```
--单行注释

#单行注释

/**/多行注释



--创建并连接数据库

cd C:\xampp\mysql\bin


--登录账号密码

--学校密码 123456，自己电脑空白

--mysql -uroot -p123456 mydb;

mysql -u root -p


--创建数据库(mydb 为自定义数据库名称)

CREATE databases mydb;

--防止创建的数据库存在，存在会返回警告信息

CREATE DATABASE IF NOT EXISTS mydb;


--查看警告信息

show warnings;


--查看数据库

show databases;

--查看指定的数据库
```

```
show CREATE DATABASE  mydb;

--选择数据库
USE mydb;

--删除数据库
DROP DATABASE mydb;
DROP DATABASE IF EXISTS mydb;



--查看所有数据表
show tables;
--查看名称中有 new 的数据表 show tables[like 匹配模式]
--%匹配一个或多个字符，_仅匹配一个字符
show tables  LIKE '%new%';

--创建数据表
CREATE TABLE 数据表名();
--temporary 创建临时表，仅当前字段可见
--字段属性，某些约束属性
CREATE [temporary] TABLE [IF NOT EXISTS] 数据表名 (
字段名 字段类型 [字段属性],
字段名 字段类型 [字段属性]
```

```
)[表选项];

--查看数据表详细内容
SELECT * FROM 数据表名;

--查看表结构
--DESCRIBE 数据表名;
DESC 数据表名;
show CREATE TABLE 数据表名;

--修改数据表
--[to/as]可省略
ALTER TABLE 旧表名 rename [to/as] 新表名;
--同时修改多个数据表
rename TABLE 旧表名 1 TO 新表名 1[,旧表名 2 TO 新表名 2]...

--修改表结构
--将数据表 goods 中名为 aaa 的字段修改为 des
--ALTER TABLE goods change aaa des varchar(255);
ALTER TABLE 数据表名 change[字段属性] 旧字段名 新字段名 字段类型[字段属性];

--修改表内容字段类型
--将数据表 goods 中 des 的字段的类型 varchar(255)修改为 char(55)
```

```
--ALTER TABLE goods modify des char(55);
ALTER TABLE 数据表名  modify[column] 字段名 新类型 [字段属性];

--移动表字段
--将 goods 表中最后一个字段 des 移动到 name 后
--ALTER TABLE goods modify des VAR(55) after name;
--将字段名 1 调整为数据表的第一个字段
ALTER TABLE 数据表名 modify[column] 字段名 1 数据类型 [字段属性] first
字段名 2;
--将字段名 1 插入字段名 2 的后面
ALTER TABLE 数据表名 modify[column] 字段名 1 数据类型 [字段属性] after
字段名 2;

--新增一个字段(默认添加到表的最后)
ALTER TABLE 数据表名 ADD[column] 新字段名 字段类型 [first 字段名];
ALTER TABLE 数据表名 ADD[column] 新字段名 字段类型 [after 字段名];
--新增多个字段(多个字段无法指定添加位置)
ALTER TABLE 数据表名 add[column] (新字段名 1 字段类型 1,新字段名 2 字段类
型 2,...)

--删除字段
ALTER TABLE 数据表名 DROP 字段名;

--删除数据表
```

```
--IF EXISTS 用于删除一个不存在的数据表时防止产生错误
DROP[temporary] TABLE[IF EXISTS] 数据表 1,数据表 2;

--添加数据
INSERT INTO 数据表名 VALUES
('1','2');

--只查看数据表编号(sno)为 u004 的详细信息
SELECT * FROM goods WHERE sno = 'u004';

--修改 u004 的会员信息(svip)
UPDATE goods SET svip = 1000 WHERE sno = 'u004';

--删除表数据编号(sno)为 u005 信息
DELETE FROM goods WHERE sno = 'u005';

--查询表中年龄超过 60 岁的抗疫英雄姓名(name)和年龄(age)。
--SELECT name,age  FROM goods WHERE age >= '60';
SELECT name,age  FROM goods WHERE age >60;

--查询表中最大年龄的抗疫英雄
--全部信息
SELECT * FROM goods WHERE age = (SELECT max(age) FROM goods);
--单独姓名
```

```
SELECT name FROM goods WHERE age = (SELECT max(age) FROM goods);

--查询表中获得“人民英雄”国家荣誉称号(honor)的抗疫英雄人数
--*代表全部
SELECT COUNT(*) FROM goods WHERE honor = '人民英雄';

--查询年龄(age)的最大值
SELECT max(age) FROM goods;

--允许中文输入
engine=innodb DEFAULT charset = utf8;

--数据类型 字节数 无符号的取值范围 有符号取值范围
TINYINT   1  0～255                    -128～127
SMALLINT  2  0～65535                  -32768～32767
MEDIUMINT 3  0～16777215               -8388608～8388607
INT       4  0～4294967295             -2147483648～2147483647
BIGINT    8  0～18446744073709551615   -9223372036854775808～
9223372036854775807

--设置零填充
--若数值宽度小于显示宽度，会在左侧填充 0
ZEROFILL
```

