蜗牛学院

重构IT职业教育新生态

MySQL

- 查询语句

目录

CONTENTS

蜗牛学院

本章目标

蜗牛学院

一．掌握变量定义及使用

二．掌握存储过程、函数及触发器的基本使用

三．熟悉游标的使用

四．了解事务的作用及基本用法

五．熟悉视图、索引的基本使用

什么是mysql高级特性

蜗牛学院

- mysql高级特性是mysql对标准SQL的一个扩充。
- 不止mysql有对SQL补充，比如msSQL有T-SQL，Oracle有PL-SQL,都是对标准SQL的扩充，内容都大同小异，存在一些语法差异。

变量

蜗牛学院

- 1、用户变量

以"@"开始，形式为"@变量名"

实例

SET @nametest=666

用户变量跟mysql客户端是绑定的，设置的变量，只对当前用户使用的客户端生效

- 2、系统变量

a、全局变量

在MYSQL启动的时候由服务器自动将它们初始化为默认值，这些默认值可以通过更改my.ini这个文件来更改。

通过：show global variables查看所有的全局变量

变量

蜗牛学院

- 2、系统变量

b.会话变量

会话变量在每次建立一个新的连接的时候，由MYSQL来初始化。MYSQL会将当前所有全局变量的值复制一份。来做为会话变量。（也就是说，如果在建立会话以后，没有手动更改过会话变量与全局变量的值，那所有这些变量的值都是一样的。）

全局变量与会话变量的区别就在于，对全局变量的修改会影响到整个服务器，但是对会话变量的修改，只会影响到当前的会话（也就是当前的数据库连接）。

通过：show session variables查看所有的会话变量。

会话变量有表达式 “set @变量名”进行声明和定义。变量名前面有一个@符号。

- 3、局部变量

局部变量一般用在sql语句块中,如存储过程或函数内定义。

作用范围在begin到end语句块之间，在该语句块里设置的变量。

declare语句专门用于定义局部变量，局部变量的值可以由set语句进行设置。

注意：局部变量前面不能加@符号，尽量不要与会话变量同名，否则会发生变量覆盖。

局部变量

蜗牛学院

- 定义

```
DECLARE var_name [, var_name] ... type [DEFAULT value]
```

- 实例

```
DECLARE v INT DEFAULT 1;
```

- 赋值

方式一:

```
SET var_name=value
```

方式二:

```
SELECT col_name[,...] INTO var_name[,...] table_expr [WHERE...];
```

- 展示值

```
SELECT var_name;
```

完整案例

蜗牛学院

案例

- 定义一个变量用于存用户名，设置默认值为张三，分别用两种方式给该变量赋值，并展示。

```
DROP PROCEDURE IF EXISTS `pro_vartest`;
CREATE PROCEDURE pro_vartest()
        BEGIN
          -- 定义varchar类型的vsname变量，并设置默认值为“张三”
          DECLARE vsname VARCHAR(20) DEFAULT "张三";
          SELECT vsname;
          -- 赋值一：
          SET vsname="李四";
          SELECT vsname;

          -- 赋值二：
          SELECT `name` INTO vsname FROM reader WHERE reader_id=111;
          SELECT vsname;
        END
```

流程控制

蜗牛学院

- 顺序
- 分支
  - IF
  - CASE
- 循环
  - REPEAT
  - WHILE

IF

蜗牛学院

- 语法

```
IF search_condition THEN statement_list
        [ELSEIF search_condition THEN statement_list] ...
        [ELSE statement_list]
END IF
```

- 注意：在mysql中，没有{}，只有BEGIN和END,代替{}

IF

蜗牛学院

- 实例

```
CREATE FUNCTION SimpleCompare(n INT, m INT)
        RETURNS VARCHAR(20)
        BEGIN
          DECLARE s VARCHAR(20);
          IF n > m THEN
            SET s = '>';
          ELSEIF n = m THEN
            SET s = '=';
          ELSE
            SET s = '<';
          END IF;
          SET s = CONCAT(n, ' ', s, ' ', m);
          RETURN s;
END
select SimpleCompare(4, 5);
```

