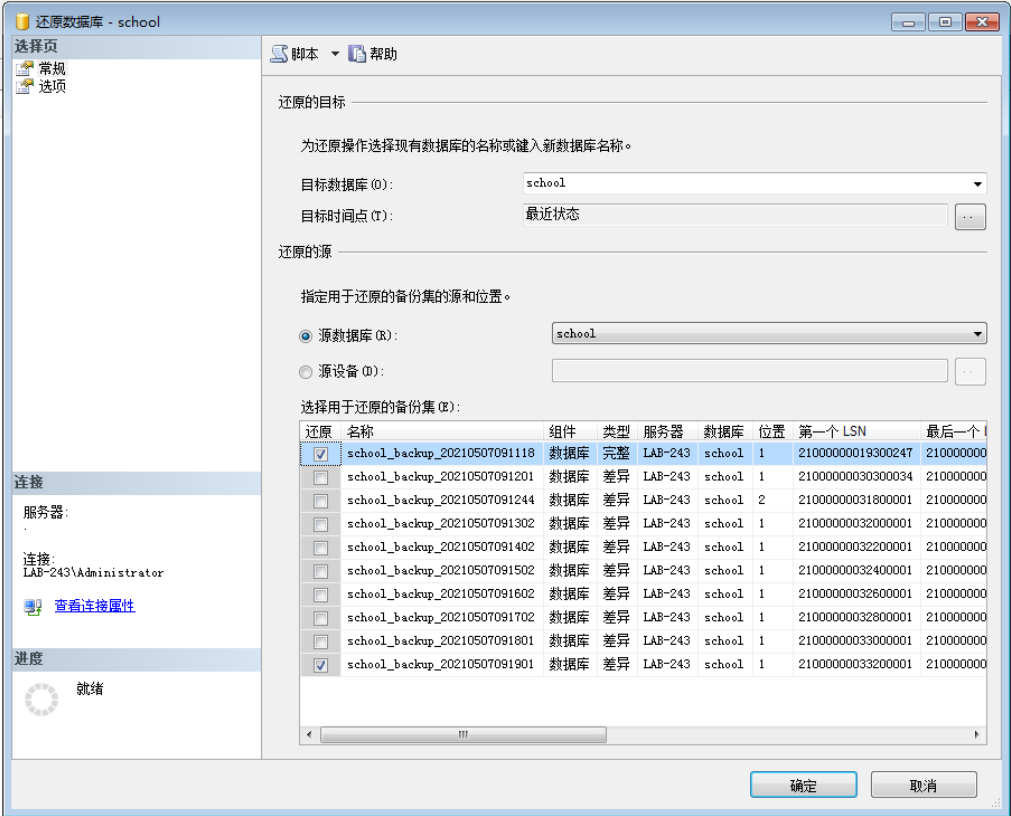
说明 (T): 每天在 0:00:00 和 23:59:59 之间、每 1 分钟 执行。将从 2021/5/7 开始使用计划。

