# 1. 课程回顾

mysql基础

1）mysql存储结构： 数据库 -> 表 -> 数据 sql语句

2）管理数据库：

增加： create database 数据库 default character utf8;

删除： drop database 数据库;

修改： alter database 数据库 default character gbk;

查询： show databases / show create database 数据库;

3) 管理表：

选择数据库：use 数据库;

增加： create table 表(字段名1 字段类型,字段名2 字段类型......);

删除： drop table 表;

修改：

添加字段： alter table 表 add [column] 字段名 字段类型;

删除字段: alter table 表 drop [column] 字段名;

修改字段类型： alter table 表 modify 字段名 新的字段类型;

修改字段名称 ： alter table 表 change 旧字段名 新字段名 字段类型;

修改表名称： alter table 表 rename [to] 新表名;

查询：

show tables / desc student;

4) 管理数据:

增加： insert into 表(字段1，字段2,。。。) values(值1，值2.。。。。);

删除： delete from 表 where 条件;

修改： update 表 set 字段1=值1，字段2=值2...... where 条件;

查询：

4.1）所有字段： select \* from 表;

4.2）指定字段： select 字段1,字段2.... from 表;

4.3)指定别名： select 字段1 as 别名 from 表;

4.4 )合并列： select (字段1+字段2) from 表;

4.5)去重： select distinct 字段 from 表;

4.6)条件查询：

a)逻辑条件 ：and(与) or（或）

select \* from 表 where 条件1 and/or 条件2

b）比较条件： > < >= <= = <> between and(在。。。之间)

select \* from 表 where servlet>=90;

c）判空条件：

判断null： is null / is not null

判断空字符串： ='' / <>''

d）模糊条件： like

%: 替换任意个字符

\_: 替换一个字符

4.7 分页查询：limit 起始行,查询行数

起始行从0开始

4.8 排序： order by 字段 asc/desc

asc: 正序，顺序

desc：反序，倒序

4.9 分组查询：group by 字段

4.10: 分组后筛选： having 条件

SQL语句的分类：

DDL： 数据定义语言

create / drop / alter

DML：数据操作语句

insert / delete /update / truncate

DQL： 数据查询语言：

select / show

今天的目标：

大纲：

1）数据约束

2）数据库设计（表设计）

3）存储过程

4）触发器

5）mysql权限问题

# 2 数据约束

## 2.1什么数据约束

对用户操作表的数据进行约束

## 2.2 默认值

作用： 当用户对使用默认值的字段不插入值的时候，就使用默认值。

注意：

1）对默认值字段插入null是可以的。

2）对默认值字段可以插入非null

|  |
| --- |
| -- 1.1 默认值  CREATE TABLE student(  id INT,  NAME VARCHAR(20),  address VARCHAR(20) DEFAULT '广州天河' -- 默认值  )  DROP TABLE student;  -- 当字段没有插入值的时候，mysql自动给该字段分配默认值  INSERT INTO student(id,NAME) VALUES(1,'张三');  -- 注意：默认值的字段允许为null  INSERT INTO student(id,NAME,address) VALUE(2,'李四',NULL);  INSERT INTO student(id,NAME,address) VALUE(3,'王五','广州番禺'); |

## 2.3 非空

作用： 限制字段必须赋值

注意：

1）非空字符必须赋值

2）非空字符不能赋null

|  |
| --- |
| -- 1.2 非空  -- 需求： gender字段必须有值（不为null）  CREATE TABLE student(  id INT,  NAME VARCHAR(20),  gender VARCHAR(2) NOT NULL -- 非空  )  -- 非空字段必须赋值  INSERT INTO student(id,NAME) VALUES(1,'李四');  -- 非空字符不能插入null  INSERT INTO student(id,NAME,gender) VALUES(1,'李四',NULL); |

## 2.4 唯一

作用： 对字段的值不能重复

注意：

1）唯一字段可以插入null

2）唯一字段可以插入多个null

|  |
| --- |
| -- 1.3 唯一  CREATE TABLE student(  id INT UNIQUE, -- 唯一  NAME VARCHAR(20)  )  INSERT INTO student(id,NAME) VALUES(1,'zs');  INSERT INTO student(id,NAME) VALUES(1,'lisi'); -- ERROR 1062 (23000): Duplicate entry '1' for key 'id'  INSERT INTO student(id,NAME) VALUES(2,'lisi'); |