--数据类型浮点数 字节数 负数取值范围 非负数取值范围

--FLOAT 的精度大约 6~7 位，DOUBLE 的精度大约 15 位左右。

FLOAT 4 -3.402823466E+38 ~-1.175494351E-

38 0 和 1.175 494 351E-38 ~3.402 823

466E+38

DOUBLE 8 -1.797 693 134 862 315 7E+308 ~ -2.225 073 858 507 201

4E-308 0 和 2.225 073 858 507 201 4E-308 ~1.797 693 134 862 315

7E+308

--定点数类型通过 DECIMAL(M,D)设置位数和精度

--DECIMAL(5,2)表示的取值范围是-999.99~999.99。

DECIMAL

--二进制数：在二进制字符串前加前缀 b。b'1000001'

--十六进制数：有两种表示方式，形如“x'41'”和“0x41”。

--转义字符：在字符前加“\”转义。

--"ab\"c" ab"c

--时间和日期类型 取值范围 日期格式 零值

YEAR 1901~2155 YYYY

0000

DATE 1000-01-01~9999-12-3 YYYY-MM-

DD 0000-00-00

TIME -

838:59:59~838:59:59 HH:MM:SS 00:00:00

DATETIME 1000-01-01 00:00:00~9999-12-31 23:59:59 YYYY-MM-DD

HH:MM:SS 0000-00-00 00:00:00

TIMESTAMP 1970-01-01 00:00:01~2038-01-19 03:14:07 YYYY-MM-DD

HH:MM:SS 0000-00-00 00:00:00

CURRENT_DATE 或者 NOW() 输入当前系统日期

--数据类型 类型说明

CHAR() 固定长度字符串

VARCHAR() 可变长度字符串

TEXT 大文本数据

ENUM('值 1', '值 2', '值 3', …, '值 n') 枚举类

型 ENUM 类型的数据只能从枚举列表中取，并且只能

取一个

SET('值 1', '值 2', '值 3', …, '值 n') 字符串对

象 set 可以从列表中选择一个或多个值来保存，多个值

之间用逗号“,”分隔

BINARY 固定长度的二进制数据

VARBINARY 可变长度的二进制数据

BLOB 二进制大对象（Binary Large

Object）

```
--默认约束：为数据表中的字段指定默认值。
字段名 数据类型  DEFAULT 默认值;
--删除默认约束 UNSIGNED
ALTER TABLE 表名 MODIFY age INT UNSIGNED;
--添加默认约束
ALTER TABLE 表名 MODIFY age INT UNSIGNED DEFAULT 18;

--非空约束：字段的值不能为 NULL。
字段名 数据类型 NOT NULL;

--唯一约束：保证数据表中字段的唯一性，即表中字段的值不能重复出现。
--列级约束
字段名 数据类型 UNIQUE;
--表级约束
UNIQUE(字段名 1, 字段名 2, …);
--添加唯一约束
ALTER TABLE 表名 ADD UNIQUE 字段名;
--删除唯一约束
ALTER TABLE 表名 DROP INDEX 字段名;
--创建测试表，添加复合唯一键
UNIQUE(字段名 1, 字段名 2)
--只有当两个字段同时发生重复时，插入记录失败
--(1,4)(1,3)可以录入，再次录入(1.4)失败
```

```
--主键约束，主键的作用：唯一标识表中的记录。
--列级约束
字段名 数据类型 PRIMARY KEY
--表级约束
PRIMARY KEY (字段名 1, 字段名 2, …)
--删除主键约束
ALTER TABLE 表名 DROP PRIMARY KEY;
--添加主键约束
ALTER TABLE 表名 ADD PRIMARY KEY (字段名);

----自动增长
字段名 数据类型 AUTO_INCREMENT
--修改自动增长值
ALTER TABLE 表名 AUTO_INCREMENT = 10;
--删除 id 自动增长
ALTER TABLE 表名 MODIFY id 字段类型 UNSIGNED;
--重新为 id 添加自动增长
ALTER TABLE 表名 MODIFY id 字段类型 UNSIGNED AUTO_INCREMENT;

--select 语句
SELECT [ALL|DISTINCT] * | 列名 1[,列名 2,……列名 n] FROM 表名
[WHERE 条件表达式] [GROUP BY 列名 [ASC | DESC]
[HAVING 条件表达式]]
[ORDER BY 列名  [ASC | DESC] , ...]
```

```
[LIMIT [OFFSET] 记录数];
--WHERE 子句：用于指定查询筛选条件。
--GROUP BY 子句：用于将查询结果按指定的列进行分组；其中 HAVING 为可选参数，用于对分组后的结果集进行筛选。
--ORDER BY 子句：用于对查询结果集按指定的列进行排序。
--LIMIT 子句：用于限制查询结果集的行数。参数 OFFSET 为偏移量，当 OFFSET 值为 0 时，表示从查询结果的第 1 条记录开始，如果 OFFSET 为 1 时，表示查询结果从第 2 条记录开始。

--as 可以用于备注表列名
--查询 Goods 中每件商品的销售总价，其中销售总价=销售数量*价格，显示商品名称和销售总价。
SELECT gdName AS 商品名称,gdPrice*gdSaleQty AS 销售总价 FROM goods;
--查询 Users（用户信息表）中，用户名和年龄。
SELECT uName AS 用户名,YEAR(NOW())-YEAR(uBirth) AS 年龄 FROM users;

--可使用 BETWEEN AND 来限制查询数据的范围
WHERE 表达式 [NOT] BETWEEN 初始值 AND 终止值
--查询 Users 表中 2000 年后出生的，且性别为“男”的用户姓名，电话号码，出生年月。
SELECT  uName AS 姓名,uPhone AS 电话号码, uBirth AS 出生日期 FROM Users
WHERE uBirth>=2000-1-1 AND uSex='男';
```

```
--查询 Goods 表中 gdCity 值为“长沙或“西安”，且 gdPriced 小于等于 50 的商品名称,商品价格。
SELECT gdName AS 商品名称 ,gdPrice AS 商品价格 FROM goods
WHERE  gdPrice<=50  AND  (gdCity='长沙' OR gdCity='西安' );
--查询 Goods 表中 gdPriced 在 100 到 500 元的商品名称,商品价格
SELECT gdName AS 商品名称 ,gdPrice AS 商品价格 FROM goods WHERE
gdPrice BETWEEN 100 AND 500;


--IN 运算符与 BETWEEN…AND 运算符类似，用来限制查询数据的范围
WHERE 表达式 [NOT] IN (值 1,值 2…值 N)
--查询 Goods 表中 gdCity 为长沙、西安、上海三个城市的商品名称,商品价格
SELECT gdName AS 商品名称 ,gdPrice AS 商品价格  FROM goods WHERE
gdcity IN('长沙','西安','上海');


--使用 LIKE 运算符，实际中当需要查询的条件只能提供不完全确定的部分信息时，就需要使用 LIKE 运算符实现字符串的模糊查询
WHERE 列名 [NOT] LIKE ‘字符串’ [ESCAPE ‘转义字符’]
--通配符          说明                  示例
                任意字符串          s%：表示查询以 s 开头的任意字符串，如 small
                                    %s：表示查询以 s 结尾的任意字符串，如 address
```

