12

第十二讲 数据库备份与恢复

教学内容

- 事务
  - 事务ACID属性
  - 事务类型
  - 隔离等级
- 锁
  - 锁概念
  - 并发性
- 备份与恢复
  - 备份
  - 恢复

转帐

- 中行(China_Bank)

C_Card_ID	C_Money
孙悟空	150

```
Update China_Bank
Set C_Money=C_Money-50
Where C_Card_ID='孙悟空'
```

- 交行(Traffic_Bank)

T_Card_ID	T_Money
猪八戒	200

```
Update Traffic_Bank
Set T_Money=T_Money+50
Where T_Card_ID='猪八戒'
```

创建示例表

```
Create Table China_Bank(
    C_Card_ID Varchar(10) Primary Key,
    C_Money double
);
Create Table Traffic_Bank(
    T_Card_ID Varchar(10) Primary Key,
    T_Money double
);
Insert Into China_Bank  Values('孙悟空',150);
Insert Into Traffic_Bank  Values('猪八戒',200);
```

事务概念

- 事务
  - 作为单个逻辑工作单元执行的一系列操作
  - 事务是针对数据库的一组操作，它可以由一条或多条SQL语句组成。
  - 一个逻辑工作单元必须有四个属性（ACID属性）
- ACID
  - Atomicity：原子性
  - Consistency：一致性
  - Isolation：隔离性
  - Durability：持久性

事务的属性（ACID）

- 原子性(Atomicity)
  - 事务必须是原子工作单元
  - 对于其数据修改，要么全都执行，要么全都不执行
- 一致性(Consistency)
  - 事务在完成时，必须使所有的数据都保持一致状态
- 隔离性(Isolation)
  - 由并发事务所作的修改必须与任何其它并发事务所作的修改隔离
- 持久性(Durability)
  - 事务完成之后，它对于系统的影响是永久性的

三种事务类型

- 显式事务
  - 在自动提交模式下以**start Transaction**开始一个事务，以**Commit**或**Rollback**结束一个事务
- 自动提交事务
  - 自动开始事务，自动提交事务
- 隐含事务
  - 无须描述事务的开始，只须提交或回滚每个事务
  - 如：创建或删除数据库、数据表，修改表结构等操作，隐式提交
- 存储引擎**InnoDB**支持事务，而**MyISAM**不支持事务

事务自动提交

- 查看
  - **SELECT @@autocommit;**
- 关闭自动提交
  - **SET AUTOCOMMIT = 0;**
  - 用户需要手动执行提交（**COMMIT**）操作。
  - 若直接终止**MySQL**会话，**MySQL**会自动进行回滚。

1：开启自动提交

0：关闭自动提交

显示事务示例

```
create procedure proc_trans()
BEGIN
    declare Rec_Count int default 0;
    start Transaction;
    Update China_Bank Set C_Money=C_Money-50 Where C_Card_ID='孙悟空';
   set Rec_Count=ROW_COUNT ();
    Update Traffic_Bank Set T_Money=T_Money+50 Where T_Card_ID='猪八戒';
   set Rec_Count=Rec_Count+ROW_COUNT ();
   select rec_count;
    if Rec_Count=2 then
            Commit;
    else
            Rollback;
    end IF;
END;
call proc_trans;
```

选课事务

```
Create Procedure Proc_Trans_Elect  (SNO varchar(15),TNO varchar(15),CID varchar(15))
begin
  Declare Rec_Count int default 0;
  start Transaction;
   Insert Into ElectCourseResult Values(SNO,TNO,CID);
   set Rec_Count=ROW_COUNT();
  Update TeacherCourseInfo set RemainPersonCount=RemainPersoncount+1
   where Teacherno=TNO  And CourseID=CID;
   set Rec_Count=Rec_Count+ROW_COUNT();
  if Rec_Count=2 THEN
    Commit;
  else
    Rollback;
  end if;
  end;
```

事务保存点的设置与回滚

- 语法
  - SAVEPOINT 保存点名;
- 功能
  - 在回滚事务时，若希望只撤销一部分
- 回滚
  - ROLLBACK TO SAVEPOINT 保存点名;
- 删除保存点
  - RELEASE SAVEPOINT 保存点名;

保存点示例

- select * from china_bank;
- start TRANSACTION;
- update china_bank set c_money=c_money-20 where C_Card_ID='孙悟空';
- savepoint minus20;
- update china_bank set c_money=c_money-30 where C_Card_ID='孙悟空';
- rollback to minus20;
- select * from china_bank;
- rollback;