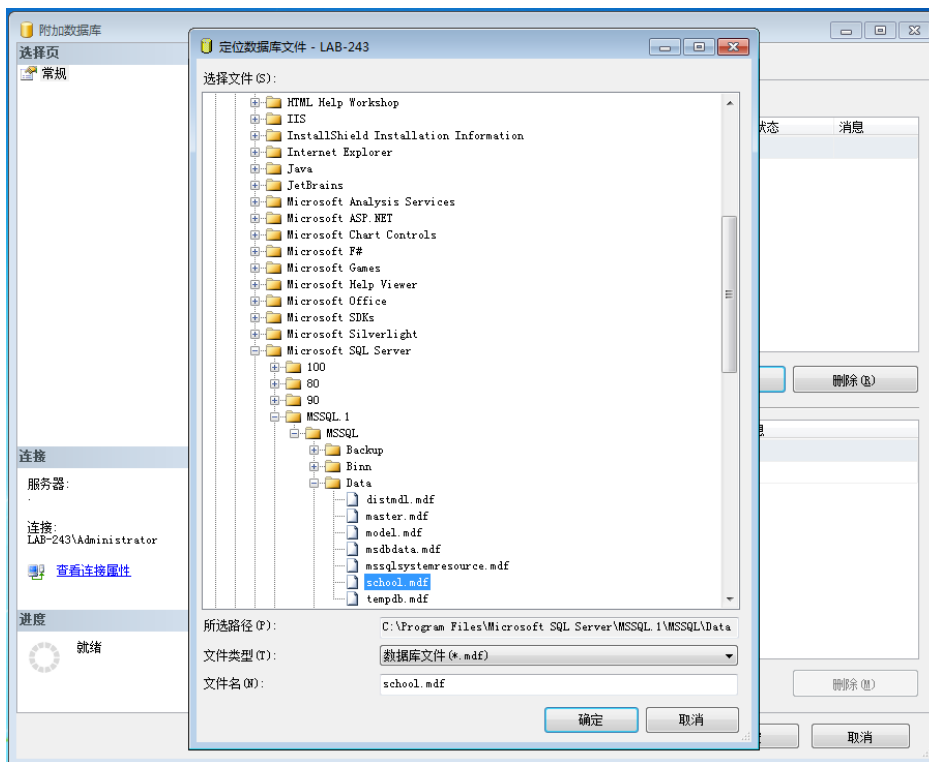
确定 取消 帮助

手工启动两个备份作业
展开“维护计划”，分别右击两个备份作业，点击执行。



删除 school 数据库
利用 1, 2 步的备份进行 school 的恢复。
右键“数据库”，点击“还原数据库”：





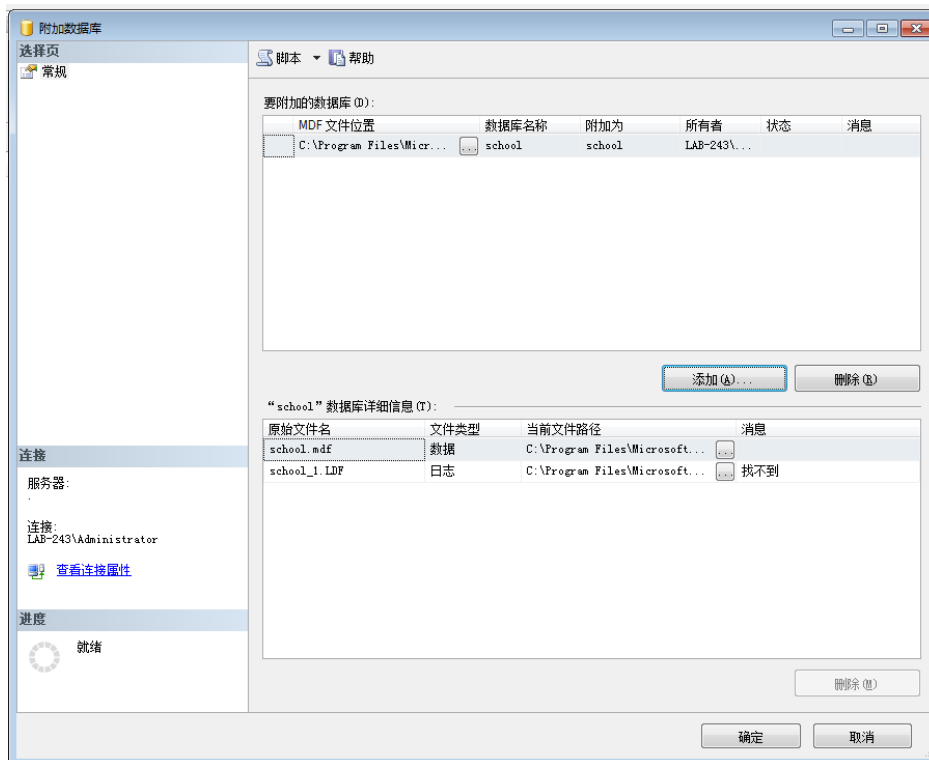
点击确定即可。

(六) 如何清除日志文件

首先使用上述方法分离数据库。

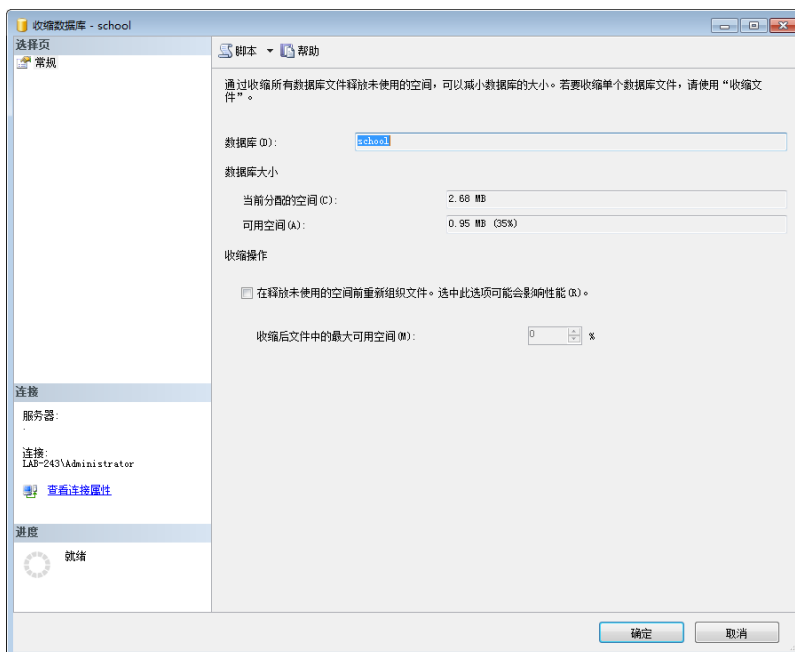
手动删除 Log 文件。

再次附加数据库，附加的时候会提醒找不到日志文件，删除掉日志条目，这样附加数据库之后将生成新的日志文件，旧的日志文件就清除了。



(七) 使用企业管理器练习数据库的压缩

右击 school 数据库->任务->收缩->数据库：

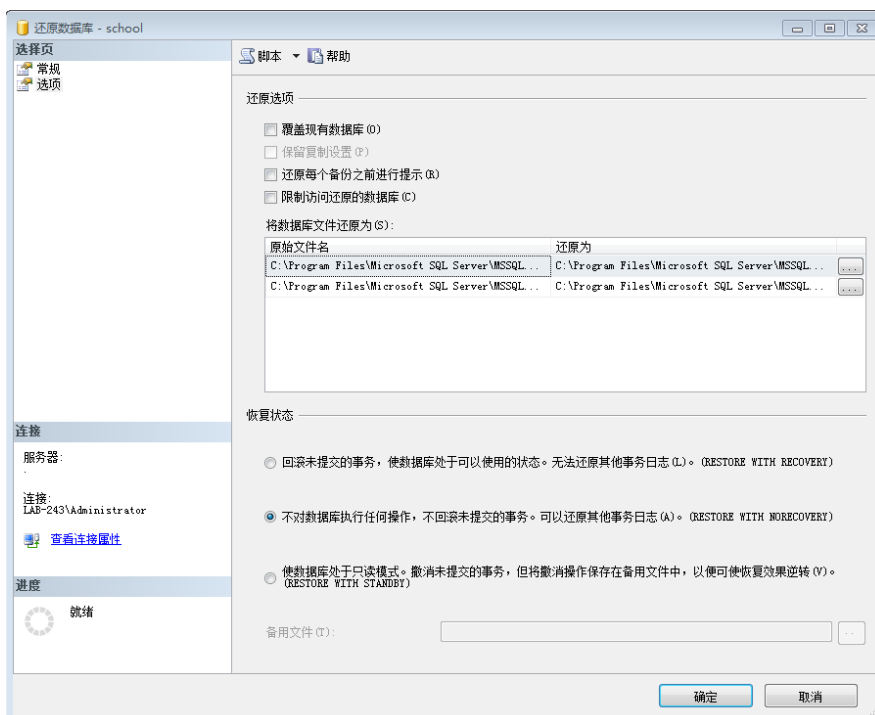


(八) 把 school 备份到其他计算机上

核心理念是，在远程电脑上建立一个共享文件夹，然后数据库备份的时候直接备份到那个共享文件夹里。

如果没有 norecovery 发生的情况：NORECOVERY 表示还原未恢复，创建了差异备份需要指定还原的差异备份文件。

企业管理器实现 NORECOVERY：“选择页”->“选项”中选择恢复状态为：不对数据库执行任何操作，不回滚未提交的事务。



四、实验总结

通过实验，我掌握了数据库的备份及恢复的方法，以及了解了备份方案的设定。

试验十三 事务

一、实验目的

掌握并理解事务

二、实验内容

(一) 理解 rollback

1 在查询分析器输入下列语句并执行 ,记录该学生的年龄。

```
Select * from student where sno='0001'
```

2 执行下列 语句序列 A:

```
BEGIN TRANsaction
    Update student set sage=sage+1 where sno='0001'
    Select * from student where sno='0002'
```

此事务结束了吗?

3 执行:

```
Select * from student where sno='0001'
```

记录该学生的年龄。

思考: student 中的 0001 的年龄确实被更改了吗? 为什么?

4 执行下列语句。

```
