1. gMysql 中distinct 应用于单列去重；并且只能放在开头 例如：SELECT DISTINCT gender FROM student;
2. <>表示不等于
3. 判断是null is null；
4. 判断空字符串 where a=’’
5. 模糊条件：like %表示一个任意字符串；\_表示一个字符；
6. 聚合查询： sum() avg() max() min() count() 其中count()不包含null的数据，即在使用时要统计不包含null的字段；
7. 分页查询 （limit 起始行，查询几行）；分页查询当前页：SELECT \* FROM student LIMIT (当前页-1)\*每页显示多少条,每页显示多少条;
8. Order by 语法： order by 字段 asc/desc; 多个排序条件：order by 字段 asc/desc，字段 asc/desc；
9. 分组查询 group by 条件： having 在group by 后使用； where 在分组前使用；例如：select name,count(\*) from student where sid<>1 group by name；

select name,count(\*) from student group by name having sid<>1; 两者的区别是后者将包含sid为1的组删除 注意：分组必须在排序和分页之前

注意：group by 后面的字段必须和显示声明字段一致，即select后有多少个字段group by 后就包含多少个字段（聚合函数的除外）。

例如：

SELECT

bsm, DONGJ, BEIW,DIMLX, "MIN"(mediapath) mediapath ，"MIN"(列名)

from d

GROUP BY bsm, DONGJ, BEIW,DIMLX

数据约束

1. 非空 not null ；
2. 唯一 unique 字段值不能重复；
3. 主键primary key 表示非空并且唯一 建议给每张表独立设计非业务含义的id；
4. 自增长 auto\_increment;
5. 外键 constraint 外键名称 foreign key（外键） rferences 参考表名（参考字段）；-- 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。删除修改时先修改副表。

数据库设计

设计建议遵守三大范式：

1. 要求表中每个字段都是不可分割的独立单元 例如 name 和 oldname；
2. 要求每张表只表达一个意思，即表的每个字段和表的主键有依赖 如员工编号 员工名称 部门名称 订单名称 这个表中的订单名称可抽出来新建一个表；
3. 要求表中主键之外的其他字段只能和主键有直接决定依赖关系；即降低数据冗余 例如： 员工编号 员工名称 部门名称 其中部门名称和员工没有直接关系，将部门名称改成部门编号；

多表查询

1. 内连接查询

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、

另一种语法：

Select empName ,deptName

From employee e

Inner join dept d

On e.deptid = d.id;

1. 左外连接查询：

下面的例子中 dept 为左表 employee为右表 左表字段值全显。右表中符合on条件的数据查出来，

Select d.deptName, e,empName

From dept d

Left outer join employee e

On d.id = e.deptid;

1. 右外连接

SELECT d.deptName,e.empName

FROM employee e

RIGHT OUTER JOIN dept d

ON d.id=e.deptId;

存储过程

Delimiter $ --声明存储过程的结束符

Create procedure pro\_test(IN id int,OUT str varchar(20)) --存储过程名称 （参数列表）

Begin

可写多条sql并添加流程控制语句；

End $

Call pro\_test(); 执行存储过程

参数：in 输入参数可携带数据进入过程中 in相当于类型 使用时

Out 输出参数 携带返回结果

Inout 输入输出参数

Mysql 变量：

全局变量：MySQL内置的变量（主要定义了mysql 的基本配置）所有连接均起作用

查看所有全局变量：show variables

查看部分全局变量：show variables like ‘%%’

查看某个变脸 ：select @@变量名

设置全局变量 set 变量名 = 新值

例如：

-- 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：客户端与服务器一次连接中起作用

定义会话变量 set @变量名 = 值

查看会话变量 ： select @变量；

局部变量

-- 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;

触发器：当操作某张表是，希望同时触发一些动作，如向员工表中插入一条记录时，希望同时往日志表中插入数据；

-- 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;

-- 修改mysql 密码

UPDATE USER SET PASSWORD=PASSWORD('123456') WHERE USER='root';

SQL高级应用：

On 、where 、having条件查询区别：

On 在创建临时表（统计）之前进行过滤 用在两个表连接时使用的过滤

Where 在创建临时表后进行过滤，

Having 对where 过滤后数据进行分组，在使用having进行过滤。

在多表联接查询时，on比where更早起作用。系统首先根据各个表之间的联接条件，把多个表合成一个临时表后，再由where进行过滤，然后再计算，计算完后再由having进行过滤。由此可见，要想过滤条件起到正确的作用，首先要明白这个条件应该在什么时候起作用，然后再决定放在那里

理论上 on where having 的速度是一次降低的。

Distinct 去重：

单列去重： 缺点只能查询单列的数据

例子 ：SELECT DISTINCT GCMC from hpf\_bhgc

多列去重：缺点必须多列数据均相同时才会判断为重复。

例子：SELECT DISTINCT GCMC, GCFL from hpf\_bhgc 当两个字段均一样时，才过滤掉。

查询多列，去重指定字段使用

Select name,min(其他列名) from tableName group by name；