完整案例

蜗牛学院

案例

- 判断一个数是正数还是负数还是零

```
CREATE PROCEDURE proc_iftest()
 BEGIN
            DECLARE num INT DEFAULT 0;
            IF num > 0 THEN
               SELECT "这是一个正数";
            ELSEIF num < 0 THEN
               SELECT "这是一个负数";
            ELSE
               SELECT "这是零";
            END IF;
END
```

- 练习
  - 判断张三是儿童(0~12)少年(12~18)还是青年(18~25)还是壮年(25~60)还是老年人(60<)

CASE

蜗牛学院

- 语法

```
CASE case_value
        WHEN when_value THEN statement_list
        [WHEN when_value THEN statement_list] ...
[ELSE statement_list]
        ...
END CASE
```

- 注意：不能用于判断null是否相等，因为NULL=NUll is false

CASE

蜗牛学院

- 实例

```
CREATE PROCEDURE p()
        BEGIN
          DECLARE v INT DEFAULT 1;
          CASE v
            WHEN 2 THEN SELECT v;
            WHEN 3 THEN SELECT 0;
          ELSE
            BEGIN
              SELECT 'hello';
            END;
          END CASE;
END
```

完整案例

蜗牛学院

案例 • 判断一个数的值

```
CREATE PROCEDURE proc_casetest()
        BEGIN
          DECLARE choice INT DEFAULT 3;
          CASE choice
            WHEN 0 THEN SELECT "0";
            WHEN 1 THEN SELECT "1";
          ELSE
            SELECT "others";
          END CASE;
END
```

REPEAT

蜗牛学院

- 语法

```
[begin_label:] REPEAT
        statement_list
UNTIL search_condition
END REPEAT [end_label] ;
```

REPEAT

蜗牛学院

- 实例

```
CREATE PROCEDURE dorepeat(p1 INT)
        BEGIN
          SET @x = 0;
          REPEAT
            SET @x = @x + 1;
            SELECT @x;
          UNTIL @x > p1 END REPEAT;
END
```

完整案例

蜗牛学院

案例

- 打印5，4，3，2，1

```
CREATE PROCEDURE proc_repeattest()
BEGIN
        DECLARE num INT DEFAULT 6;
        REPEAT
           SET num = num - 1;
           SELECT num;
        UNTIL num < 1
        END REPEAT outtest;
END
```

- 练习：计算0~10的和

WHILE

蜗牛学院

- 语法

```
[begin_label:] WHILE search_condition DO
        statement_list
 END WHILE [end_label]
```

WHILE

蜗牛学院

- 实例

```
CREATE PROCEDURE dowhile()
BEGIN
        DECLARE v1 INT DEFAULT 5;
        WHILE v1 > 0 DO
            SELECT 'hello';
            SET v1 = v1 - 1;
        END WHILE
END
```

完整案例

蜗牛学院

案例

- 输出5，4，3，2，1，0

```
CREATE PROCEDURE proc_whiletest()
        BEGIN
          DECLARE num INT DEFAULT 5;
          WHILE num > 0 DO
            SELECT num;
            SET num = num - 1;
          END WHILE;
END
```

循环跳出

蜗牛学院

- ITERATE：只能出现在Loop、repeat、while语句中，用于开始下次循环，类似continue
  - 语法
    - ITERATE label

- LEAVE：通过指定的label来退出流程控制块，如果label是在是最外面的程序块则退出该程序。可以在begin...end和循环结构中使用。相当于break
  - 语法
    - LEAVE label

循环跳出

蜗牛学院

- 实例

```
create PROCEDURE testITERATE(p int)
BEGIN
        outw:while(true) DO
                set p=p+1;
                if(p=5) THEN
                   ITERATE outw;
                ELSEIF (p=10) THEN
                   LEAVE outw;
                END IF;
                SELECT p;
         end WHILE outw;
END
```

```
调用testITERATE
SET @a=2;
CALL testITERATE(@a);
```