ROLLBACK  TRANsaction
```

然后再执行:

```
Select * from student where sno='0001'
```

观察 0001 的年龄,解释发生这种现象的原因。

(二) 理解 commit

1 在查询分析器输入下列语句并执行 ,记录该学生的年龄。

```
Select * from student where sno='0001'
```

2 执行下列 语句序列 A:

```
BEGIN TRANsaction
    Update student set sage=sage+1 where sno='0001'
    Select * from student where sno='0002'
```

3 执行:

```
commit transaction
Select * from student where sno='0001'
```

记录结果, 此时更改后的数据被永久保存了吗?

(三) 执行下列 语句序列

```
BEGIN TRANsaction
    Update student set sage=sage+1 where sno='0001'
    Update sc set grade=grade + 1 where sno='0002' and cno='1001'
Rollback
```

上述指令执行后, 数据库发生了什么变化?

三、实验结果

(一) 理解 rollback

1 在查询分析器输入下列语句并执行 ,记录该学生的年龄。

```
Select * from student where sno='0001'
```


	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX

2 执行下列 语句序列 A:

BEGIN TRANsaction

Update student set sage=sage+1 where sno='0001'

Select * from student where sno='0002'

	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ

此事务没有结束

3 执行:

Select * from student where sno='0001'

记录该学生的年龄。

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	21	SX