## 2.5 主键

作用： 非空+唯一

注意：

1）通常情况下，**每张表都会设置一个主键字段。**用于标记表中的每条记录的唯一性。

2）建议不要选择表的包含业务含义的字段作为主键，建议给每张表独立设计一个非业务含义的 id字段。

|  |
| --- |
| -- 1.4 主键（非空+唯一）  DROP TABLE student;  CREATE TABLE student(  id INT PRIMARY KEY, -- 主键  NAME VARCHAR(20)  )  INSERT INTO student(id,NAME) VALUES(1,'张三');  INSERT INTO student(id,NAME) VALUES(2,'张三');  -- INSERT INTO student(id,NAME) VALUES(1,'李四'); -- 违反唯一约束： Duplicate entry '1' for key 'PRIMARY'  -- insert into student(name) value('李四'); -- 违反非空约束： ERROR 1048 (23000): Column 'id' cannot be null |

## 2.6 自增长

作用： 自动递增

|  |
| --- |
| -- 1.5 自增长  CREATE TABLE student(  id INT(4) ZEROFILL PRIMARY KEY AUTO\_INCREMENT, -- 自增长，从0开始 ZEROFILL 零填充  NAME VARCHAR(20)  )  -- 自增长字段可以不赋值，自动递增  INSERT INTO student(NAME) VALUES('张三');  INSERT INTO student(NAME) VALUES('李四');  INSERT INTO student(NAME) VALUES('王五');  SELECT \* FROM student;  -- 不能影响自增长约束  DELETE FROM student;  -- 可以影响自增长约束  TRUNCATE TABLE student; |

## 2.7 外键

作用：约束两种表的数据

出现两种表的情况：

解决数据冗余高问题： 独立出一张表

例如： 员工表 和 部门表

问题出现：在插入员工表数据的时候，员工表的部门ID字段可以随便插入！！！！！

使用外键约束：约束插入员工表的部门ID字段值

解决办法： 在员工表的部门ID字段添加一个外键约束

|  |
| --- |
| -- 部门表（主表）  CREATE TABLE dept(  id INT PRIMARY KEY,  deptName VARCHAR(20)  )  -- 修改员工表（副表/从表）  CREATE TABLE employee(  id INT PRIMARY KEY,  empName VARCHAR(20),  deptId INT,-- 把部门名称改为部门ID  -- 声明一个外键约束  CONSTRAINT emloyee\_dept\_fk FOREIGN KEY(deptId) REFERENCES dept(id)  -- 外键名称 外键 参考表(参考字段)  ) |

注意：

1）被约束的表称为副表，约束别人的表称为主表，外键设置在副表上的！！！

2）主表的参考字段通用为主键！

**3）添加数据： 先添加主表，再添加副表**

**4）修改数据： 先修改副表，再修改主表**

**5）删除数据： 先删除副表，再删除主表**

|  |
| --- |
| -- 1.6 外键约束  -- 员工表  CREATE TABLE employee(  id INT PRIMARY KEY,  empName VARCHAR(20),  deptName VARCHAR(20) -- 部门名称  )  INSERT INTO employee VALUES(1,'张三','软件开发部');  INSERT INTO employee VALUES(2,'李四','软件开发部');  INSERT INTO employee VALUES(3,'王五','应用维护部');  SELECT \* FROM employee;  -- 添加员工，部门名称的数据冗余高  INSERT INTO employee VALUES(4,'陈六','软件开发部');  -- 解决数据冗余高的问题：给冗余的字段放到一张独立表中  -- 独立设计一张部门表  CREATE TABLE dept(  id INT PRIMARY KEY,  deptName VARCHAR(20)  )  DROP TABLE employee;  -- 修改员工表  CREATE TABLE employee(  id INT PRIMARY KEY,  empName VARCHAR(20),  deptId INT,-- 把部门名称改为部门ID  -- 声明一个外键约束  CONSTRAINT emlyee\_dept\_fk FOREIGN KEY(deptId) REFERENCES dept(id) ON UPDATE CASCADE ON DELETE CASCADE -- ON CASCADE UPDATE ：级联修改  -- 外键名称 外键 参考表(参考字段)  )  INSERT INTO dept(id,deptName) VALUES(1,'软件开发部');  INSERT INTO dept(id,deptName) VALUES(2,'应用维护部');  INSERT INTO dept(id,deptName) VALUES(3,'秘书部');  INSERT INTO employee VALUES(1,'张三',1);  INSERT INTO employee VALUES(2,'李四',1);  INSERT INTO employee VALUES(3,'王五',2);  INSERT INTO employee VALUES(4,'陈六',3);  -- 问题: 该记录业务上不合法，员工插入了一个不存在的部门数据  INSERT INTO employee VALUES(5,'陈六',4); -- 违反外键约束： Cannot add or update a child row: a foreign key constraint fails (`day16`.`employee`, CONSTRAINT `emlyee\_dept\_fk` FOREIGN KEY (`deptId`) REFERENCES `dept` (`id`))  -- 1）当有了外键约束，添加数据的顺序： 先添加主表，再添加副表数据  -- 2）当有了外键约束，修改数据的顺序： 先修改副表，再修改主表数据  -- 3）当有了外键约束，删除数据的顺序： 先删除副表，再删除主表数据  -- 修改部门(不能直接修改主表)  UPDATE dept SET id=4 WHERE id=3;  -- 先修改员工表  UPDATE employee SET deptId=2 WHERE id=4;    -- 删除部门  DELETE FROM dept WHERE id=2;  -- 先删除员工表  DELETE FROM employee WHERE deptId=2;  SELECT \* FROM dept;  SELECT \* FROM employee; |