完整案例

蜗牛学院

案例 • 打印5，3

```
CREATE PROCEDURE proc_repeattest()
BEGIN
        DECLARE num INT DEFAULT 6;
        outtest:REPEAT
                SET num = num - 1;
                IF num = 4 THEN
                  ITERATE outtest;
                ELSEIF num = 2 THEN
                  LEAVE outtest;
                END IF;
                SELECT num;
        UNTIL num < 1
        END REPEAT outtest;
END
```

存储过程

蜗牛学院

- **概述**
  - 存储过程是一种数据库对象，是为了实现某个特定任务，将一组预编译的SQL语句以一个存储单元的形式存储在服务器上，供用户调用。
  - 存储过程在第一次执行时进行编译，然后将编译好的代码保存在高速缓存中以便以后调用，这样可以提高代码的执行效率。
- **特点**
  - 接收输入参数并以输出参数的形式将多个值返回至调用过程或批处理
  - 包含执行数据库操作（包括调用其它过程）的编程语句
  - 向调用过程返回状态值，以表明成功或失败以及失败原因
- **优点**
  - 安全机制：只给用户访问存储过程的权限，而不授予用户访问表和视图的权限
  - 改良了执行性能：只在第一次进行编译，以后执行无需重新编译，而一般SQL语句每执行一次就编译一次
  - 减少网络流量：存储过程存在于服务器上，调用时只需传递执行存储过程的执行命令和返回结果
  - 模块化的程序设计：增强了代码的可重用性，提高了开发效率

存储过程

蜗牛学院

- 语法
  - 创建

```
CREATE PROCEDURE sp_name([proc_parameter]) [characteristics]
BEGIN
        routine_body
END
proc_parameter:
   [ IN | OUT | INOUT ] param_name type
```

  - 删除

```
DROP PROCEDURE [IF EXISTS] sp_name;
```

  - 查看

```
SHOW CREATE PROCEDURE sp_name;
```

  - 使用

```
CALL db_name.sp_name;
```

存储过程

蜗牛学院

- 实例

```
# 创建
CREATE PROCEDURE simpleproc (OUT param1 INT)
BEGIN
   SELECT COUNT(*) INTO param1 FROM t;
END

# 使用
CALL simpleproc(@a);
SELECT @a;
```

无输入参数和无输出参数的存储过程

蜗牛学院

案例

- 查询学生信息，并创建存储过程

```
CREATE PROCEDURE proc_search()
BEGIN
    SELECT * FROM student;
END
```

- 练习：查询学生的成绩信息，包括姓名、科目名、成绩，创建存储过程

有输入参数的存储过程

蜗牛学院

案例

- 查询某人的成绩信息

```
CREATE PROCEDURE proc_scbyname(IN psname VARCHAR(20))
BEGIN
    SELECT * FROM sc INNER JOIN student ON sc.sno = student.sno
    WHERE sname=psname;
END
```

- 练习：
  - 1、根据性别查询学生名单
  - 2、根据年龄段查询学生名单

有输入参数有输出参数

蜗牛学院

案例

- 根据学生名字查询该学生的平均分，总分

```
CREATE PROCEDURE proc_avgsumbyname(IN psname VARCHAR(20),OUT avgscore
FLOAT,OUT sumscore FLOAT)
 BEGIN
      SELECT AVG(score),SUM(score) INTO avgscore,sumscore FROM sc
      INNER JOIN student ON sc.sno=student.sno WHERE sname=psname;
 END

CALL pro_avgsumbyname("李四",@uavg,@usum);
SELECT @uavg,@usum;
```