student 中的 0001 的年龄确实被更改了, 因为在事务中执行了 update 语句

4 执行下列语句。

ROLLBACK TRANsaction

然后再执行:

Select * from student where sno='0001'

观察 0001 的年龄, 解释发生这种现象的原因。

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX

Rollback 回滚了事务中所有已执行的语句。

(二) 理解 commit

1 在查询分析器输入下列语句并执行 ,记录该学生的年龄。

Select * from student where sno='0001'

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	20	SX

2 执行下列 语句序列 A:

BEGIN TRANsaction

Update student set sage=sage+1 where sno='0001'

Select * from student where sno='0002'

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ

3 执行:

commit transaction

Select * from student where sno='0001'

	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	21	SX

此时更改后的数据被永久保存了

(三) 执行下列 语句序列

```
BEGIN TRANsaction
```

```
Update student set sage=sage+1 where sno='0001'
```

```
Update sc set grade=grade + 1 where sno='0002' and cno='1001'
```

```
Rollback
```

(1 行受影响)

(1 行受影响)

完成时间: 2021-05-03T23:39:28.5201181+08:00

上述指令执行后，数据库没有发生变化。

四、实验总结

通过实验，我掌握并理解了数据库中事务的含义和作用。事务能够对整个操作进行控制，如果任何一个语句操作失败那么整个操作就失败，以后操作就会回滚到操作前状态，或者是上个节点。确保要么执行，要么不执行。而锁是实现事务的关键，可以保证事务的完整性和并发性。