事务隔离级别

- 数据库是一个多用户的共享资源，MySQL允许多线程并发访问
- 用户可以通过不同的线程执行不同的事务。
- 事务设置隔离级以事务之间不受影响
- 事务隔离级别
  - REPEATABLE READ：可重复读
  - READ UNCOMMITTED：读取未提交
  - READ COMMITTED：读取提交
  - SERIALIZABLE：可串行化

查看隔离级别

- 全局隔离级
  - 影响所有连接**MySQL**用户
  - **SELECT @@global.transaction_isolation;**
- 当前会话隔离级
  - 只影响当前正在登录**MySQL**服务器的用户。
  - **SELECT @@session.transaction_isolation;**
- 下一个事务的隔离级
  - 仅对当前用户的下一个事务操作有影响。
  - **SELECT @@transaction_isolation;**

设置事务隔离级别

- 语法
  - **SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL 参数值**
- 参数
  - **SESSION**：当前会话
  - **GLOBAL**：全局
  - 直接省略：下一个事务的隔离级
  - **TRANSACTION**：事务
  - **ISOLATION**：隔离
  - **LEVEL**：级别

修改隔离级别

- SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
- SELECT @@session.transaction_isolation;
- SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;

只读事务

- 事务默认模式为 **READ WRITE**（读/写模式）
- 事务可执行：读（查询）或 写（更改、插入、删除等）操作。
- 设置只读事务
  - **SET [SESSION | GLOBAL] TRANSACTION READ ONLY**
- 恢复读写事务
  - **SET [SESSION | GLOBAL] TRANSACTION READ WRITE**

4种隔离级别

- **READ UNCOMMITTED（读取未提交）**
  - 事务中最低的级别，可以读取到其他事务中未提交的数据。
  - 也称为脏读（**Dirty Read**）：一个事务读取了另外一个事务未提交的数据。

脏读示例

- insert into china_bank values('孙悟空',200), ('猪八戒',100);
- start TRANSACTION;
- update china_bank set c_money=c_money-50 where C_Card_ID='孙悟空';
- update china_bank set c_money=c_money+50 where C_Card_ID='猪八戒';
- select * from china_bank where C_Card_ID='猪八戒';
- rollback;

脏读示例

- SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
- select * from china_bank where C_Card_ID='猪八戒';
- 提高隔离级别，解决脏读问题
  - SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
  - SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;

REPEATABLE READ（可重复读）

- 解决了脏读和不可重复读的问题
- 确保了同一事务的多个实例在并发读取数据时，会看到同样的结果。
- 幻读
  - 又被称为虚读，是指在一个事务内两次查询中数据条数不一致
  - 如：其他事务做了插入记录的操作，导致记录数有所增加。

幻读示例

- SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
- START TRANSACTION;
- SELECT SUM(C_Money) from CHINA_BANK;

INSERT INTO CHINA_BANK VALUES('沙和尚',200);

- SELECT SUM(C_Money) from CHINA_BANK;
- Commit;

REPEATABLE READ避免幻读

- SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;
- START TRANSACTION;
- SELECT SUM(C_Money) from CHINA_BANK;

用户B

INSERT INTO CHINA_BANK VALUES('沙和尚',200);

用户A

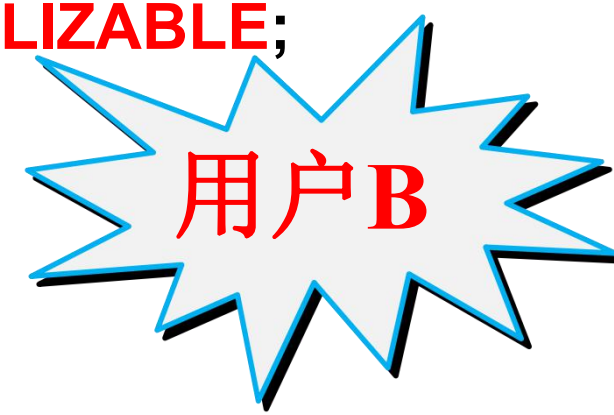
- SELECT SUM(C_Money) from CHINA_BANK;
- Commit;

SERIALIZABLE（可串行化）

- 隔离级的最高级别，它在每个读的数据行上加锁，使之不会发生冲突
- 解决了脏读、不可重复读和幻读的问题。
- 由于加锁可能导致超时（Timeout）和锁竞争（Lock Contention）现象
- 性能是4种隔离级中最低的。
- 除非为了数据的稳定性，需要强制减少并发的情况时，才会选择此种隔离级。

SERIALIZABLE示例

- SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
- START TRANSACTION;
- UPDATE China_Bank SET C_Money=550
- where c_card_id='孙悟空';



UPDATE China_Bank SET C_Money=500
where c_card_id='猪八戒';