- 练习：根据课程名查询该课程最高分的学员姓名，返回并输出显示

函数

蜗牛学院

- 概述
  - 函数功能类似于存储过程，只是存储过程没有返回值，函数有返回值，存储过程的参数类型比函数多。
  - 函数与存储过程的调用方式不同。
- 分类
  - 系统函数
    - 流程控制函数
    - 字符串函数
    - 时间函数等
  - 自定义函数

流程控制函数

蜗牛学院

- CASE
  - 语法

```
CASE exper WHEN [condition] THEN result [WHEN [condition] THEN result ...] [ELSE result] END
```

  - 实例

```
SELECT CASE 1 WHEN 1 THEN 'one' WHEN 2 THEN 'two' ELSE 'more' END;
```

流程控制函数

蜗牛学院

- IF
  - 语法

```
# 如果expr1(条件表达式)返回结果为true，则返回expr2的值，否则返回expr3的值
IF(expr1,expr2,expr3);
```

  - 实例

```
SELECT IF(1>2,2,3);
```

流程控制函数

蜗牛学院

- IFNULL
  - 语法

```
# 如果expr1不为null，则返回expr1否则返回expr2
IFNULL(expr1,expr2);
```

  - 实例

```
SELECT IFNULL(1,0);
SELECT IFNULL(NULL,10);
```

流程控制函数

蜗牛学院

- NULLIF
  - 语法

```
# 若expr1等于expr2，则返回null，否则返回第一个参数

NULLIF(expr1,expr2);
```

  - 实例

```
SELECT NULLIF(1,1);
```

字符串函数

蜗牛学院

- 语法

CONCAT(s1,s2...,sn)：将s1,s2...,sn连接成字符串
CONCAT_WS(sep,s1,s2...,sn)：将s1,s2...,sn连接成字符串，并用sep字符间隔
substring（被截取字段，从第几位开始截取，截取长度）
TRIM(str)：去除字符串首部和尾部的所有空格
UUID()：生成具有唯一性的字符串
LAST_INSERT_ID()：返回最后插入的id值

时间函数

蜗牛学院

- 语法

CURDATE()或CURRENT_DATE() 返回当前的日期

CURTIME()或CURRENT_TIME() 返回当前的时间

DATE_ADD(date,INTERVAL int unit) 返回日期date加上间隔时间int的结果(int必须按照关键字进行格式化),如：SELECT DATE_ADD(CURRENT_DATE,INTERVAL 6 DAY);

DATE_FORMAT(date,fmt) 依照指定的fmt格式格式化日期date值，如：select DATE_FORMAT(CURDATE(),'%Y-%m-%d')

DATE_SUB(date,INTERVAL int unit) 返回日期date减去间隔时间int的结果(int必须按照关键字进行格式化),如：SELECT DATE_SUB(CURRENT_DATE,INTERVAL 6 MONTH);

NOW() 返回当前的日期和时间

自定义函数

蜗牛学院

- 语法
  - 创建

```
CREATE FUNCTION sp_name ([param_name type[,...]])
RETURNS type    -- 定义返回值类型
BEGIN
                routine_body
END
```

  - 删除

```
DROP function [IF EXISTS] sp_name;
```

自定义函数

蜗牛学院

- 语法
  - 查看

```
SHOW CREATE FUNCTION sp_name;
```

  - 使用

```
SELECT db_name.sp_name;
```

自定义函数

蜗牛学院

- 实例

```
# 创建
CREATE FUNCTION hello (s CHAR(20))
 RETURNS CHAR(50) DETERMINISTIC
            RETURN CONCAT('Hello', 's','!');
# 使用
SELECT hello('world');
```

- 说明
  - DETERMINISTIC：如果对于相同的输入参数返回相同的结果时设置为该参数
  - DETERMINISTIC(确定性的)，否则为NOT DETERMINISTIC(不确定的：默认)
  - 至少有一个return语句

自定义函数

蜗牛学院

案例

- 定义函数，将参数返回

```
CREATE FUNCTION fun_test(num INT)
RETURNS VARCHAR(20)   -- 注意这里是RETURNS
        RETURN CONCAT("这个数是:",num);   -- 方法体中返回值是用RETURN
```