试验十四 锁

一、实验目的

理解锁的概念及锁的作用

二、实验内容

(一) 利用帮助系统了解 Sql-server 的下列语句的含义

1 锁的隔离级别

```
SET TRANSACTION ISOLATION LEVEL Serializable
```

2 设置锁定超时时间

```
SET LOCK_TIMEOUT 5000
```

3 SP_LOCK

(二) 观察封锁

1 执行语句序列 A

```
BEGIN TRANsaction
```

```
Update student set sage=sage+1 where sno='0001'
```

```
Select * from student where sno='0002'
```

2 在查询分析器中打开第二个连接(连接 school)[文件-连接], 输入下列语句:

```
1) select * from student where sno='0002'
```

记录执行结果,说明原因。

```
2) select * from student where sno='0001'
```

记录执行结果,说明原因。(如上一没有停止, 则强行终止)

```
3) update student set sname='aaa' where sno='0002'
```

记录执行结果,说明原因。(如上一没有停止, 则强行终止)

4) 强行终止上一步的命令, 然后执行语句:

```
DBCC opentran
```

记录结果 ,思考: 如何知道此事务是那一台计算机发出的?

5) 执行:

```
select * from student where sno='0001'
```

记录执行结果,说明原因

然后回到**第一个连接**中, 执行语句:

```
commit Tran
```

观察并记录第二个连接窗口中的现象, 说明原因

(三) 了解锁的类型

1 执行下列语句

```
BEGIN TRAN
```

```
Select * from student where sno='0001'
```

```
Print 'server process ID (spid) : '
```

```
Print @@spid
```

1) 然后执行下列语句

```
exec sp_lock
```

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

2) 然后执行下列语句

```
commit tran
```

```
exec sp_lock
```

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

2 执行下列语句

```
BEGIN TRAN
```

```
Update student set sage=sage + 1 where sno='1001'
```

```
Print 'server process ID (spid) : '
```

```
Print @@spid
```

1) 然后执行下列语句

```
exec sp_lock
```

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

2) 然后执行下列语句

```
commit tran
```

```
exec sp_lock
```

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁。

使用 SET TRANSACTION ISOLATION LEVEL Serializable

然后重新执行 三和四步, 观察与原来有何不同。

3 了解表级锁 (查看帮助文件)

```
BEGIN TRAN
```

```
Select * from student (TABLOCKX) where sno='1002'
```

```
Print 'Server Process ID (spid): '
```

```
Print @@spid
```

然后执行:

```
exec sp_lock
```

注意根据事务中输出的 spid ,观察结果中相应 spid 的记录, 观察加锁类型。

4 了解锁定超时

1) 执行下列语句, 设置锁定超时为 1000 ms

```
set lock_timeout 1000
```

```
go
```

```
BEGIN TRAN
```

```
Select * from student (TABLOCKX) where sno='1002'
```

2) 打开第二个连接

执行:

```
select * from student
```

记录观察到的现象。

3) 在打开的第二个连接中

```
set lock_timeout 10000
```

```
go
```

```
select * from student
```

记录观察到的现象。

(四) 编程实例

1 编写存储过程 usp_update1, 传入参数为课程号,处理逻辑:

对传入的这门课,进行如下处理:

如某学生该门课成绩>80, 则加 2 分

如某学生该门课成绩>60, 则加 1 分

如某学生该门课成绩<=60,扣 1 分

要求: 在存储过程中, 要么全部学生的成绩被处理成功, 要么全部不处理

思考: 在调试中, 采用那些措施, 使存储过程运行时执行回滚操作 (rollback) 。

2 编写触发器，

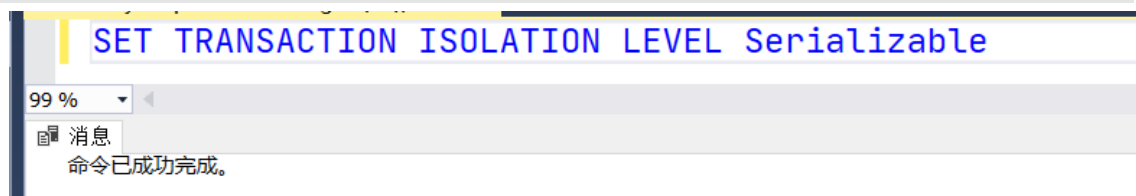
对 insert、update 语句进行监控，当学生的年龄超过 40 岁时，把该学生的系科改为 'BAK',同时删除该学生的所有选课记录。（注意，利用事务，使修改系科及删除选课记录要么全做，要么全不做）