## 2.8 级联操作

问题： 当有了**外键约束**的时候，必须先修改或删除副表中的所有关联数据，才能修改或删除主表！但是，我们希望直接修改或删除主表数据，从而影响副表数据。可以使用级联操作实现！！！

级联修改： ON UPDATE CASCADE

级联删除： ON DELETE CASCADE

|  |
| --- |
| CREATE TABLE employee(  id INT PRIMARY KEY,  empName VARCHAR(20),  deptId INT,-- 把部门名称改为部门ID  -- 声明一个外键约束  CONSTRAINT emlyee\_dept\_fk FOREIGN KEY(deptId) REFERENCES dept(id) ON UPDATE CASCADE ON DELETE CASCADE -- ON CASCADE UPDATE ：级联修改  -- 外键名称 外键 参考表(参考字段)  )  **注意： 级联操作必须在外键基础上使用** |

|  |
| --- |
| -- 级联修改（修改）  -- 直接修改部门  UPDATE dept SET id=5 WHERE id=4;  -- 级联删除  -- 直接删除部门  DELETE FROM dept WHERE id=1; |

# 3 数据库设计

## 3.1 引入

需求分析 - 需求分析师 -》 原始需求- > 抽取业务模型

图书模型：图书名称，版本号，作者

学生模型： 学号，学生姓名 手机号码

......

角色：学生 老师，图书管理员

《需求说明书》

需求设计 -

概要设计：

抽取实体：业务模型 -> 实体模型（java 类 c++类）内存

class Book{ name, bookNo,author }

**数据库设计:**

业务模型/实体模型 - > 数据模型 (硬盘)