- 练习：计算参数1到参数2之和

触发器

蜗牛学院

- 概述
  - 触发器是一种特殊类型的存储过程，不由用户直接调用，而且可以包含复杂的SQL语句。它们主要用于强制复杂的业务规则或要求。
  - 触发器还有助于强制引用完整性，以便在添加、更新或删除表中的行时保留表之间已定义的关系
- 特点
  - 它与表紧密相连，可以看作表定义的一部分；
  - 它不能通过名称被直接调用，更不允许带参数，而是当用户对表中的数据进行修改时，自动执行；
  - 它可以用于MySQL约束、默认值和规则的完整性检查，实施更为复杂的数据完整性约束。

触发器

蜗牛学院

- 常用操作
  - 自动生成自增字段
  - 执行更复杂的业务逻辑
  - 防止无意义的数据操作
  - 提供审计
  - 允许或限制修改某些表
  - 实现完整性规则
  - 保证数据的同步复制

触发器

蜗牛学院

- 语法
  - 创建

```
CREATE  [DEFINER = { user | CURRENT_USER }]  TRIGGER trigger_name
trigger_time trigger_event ON tbl_name FOR EACH ROW
BEGIN
        routine_body
END
#  trigger_time: { BEFORE | AFTER }
#  trigger_event: { INSERT | UPDATE | DELETE }
```

  - 注意事项
    - 不能有返回值或返回结果集
    - MYSQL中触发器中不能对本表进行 insert ,update ,delete 操作，以免递归循环触发

触发器

蜗牛学院

- old和new的使用
  - 当使用insert语句的时候，如果原表中没有数据的话，那么对于插入数据后表来说新插入的那条数据就是new。
  - 当使用update语句的时候，当修改原表数据的时候相对于修改数据后表的数据来说原表中修改的那条数据就是old，而修改数据后表被修改的那条数据就是new。
  - 当使用delete语句的时候，删除的那一条数据相对于删除数据后表的数据来说就是old。

触发器

蜗牛学院

- 语法
  - 删除

```
DROP TRIGGER [IF EXISTS] [schema_name.]trigger_name;
```

  - 查看

```
SHOW CREATE TRIGGER sp_name;
```

  - 使用
    - 执行触发器监听操作，不需要显示调用

案例

- 插入数据时检查性别的有效，无效数据将性别设置为女

```
CREATE TRIGGER tri_checksex BEFORE INSERT ON student FOR EACH ROW
BEGIN
        IF new.ssex != "男" AND new.ssex != "女" THEN
          SET new.ssex="女";
        END IF;
 END
```

条件处理器(异常处理)

蜗牛学院

- 条件处理器相当于对异常的处理，当出现某种错误时将交由相应的处理器进行处理；
- 语法
  - 创建

```
DECLARE handler_action HANDLER
        FOR condition_value [, condition_value] ...
        statement
```

  - handler_action:
    - CONTINUE: 继续执行;
    - EXIT:跳出begin end语句块;
    - UNDO:不支持。
  - condition_value:
    - mysql_error_code | SQLSTATE [VALUE] sqlstate_value
    - | condition_name | SQLWARNING | NOT FOUND | SQLEXCEPTION

条件处理器(异常处理)

蜗牛学院

- 异常处理例子

```
DROP PROCEDURE IF EXISTS exp_pro_demo;
CREATE PROCEDURE exp_pro_demo(IN p_name VARCHAR(10))
BEGIN
        -- 所有局部变量声明必须放在异常声明前面，会话变量无此限制
        DECLARE vname VARCHAR(10) DEFAULT "hello";
        DECLARE EXIT HANDLER FOR NOT FOUND SET @info='data not found';
        SELECT `sname` INTO vname from stu where sname = p_name;
        SELECT vname;
END

-- 调用存储过程
CALL exp_pro_demo('jame');
SELECT @info;
```