三、实验结果

（一） 利用帮助系统了解 Sql-server 的下列语句的含义

1 锁的隔离级别

SET TRANSACTION ISOLATION LEVEL Serializable



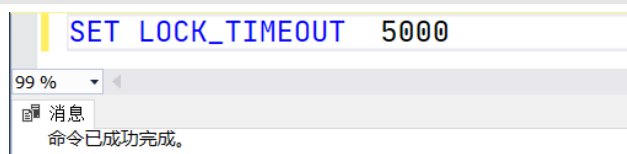
指定锁的隔离级别。事务隔离级别通过影响读操作来间接地影响写操作。

隔离级别：READ UNCOMMITTED < READ COMMITTED < REPEATABLE READ < SERIALIZABLE。隔离级别越高，读操作的请求锁定就越严格，锁的持有时间久越长；所以隔离级别越高，一致性就越高，并发性就越低，同时性能也相对影响越大。

- READ UNCOMMITTED（未提交读）：读操作不申请锁，允许读脏数据，读操作不会影响写操作请求排他锁。
- READ COMMITTED（已提交读）是 SQL SERVER 默认的隔离级别，可以避免读取未提交的数据。该隔离级别读操作之前首先申请并获得共享锁，允许其他读操作读取该锁定的数据，但是写操作必须等待锁释放，一般读操作读取完就会立刻释放共享锁。
- REPEATABLE READ（可重复读）：保证在一个事务中的两个读操作之间，其他的事务不能修改当前事务读取的数据。该级别事务获取数据前必须先获得共享锁，同时获得的共享锁不立即释放，一直保持共享锁至事务完成。
- 对于 REPEATABLE READ，能保证事务可重复读，但是事务只锁定查询第一次运行时获取的数据资源（数据行），而不能锁定查询结果之外的行，就是原本不存在于数据表中的数据，这就导致了幻读现象（在一个事务中当第一个查询和第二个查询过程之间，有其他事务执行插入操作且插入数据满足第一次查询读取过滤的条件时，那么在第二次查询的结果中就会存在这些新插入的数据，使两次查询结果不一致）。使用 SERIALIZABLE（可序列化）可以避免出现幻读。

2 设置锁定超时时间

SET LOCK_TIMEOUT 5000



指定语句等待锁释放的毫秒数。参数 timeout_period 给出了在 SQL Server 返回锁定错误前经过的毫秒数。值为 -1（默认值）时表示没有超时期限（即无限期待）。当锁等待超过超时值时，将返回错误。值为 0 时表示根本不等待，一遇到锁就返回消息。

3 SP_LOCK

SP_LOCK								
99 %								
结果 消息								
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	63	5	0	0	DB		S	GRANT
2	63	1	1787153412	0	TAB		IS	GRANT
3	63	32767	-571204656	0	TAB		Sch-S	GRANT

报告有关锁的信息。Sp_lock 结果包含对于 @spid1 和 @spid2 参数中指定的会话所持有的每个锁。如果两者都没有指定，则结果为实例中当前活动的会话的所有会话的锁。

(二) 观察封锁

1 执行语句序列 A

```
BEGIN TRANsaction
```

```
Update student set sage=sage+1 where sno='0001'
```

```
Select * from student where sno='0002'
```

结果 消息					
	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ

2 在查询分析器中打开第二个连接(连接 school)[文件-连接], 输入下列语句:

```
1) select * from student where sno='0002'
```

记录执行结果,说明原因。

结果 消息					
	Sno	Sname	Ssex	Sage	Sdept
1	0002	李文庆	男	23	JSJ

能看到结果,因为在未结束事务中对 student 的 0002 学生加了共享锁,还可以对其加共享锁。

```
2) select * from student where sno='0001'
```

记录执行结果,说明原因。(如上一步没有停止,则强行终止)

正在执行查询...