- Commit;
- `SELECT @@innodb_lock_wait_timeout; #超时设置`

数据锁

- 锁定（Lock）
  - 将指定的数据临时锁起来使用，以防止该数据被别人修改或读取
- 并发性(Concurrency)
  - 允许多个事务同时进行数据处理的性质

死锁问题

- 当多个事务的手中都锁定了某些资源，却又在等待另外一些被彼此锁定的资源时，就会发生死锁**(Deadlock)**
- 避免死锁发生的技巧
  - 使用相同的顺序来存取数据
  - 尽量缩短事务的时间
  - 尽量使用较低的隔离等级

数据库备份概念

- 备份和恢复
  - 是维护数据库的安全性和完整性的重要组成部分
  - 备份：可防止因各种原因而造成的数据破坏和丢失
  - 恢复：指在造成数据丢失和破坏以后利用备份来恢复数据的操作
- 造成数据丢失的因素
  - 用户的错误操作和蓄意破坏
  - 病毒攻击
  - 自然界不可抗力

备份目的

- 做灾难恢复：对损坏的数据进行恢复和还原
- 需求改变：因需求改变而需要把数据还原到改变以前
- 测试：测试新功能是否可用

备份需要考虑的问题

- 可以容忍丢失多长时间的数据；
- 恢复数据要在多长时间内完；
- 恢复的时候是否需要持续提供服务；
- 恢复的对象，是整个库，多个表，还是单个库，单个表。

备份类型——数据库离线

- 冷备（cold backup）
  - 需要关mysql服务，读写请求均不允许状态下进行；
- 温备（warm backup）
  - 服务在线，但仅支持读请求，不允许写请求；
- 热备（hot backup）
  - 备份的同时，业务不受影响。

备份类型——数据范围

- 完全备份：**full backup**，备份全部字符集。
- 增量备份: **incremental backup** 上次完全备份或增量备份以来改变了的数据，不能单独使用，要借助完全备份，备份的频率取决于数据的更新频率。
- 差异备份：**differential backup** 上次完全备份以来改变了的数据。
- 建议的恢复策略：
  - 完全+增量+二进制日志
  - 完全+差异+二进制日志

语句备份

- Select * from tablename into outfile filename
- Select * from studinfo where studgender='男' into outfile 'd:/studinfo2.txt';
- delete from studinfo where studgender='男';
- load data infile 'd:/studinfo2.txt' into table studinfo;

mysqldump

- 语法
  - **mysqldump [选项] 数据库名 [表名] > 脚本名**
  - **mysqldump [选项] --数据库名 [选项 表名] > 脚本名**
  - **mysqldump [选项] --all-databases [选项] > 脚本名**
- 功能
  - 逻辑备份工具
  - 备份原理是通过协议连接到 **MySQL** 数据库，将需要备份的数据查询出来转换成对应的**insert** 语句
  - 还原数据时，执行 **insert** 语句

选项说明

参数名	缩写	含义
--host	-h	服务器IP地址
--port	-P	服务器端口号
--user	-u	MySQL 用户名
--pasword	-p	MySQL 密码
--databases		指定要备份的数据库
--all-databases		备份mysql服务器上的所有数据库
--compact		压缩模式，产生更少的输出
--comments		添加注释信息
--complete-insert		输出完成的插入语句
--lock-tables		备份前，锁定所有数据库表
--no-create-db/--no-create-info		禁止生成创建数据库语句
--force		当出现错误时仍然继续备份操作
--default-character-set		指定默认字符集
--add-locks		备份数据库表时锁定数据库表

导出数据库

- 导出所有数据库
  - **mysqldump -uroot -proot --all-databases >d:/alldb.sql**
- 导出**db1、db2**两个数据库的所有数据
  - **mysqldump -uroot -proot --databases db1 db2 >d:/dbbackup.db**

导出表

- 导出指定表
  - **mysqldump -uroot -p --databases studscore_db --tables StudInfo StudScoreInfo >d:/test.sql**
- 注：
  - 导出指定表只能针对一个数据库进行导出
  - 导出指定表的导出文本中没有创建数据库的判断语句，只有删除表-创建表-导入数据
- 排除某些表
  - **mysqldump -uroot -p --databases studscore_db --ignore-table=studscore_db.StudScoreInfo >d:/test2.sql**

条件导出

- Where子句
  - mysqldump -uroot -pgenius --databases studscore_db --tables studscoreinfo --where="studscore<60" >d:/studscore60.sql
  - Delete from studscoreinfo where studscore<60;