游标

蜗牛学院

- 概述
  - 游标（ cursor ）是一个存储在MySQL服务器上的数据库查询，它不是一条 SELECT语句，而是被该语句检索出来的结果集。
  - 在存储了游标之后，应用程序可以根据需要滚动或浏览其中的数据。
  - 游标主要用于交互式应用，其中用户需要滚动屏幕上的数据，并对数据进行浏览或做出更改。

- 注意
  - MySQL 游标只能用于存储过程（和函数）。

游标

蜗牛学院

- 语法
  - 定义
    - 定义要使用的 SELECT语句

```
DECLARE cursor_name CURSOR FOR select_statement;
```

  - 打开游标
    - 用前面定义的SELECT语句把数据实际检索出来

```
open cursor_name;
```

  - 取值
    - 对于填有数据的游标，根据需要取出（检索）各行

```
FETCH cursor_name  INTO var1,var2[,...];
```

  - 关闭游标
    - 在结束游标使用时，必须关闭游标。

```
close cursor_name;
```

游标

蜗牛学院

案例

- 创建游标存储所有女生的姓名

```
CREATE PROCEDURE proc_cursortest()
 BEGIN
        DECLARE vsname VARCHAR(20);
        DECLARE break INT DEFAULT FALSE;
      --  定义游标
        DECLARE girl_cur CURSOR FOR SELECT sname FROM student WHERE ssex = "女";
        DECLARE CONTINUE HANDLER FOR NOT Found set break=true;
      --  开启游标
        OPEN girl_cur;
      --  取值
          outt:WHILE TRUE DO
            FETCH girl_cur INTO vsname;
            IF break THEN
              LEAVE outt;
            ELSE
              SELECT vsname;
            END IF;
          END WHILE outt;
      -- 关闭游标
        CLOSE girl_cur;
 END
```

游标

蜗牛学院

练习

- 先创建两张表reader_male 和 reader_female, 表结构必须与reader表完全一致。
- 创建一个游标，对reader表的数据进行判断，如果性别为女，则将该记录插入到reader_female表中，如果性别为男，则将该记录插入到reader_male表中。

事务

蜗牛学院

- 概述
  - 事务是作为单个逻辑单元执行的一系列操作。
  - 多个操作作为一个整体向系统提交，要么执行、要么都不执行，事务是一个不可分割的工作逻辑单元。
  - 这特别适用于多用户同时操作的数据通信系统。例如：订票、银行、保险公司以及证券交易系统等。
- 注意
  - MyISAM：不支持事务，用于只读程序提高性能
  - Innodb：引擎 不能结构化编程，只能通过标记为开启、提交或回滚事务

事务

蜗牛学院

- 特性(Atom, Constant, Isolation, Duration)
  - **原子性**：组成事务处理的语句形成了一个逻辑单元，不能只执行其中的一部分。换句话说，事务是不可分割的最小单元。比如：银行转帐过程中，必须同时从一个帐户减去转帐金额，并加到另一个帐户中，只改变一个帐户是不合理的。
  - **一致性**：在事务处理执行前后，数据库是一致的。也就是说，事务应该正确的转换系统状态。比如：银行转帐过程中，要么转帐金额从一个帐户转入另一个帐户，要么两个帐户都不变，没有其他的情况。
  - **隔离性**：一个事务处理对另一个事务处理没有影响。就是说任何事务都不可能看到一个处在不完整状态下的事务。比如说，银行转帐过程中，在转帐事务没有提交之前，另一个转帐事务只能处于等待状态。
  - **持久性**：事务处理的效果能够被永久保存下来。反过来说，事务应当能够承受所有的失败，包括服务器、进程、通信以及媒体失败等等。比如：银行转帐过程中，转帐后帐户的状态要能被保存下来。