不能看到结果,处于等待状态。因为在未结束事务中对 student 的 0001 学生加了排他锁,其他事务不可以对其加任何锁。

```
3) update student set sname='aaa' where sno='0002'
```

记录执行结果,说明原因。(如上一步没有停止,则强行终止)

正在执行查询...

不能看到结果。在未结束事务中对 student 的 0002 学生加了共享锁,其它事务不可以对其添加排他锁。

4) 强行终止上一步的命令,然后执行语句:

```
DBCC opentran
```

记录结果,思考:如何知道此事务是那一台计算机发出的?

消息

数据库 'school' 的事务信息。

最早的活动事务:

SPID (服务器进程 ID): 53

UID (用户 ID): -1

名称 : user_transaction

LSN : (39:672:1)

开始时间 : 05 7 2021 10:50:48:680AM

SID : 0x010500000000000515000000749106fb19bed3c1c1a42673eb030000

DBCC 执行完毕。如果 DBCC 输出了错误信息,请与系统管理员联系。

完成时间: 2021-05-07T10:58:48.5725273+08:00

可以查看当前的 UID （用户 ID）

5) 执行:

```
select * from student where sno='0001'
```

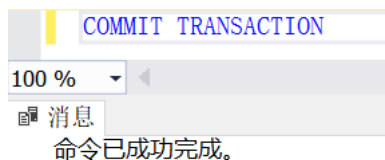
记录执行结果,说明原因

正在执行查询...

不能看到结果,处于等待状态。因为在未结束事务中对 student 的 0001 学生加了排他锁,其他事务不可以对其加任何锁。

然后回到第一个连接中,执行语句:

```
commit Tran
```



观察并记录第二个连接窗口中的现象,说明原因

结果		消息			
	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	22	SX

0001 学生的信息马上被显示出来,因为 commit 结束了事务,锁被释放,等待的操作可以加锁成功。

(三) 了解锁的类型

1 执行下列语句

```
BEGIN TRAN  
  
Select * from student where sno='0001'  
Print 'server process ID (spid) : '  
Print @@spid
```

结果		消息			
	Sno	Sname	Ssex	Sage	Sdept
1	0001	周志林	男	22	SX

(1 行受影响)
server process ID (spid) :
62
完成时间: 2021-05-07T11:23:05.0099360+08:00

1) 然后执行下列语句

```
exec sp_lock
```

注意根据事务中输出的 spid,观察结果中相应 spid 的记录,观察加锁。

结果		消息						
	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	62	5	0	0	DB		S	GRANT
2	62	1	1787153412	0	TAB		IS	GRANT
3	62	32767	-571204656	0	TAB		Sch-S	GRANT

2) 然后执行下列语句

```
commit tran
```

```
exec sp_lock
```

注意根据事务中输出的 spid,观察结果中相应 spid 的记录,观察加锁。

结果 消息

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	62	5	0	0	DB		S	GRANT
2	62	1	1787153412	0	TAB		IS	GRANT
3	62	32767	-571204656	0	TAB		Sch-S	GRANT

2 执行下列语句

BEGIN TRAN

Update student set sage=sage + 1 where sno='1001'

Print 'server process ID (spid) : '

Print @@spid

(0 行受影响)

server process ID (spid) :
62

1) 然后执行下列语句

exec sp_lock

注意根据事务中输出的 spid,观察结果中相应 spid 的记录, 观察加锁。

结果 消息

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	62	5	0	0	DB		S	GRANT
2	62	5	581577110	1	PAG	1:352	IX	GRANT
3	62	5	581577110	0	TAB		IX	GRANT
4	62	1	1787153412	0	TAB		IS	GRANT
5	62	32767	-571204656	0	TAB		Sch-S	GRANT

2) 然后执行下列语句

commit tran

exec sp_lock

注意根据事务中输出的 spid,观察结果中相应 spid 的记录, 观察加锁。

结果 消息

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	62	5	0	0	DB		S	GRANT
2	62	1	1787153412	0	TAB		IS	GRANT
3	62	32767	-571204656	0	TAB		Sch-S	GRANT