数据库表设计

问题： 如何设计？

详细设计

类详细，属性和方法

## 3.2 三大范式

设计原则： 建议设计的表尽量遵守三大范式。

第一范式： 要求表的**每个字段必须是不可分割的独立单元**

student : name -- 违反第一范式

张小名|狗娃

sutdent ： name old\_name --符合第一范式

张小名 狗娃

第二范式： **在第一范式的基础上，要求每张表只表达一个意思**。表的每个字段都和表的主键有依赖。

employee（员工）: 员工编号 员工姓名 部门名称 订单名称 --违反第二范式

员工表：员工编号 员工姓名 部门名称

订单表： 订单编号 订单名称 -- 符合第二范式

第三范式： 在第二范式基础，要求每张表的主键之外的其他字段都只能和主键有直接决定依赖关系。

员工表： 员工编号（主键） 员工姓名 部门编号 部门名 --符合第二范式，违反第三范式 （数据冗余高）

员工表：员工编号（主键） 员工姓名 部门编号 --符合第三范式（降低数据冗余）

部门表：部门编号 部门名

# 4 关联查询(多表查询)

|  |
| --- |
| -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*二、关联查询（多表查询）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*----  -- 需求：查询员工及其所在部门(显示员工姓名，部门名称)  -- 2.1 交叉连接查询（不推荐。产生笛卡尔乘积现象：4 \* 4=16，有些是重复记录）  SELECT empName,deptName FROM employee,dept;  -- 需求：查询员工及其所在部门(显示员工姓名，部门名称)  -- 多表查询规则：1）确定查询哪些表 2）确定哪些哪些字段 3）表与表之间连接条件 (规律：连接条件数量是表数量-1)  -- 2.2 内连接查询：只有满足条件的结果才会显示(使用最频繁)  SELECT empName,deptName -- 2）确定哪些哪些字段  FROM employee,dept -- 1）确定查询哪些表  WHERE employee.deptId=dept.id -- 3）表与表之间连接条件    -- 内连接的另一种语法  SELECT empName,deptName  FROM employee  INNER JOIN dept  ON employee.deptId=dept.id;    -- 使用别名  SELECT e.empName,d.deptName  FROM employee e  INNER JOIN dept d  ON e.deptId=d.id;  -- 需求： 查询每个部门的员工  -- 预期结果：  -- 软件开发部 张三  -- 软件开发部 李四  -- 应用维护部 王五  -- 秘书部 陈六  -- 总经办 null  -- 2.2 左[外]连接查询： 使用左边表的数据去匹配右边表的数据，如果符合连接条件的结果则显示，如果不符合连接条件则显示null  -- （注意： 左外连接：左表的数据一定会完成显示！）  SELECT d.deptName,e.empName  FROM dept d  LEFT OUTER JOIN employee e  ON d.id=e.deptId;  -- 2.3 右[外]连接查询: 使用右边表的数据去匹配左边表的数据，如果符合连接条件的结果则显示，如果不符合连接条件则显示null  -- （注意： 右外连接：右表的数据一定会完成显示！）  SELECT d.deptName,e.empName  FROM employee e  RIGHT OUTER JOIN dept d  ON d.id=e.deptId;  -- 2.4 自连接查询  -- 需求：查询员工及其上司  -- 预期结果：  -- 张三 null  -- 李四 张三  -- 王五 李四  -- 陈六 王五  SELECT e.empName,b.empName  FROM employee e  LEFT OUTER JOIN employee b  ON e.bossId=b.id; |

# 5 存储过程

## 5.1 什么是存储过程

存储过程，带有逻辑的sql语句

之前的sql没有条件判断，没有循环

存储过程带上流程控制语句（if while）

## 5.2 存储过程特点

1）执行效率非常快！存储过程是在数据库的服务器端执行的！！！

2）移植性很差！不同数据库的存储过程是不能移植。

## 5.3 存储过程语法

|  |
| --- |
| -- 创建存储过程  DELIMITER $ -- 声明存储过程的结束符  CREATE PROCEDURE pro\_test() --存储过程名称(参数列表)  BEGIN -- 开始  -- 可以写多个sql语句; -- sql语句+流程控制  SELECT \* FROM employee;  END $ -- 结束 结束符  -- 执行存储过程  CALL pro\_test(); -- CALL 存储过程名称(参数);  参数：  IN： 表示输入参数，可以携带数据带存储过程中  OUT： 表示输出参数，可以从存储过程中返回结果  INOUT： 表示输入输出参数，既可以输入功能，也可以输出功能 |