事务

蜗牛学院

- 事务控制语句
  - BEGIN或START TRANSACTION；显式地开启一个事务；
  - COMMIT；也可以使用COMMIT WORK，不过二者是等价的。COMMIT会提交事务，并使已对数据库进行的所有修改称为永久性的；
  - ROLLBACK；有可以使用ROLLBACK WORK，不过二者是等价的。回滚会结束用户的事务，并撤销正在进行的所有未提交的修改；
  - SAVEPOINT identifier；SAVEPOINT允许在事务中创建一个保存点，一个事务中可以有多个SAVEPOINT；
  - RELEASE SAVEPOINT identifier；删除一个事务的保存点，当没有指定的保存点时，执行该语句会抛出一个异常；
  - ROLLBACK TO identifier；把事务回滚到标记点；
  - SET TRANSACTION；用来设置事务的隔离级别。InnoDB存储引擎提供事务的隔离级别有READ UNCOMMITTED、READ COMMITTED、REPEATABLE READ和SERIALIZABLE。

事务的处理方式

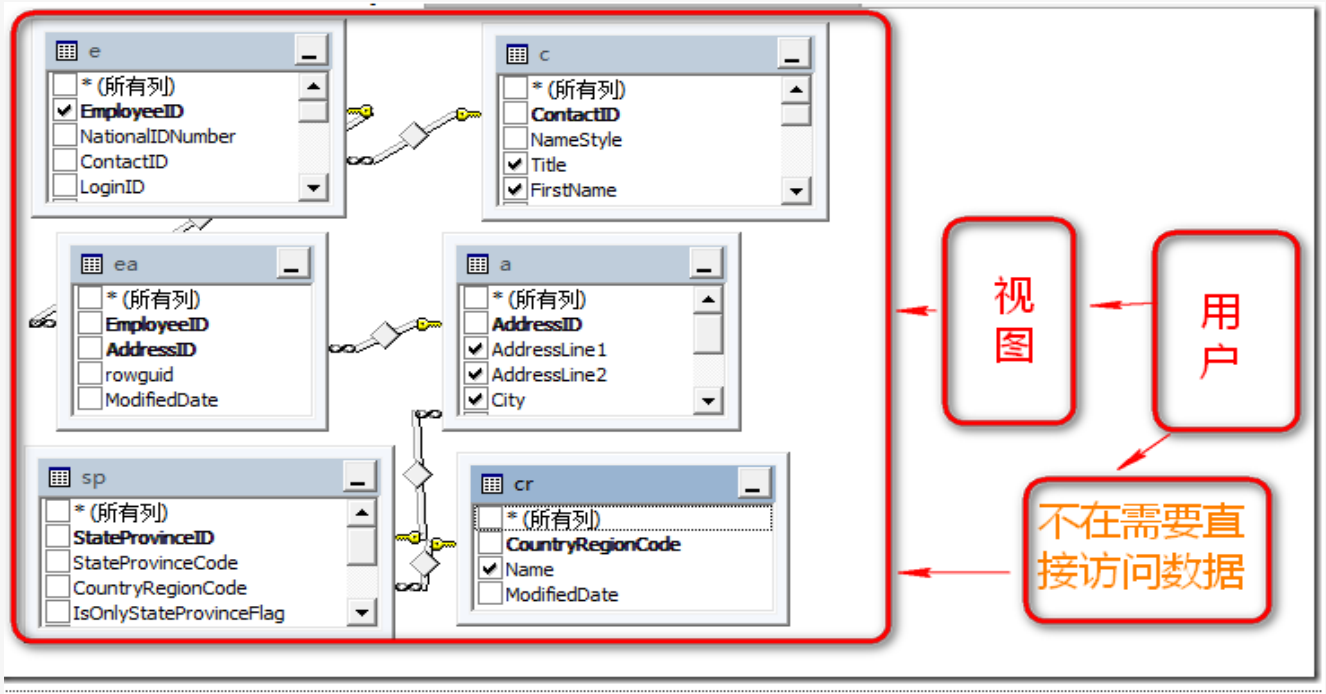
蜗牛学院

- 方式一：（用 BEGIN, ROLLBACK, COMMIT来实现）
  - BEGIN 开始一个事务
  - ROLLBACK 事务回滚
  - COMMIT 事务确认
- 方式二：（直接用 SET 来改变 MySQL 的自动提交模式）
  - SET @@AUTOCOMMIT=OFF 禁止自动提交
  - SET @@AUTOCOMMIT=ON 开启自动提交