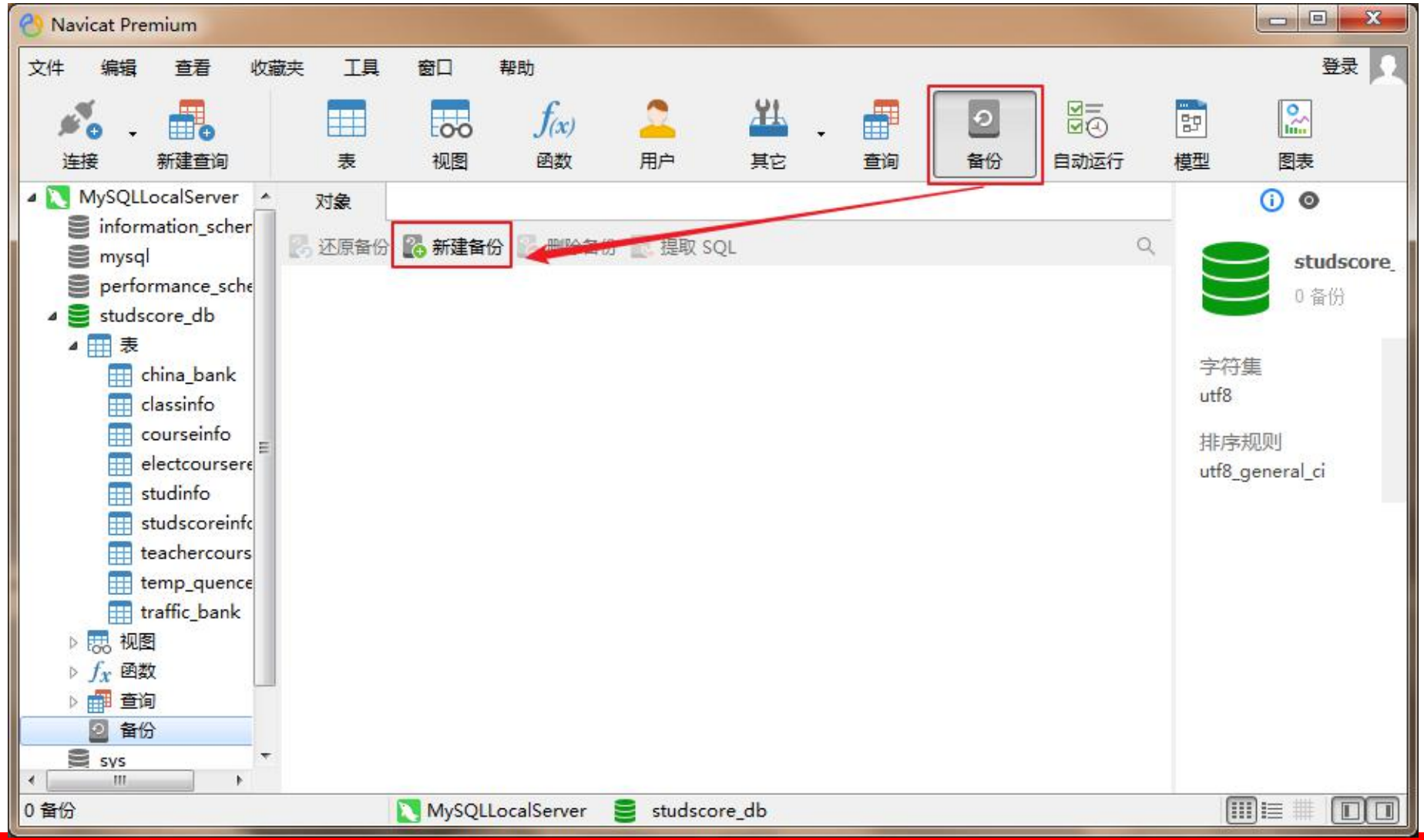
恢复数据

- 恢复数据
  - 语法：**mysql -h[ip] -P[(大写)端口] -u[用户名] -p[密码] [数据库名] < d:XX.sql(路径)**
  - 示例：**mysql -uroot -p -h127.0.0.1 -P3306 studscore_db < d:/studscore60.sql**
- **Source方法**
  - **mysql -uroot -p -h127.0.0.1**
  - **Use studscore_db**
  - **source d:/studscore60.sql**

自动备份

- **MyBackupDB.BAT**
  - **d:**
  - **cd D:/mysql-8.0.19-winx64/bin**
  - **mysqldump -uroot -pgenius studscore_db > d:/StudScore_DB_back.sql**
  - **Exit**
- 控制面板→管理工具→任务计划程序→创建基本任务…

Navicat备份



下次课内容

- 好关系模式判定标准
  - 尽可能少的数据冗余
  - 没有插入异常
  - 没有删除异常
  - 没有更新异常
- 函数依赖
- 范式理论
  - 第一范式
  - 第二范式
  - 第三范式
- 关系模式分解方法