使用 SET TRANSACTION ISOLATION LEVEL Serializable

然后重新执行 三和四步, 观察与原来有何不同。

BEGIN TRAN

Update student set sage=sage + 1 where sno='1001'

Print 'server process ID (spid) : '

Print @@spid

(0 行受影响)

server process ID (spid) :
62

1) 然后执行下列语句

exec sp_lock

结果 消息

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	62	5	0	0	DB		S	GRANT
2	62	5	581577110	1	PAG	1:352	IX	GRANT
3	62	5	581577110	0	TAB		IX	GRANT
4	62	1	1787153412	0	TAB		IS	GRANT
5	62	5	581577110	1	KEY	(13ed9b5f62eb)	RangeX-X	GRANT
6	62	32767	-571204656	0	TAB		Sch-S	GRANT

2) 然后执行下列语句

```
commit tran
exec sp_lock
```

结果 消息

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	62	5	0	0	DB		S	GRANT
2	62	1	1787153412	0	TAB		IS	GRANT
3	62	32767	-571204656	0	TAB		Sch-S	GRANT

3 了解表级锁 (查看帮助文件)

BEGIN TRAN

Select * from student (TABLOCKX) where sno='1002'

Print 'Server Process ID (spid): '

Print @@spid

结果 消息

	Sno	Sname	Ssex	Sage	Sdept
1	1002	张三	男	21	JSJ

结果 消息

(1 行受影响)
Server Process ID (spid):
52

然后执行:

```
exec sp_lock
```

注意根据事务中输出的 spid,观察结果中相应 spid 的记录,观察加锁类型。

结果 消息

	spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	52	5	0	0	DB		S	GRANT
2	52	5	581577110	0	TAB		X	GRANT
3	52	1	1787153412	0	TAB		IS	GRANT
4	52	32767	-571204656	0	TAB		Sch-S	GRANT

4 了解锁定超时

1) 执行下列语句, 设置锁定超时为 1000 ms

```
set lock_timeout 1000
```

```
go
```

BEGIN TRAN

Select * from student (TABLOCKX) where sno='1002'

结果 消息


	Sno	Sname	Ssex	Sage	Sdept
1	1002	张三	男	21	JSJ

2) 打开第二个连接

执行:

```
select * from student
```

记录观察到的现象：

 正在执行查询...

不能看到结果，处于等待状态。

3) 在打开的第二个连接中

```
set lock_timeout 10000
```

```
go
```

```
select * from student
```

记录观察到的现象。

 消息

消息 1222, 级别 16, 状态 56, 第 3 行
已超过了锁请求超时段。

完成时间: 2021-05-07T11:39:59.4072166+08:00

(执行用时 10s)

(四) 编程实例

存储过程 usp_update1

```
CREATE PROCEDURE usp_update1(@cno CHAR(4)) AS
BEGIN TRANSACTION
    UPDATE SC
    SET Grade = CASE
        WHEN Grade > 80 THEN Grade + 2
        WHEN Grade > 60 AND Grade <= 80 THEN Grade + 1
        WHEN Grade <= 60 THEN Grade - 1
    END
    WHERE Cno = @cno;
    IF @@ERROR = 0
        COMMIT TRANSACTION;
    ELSE
        ROLLBACK TRANSACTION;
RETURN;
```

触发器

```
CREATE TRIGGER usp_updateStrudent ON Student FOR UPDATE AS
BEGIN
    BEGIN TRANSACTION
        UPDATE Student
        SET Sdept = 'BAK'
        WHERE Sno IN (SELECT Sno FROM inserted WHERE Sage > 40);
        DELETE FROM SC
        WHERE Sno IN (SELECT Sno FROM inserted WHERE Sage > 40);
    IF @@ERROR = 0
        COMMIT TRANSACTION;
    ELSE
        ROLLBACK TRANSACTION;
END;
```

四、实验总结

通过实验，我理解了锁的概念以及锁的作用。