视图

蜗牛学院

- 视图可以看作定义在Mysql上的虚拟表.视图正如其名字的含义一样，是另一种查看数据的入口.常规视图本身并不存储实际的数据，而仅仅存储一个Select语句和所涉及表的metadata.



视图相关知识

蜗牛学院

操作指令	代码
创建视图	CREATE VIEW 视图名(列1，列2...) AS SELECT (列1，列2...) FROM ...;
使用视图	当成普通的表使用
修改视图	CREATE OR REPLACE VIEW 视图名 AS SELECT [...] FROM [...];
查看数据库已有视图	SHOW TABLES [like...]（可以使用模糊查找）
查看视图详情	DESC 视图名或者SHOW FIELDS FROM 视图名

视图示例：

Create view vw1 as

Select st.学号,st.姓名,st.所属院系 from student as st,course as co,score as sc

Where co.课名='心理学' and sc.考试成绩>80 and st.学号=sc.学号 and co.课号=sc.课号

视图

蜗牛学院

- 视图中的数据并不属于视图本身，而是属于基本的表，对视图在一定条件下可以像表一样进行insert,update,delete操作。
- 视图的结构不能被修改，表修改或者删除后应该删除视图再重建或者刷新视图（sp_refreshview）。
- 视图的数量没有限制，但是命名不能和视图以及表重复，具有唯一性。
- 视图可以被嵌套，一个视图中可以嵌套另一个视图。
- 视图不能索引，不能有相关联的触发器和默认值，SQL 不能在视图后使用order by排序

索引

蜗牛学院

- 什么是索引？为什么要建立索引？

索引用于快速找出在某个列中有一特定值的行；不使用索引，MySQL必须从第一条记录开始读完整个表，直到找出相关的行，表越大，查询数据所花费的时间就越多，如果表中查询的列有一个索引，MySQL能够快速到达一个位置去搜索数据文件，而不必查看所有数据，那么将会节省很大一部分时间。

- MySsql中索引的优缺点：

优点：

1.所有的MySql列类型(字段类型)都可以被索引，也就是可以给任意字段设置索引；
2.大大加快数据的查询速度。

缺点：

1.创建索引和维护索引要耗费时间，并且随着数据量的增加所耗费的时间也会增加；
2.索引也需要占空间，如果有大量的索引，索引文件可能会比数据文件更快达到上限值；
3.当对表中的数据进行增、删、改时，索引也需要动态的维护，降低了数据的维护速度。

索引

蜗牛学院

- MySql索引分类

  单列索引(普通索引，唯一索引，主键索引)、组合索引、全文索引、空间索引

- 创建普通索引： MySQL中基本索引类型，没有什么限制，允许在定义索引的列中插入重复值和空值，纯粹为了查询数据更快一点。

```
CREATE INDEX indexName ON mytable(username(length)); -- 建立索引
或者：
ALTER table tableName ADD INDEX indexName(columnName)；-- 添加索引
或者：
CREATE TABLE mytable(
ID INT NOT NULL,
username VARCHAR(16) NOT NULL,
INDEX [indexName] (username(length)) );  -- 建表时创建索引
```

- 删除索引：

```
DROP INDEX [indexName] ON mytable;
```

- 唯一索引列中的值必须是唯一的，但是允许为空值，创建方式与普通索引类似，增加UNIQUE关键字。
- 主键索引不允许为空值；主键约束就是主键索引。

谢 谢 聆 听

蜗牛学院，只为成就更好的你！