|  |
| --- |
| -- \*\*\*\*\*\*\*\*\*\*\*\*\*\*三、存储过程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-  -- 声明结束符  -- 创建存储过程  DELIMITER $  CREATE PROCEDURE pro\_test()  BEGIN  -- 可以写多个sql语句;  SELECT \* FROM employee;  END $  -- 执行存储过程  CALL pro\_test();  -- 3.1 带有输入参数的存储过程  -- 需求：传入一个员工的id，查询员工信息  DELIMITER $  CREATE PROCEDURE pro\_findById(IN eid INT) -- IN: 输入参数  BEGIN  SELECT \* FROM employee WHERE id=eid;  END $  -- 调用  CALL pro\_findById(4);  -- 3.2 带有输出参数的存储过程  DELIMITER $  CREATE PROCEDURE pro\_testOut(OUT str VARCHAR(20)) -- OUT：输出参数  BEGIN  -- 给参数赋值  SET str='helljava';  END $  -- 删除存储过程  DROP PROCEDURE pro\_testOut;  -- 调用  -- 如何接受返回参数的值？？  -- \*\*\*mysql的变量\*\*\*\*\*\*  -- 全局变量（内置变量）：mysql数据库内置的变量 （所有连接都起作用）  -- 查看所有全局变量： show variables  -- 查看某个全局变量： select @@变量名  -- 修改全局变量： set 变量名=新值  -- character\_set\_client: mysql服务器的接收数据的编码  -- character\_set\_results：mysql服务器输出数据的编码    -- 会话变量： 只存在于当前客户端与数据库服务器端的一次连接当中。如果连接断开，那么会话变量全部丢失！  -- 定义会话变量: set @变量=值  -- 查看会话变量： select @变量    -- 局部变量： 在存储过程中使用的变量就叫局部变量。只要存储过程执行完毕，局部变量就丢失！！  -- 1)定义一个会话变量name, 2)使用name会话变量接收存储过程的返回值  CALL pro\_testOut(@NAME);  -- 查看变量值  SELECT @NAME;  -- 3.3 带有输入输出参数的存储过程  DELIMITER $  CREATE PROCEDURE pro\_testInOut(INOUT n INT) -- INOUT： 输入输出参数  BEGIN  -- 查看变量  SELECT n;  SET n =500;  END $  -- 调用  SET @n=10;  CALL pro\_testInOut(@n);  SELECT @n;  -- 3.4 带有条件判断的存储过程  -- 需求：输入一个整数，如果1，则返回“星期一”,如果2，返回“星期二”,如果3，返回“星期三”。其他数字，返回“错误输入”;  DELIMITER $  CREATE PROCEDURE pro\_testIf(IN num INT,OUT str VARCHAR(20))  BEGIN  IF num=1 THEN  SET str='星期一';  ELSEIF num=2 THEN  SET str='星期二';  ELSEIF num=3 THEN  SET str='星期三';  ELSE  SET str='输入错误';  END IF;  END $  CALL pro\_testIf(4,@str);    SELECT @str;  -- 3.5 带有循环功能的存储过程  -- 需求： 输入一个整数，求和。例如，输入100，统计1-100的和  DELIMITER $  CREATE PROCEDURE pro\_testWhile(IN num INT,OUT result INT)  BEGIN  -- 定义一个局部变量  DECLARE i INT DEFAULT 1;  DECLARE vsum INT DEFAULT 0;  WHILE i<=num DO  SET vsum = vsum+i;  SET i=i+1;  END WHILE;  SET result=vsum;  END $  DROP PROCEDURE pro\_testWhile;  CALL pro\_testWhile(100,@result);  SELECT @result;  USE day16;  -- 3.6 使用查询的结果赋值给变量（INTO）  DELIMITER $  CREATE PROCEDURE pro\_findById2(IN eid INT,OUT vname VARCHAR(20) )  BEGIN  SELECT empName INTO vname FROM employee WHERE id=eid;  END $  CALL pro\_findById2(1,@NAME);  SELECT @NAME; |

# 6 触发器

## 6.1 触发器作用

当操作了某张表时，希望同时触发一些动作/行为，可以使用触发器完成！！

例如： 当向员工表插入一条记录时，希望同时往日志表插入数据

|  |
| --- |
| -- 需求： 当向员工表插入一条记录时，希望mysql自动同时往日志表插入数据  -- 创建触发器(添加)  CREATE TRIGGER tri\_empAdd AFTER INSERT ON employee FOR EACH ROW -- 当往员工表插入一条记录时  INSERT INTO test\_log(content) VALUES('员工表插入了一条记录');    -- 插入数据  INSERT INTO employee(id,empName,deptId) VALUES(7,'扎古斯',1);  INSERT INTO employee(id,empName,deptId) VALUES(8,'扎古斯2',1);  -- 创建触发器(修改)  CREATE TRIGGER tri\_empUpd AFTER UPDATE ON employee FOR EACH ROW -- 当往员工表修改一条记录时  INSERT INTO test\_log(content) VALUES('员工表修改了一条记录');    -- 修改  UPDATE employee SET empName='eric' WHERE id=7;    -- 创建触发器(删除)  CREATE TRIGGER tri\_empDel AFTER DELETE ON employee FOR EACH ROW -- 当往员工表删除一条记录时  INSERT INTO test\_log(content) VALUES('员工表删除了一条记录');    -- 删除  DELETE FROM employee WHERE id=7; |

# 7 mysql权限问题

|  |
| --- |
| -- \*\*\*\*\*\*\*\*\*\*\*五、mysql权限问题\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  -- mysql数据库权限问题：root ：拥有所有权限（可以干任何事情）  -- 权限账户，只拥有部分权限（CURD）例如，只能操作某个数据库的某张表  -- 如何修改mysql的用户密码？  -- password: md5加密函数(单向加密)  SELECT PASSWORD('root'); -- \*81F5E21E35407D884A6CD4A731AEBFB6AF209E1B    -- mysql数据库，用户配置 : user表  USE mysql;  SELECT \* FROM USER;  -- 修改密码  UPDATE USER SET PASSWORD=PASSWORD('123456') WHERE USER='root';  -- 分配权限账户  GRANT SELECT ON day16.employee TO 'eric'@'localhost' IDENTIFIED BY '123456';  GRANT DELETE ON day16.employee TO 'eric'@'localhost' IDENTIFIED BY '123456'; |