%		%s%：表示查询包含 s 的任意字符串，如 super、course
_	任何单个字符	_s：表示查询以 s 结尾且长度为 2 的字符串，如 as s_：表示查询以 s 开头且长度为 2 的字符串，如 sa

```
--查询 Users 表中 gdName 为“李”开头的用户姓名、性别和手机号
SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users
WHERE uName LIKE '李%';
--查询 Users 表中 gdName 第 2 个字为“湘”的用户姓名、性别和手机号
SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users
WHERE uName LIKE '_湘%';
--查询 Goods 表中 gdName 以“华为 P9_”开头的 gdCode
SELECT gdCode FROM goods WHERE gdName LIKE '华为 P9\_%';
```

```
--使用 IS NULL 运算符，系统自动将其设置为空值
WHERE 列名 IS [NOT] NULL
--查询 Users 表中 uImage 为空的用户姓名和性别
SELECT uName AS 用户姓名,usex AS 性别 FROM users WHERE uimage IS NULL;
```

```
--使用 DISTINCT 消除重复结果集
--查询 Goods 表中 gdPrice 大于 200 的商品来源哪些城市
SELECT DISTINCT gdname AS 商品名称,gdCity AS 来源城市 FROM goods WHERE
gdPrice >200;


--使用 REGEXP 运算符
WHERE 列名 REGEXP '模式串'
--模式      说明                                    示例
^           匹配字符串的开始位置                    '^d': 匹配以字母
d 开头的字条串，如 dear，do
$           匹配字符串的结束位置                    'st$': 匹配以 st
结束的字符串，如 test，resist
.           匹配除 "\n" 之外的任何单个字符          'h.t': 匹配任何 h
和 t 间的一个字符，如 hit，hot
[...]       匹配字符集合中的任意一个字符            '[ab]': 匹配 ab 中
的任意一个字符， 如: plain，hobby
[^..]       匹配非字符集合中的任意一个字符          '[^ab]': 匹配任何
不包含 a 或 b 的字符串
p1|p2|p3    匹配 p1 或 p2 或 p3                     'z|food' : 匹配
"z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。
*           匹配零个或多个在它前面的字符           zo* : 匹配 "z"
以及 "zoo"。* 等价于{0,}。
+           匹配前面的字符 1 次或多次               'zo+' : 匹配
"zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
```

{n}　　匹配前面的字符串至少 n 次，n 是一个非负整数　　'o{2}' ：匹配 "food" 中的两个 o，但不能匹配 "Bob" 中的 'o'

{n,m}　　匹配前面的字符串至少 n 次，至多 m 次，m 和 n 均为非负整数，其中 n <= m　　'o{2,4}' ：匹配至少 2 个 o，最多 4 个 o 的字符串。如 oo，oooo

```
--查询 Users 表中 uPhone 以“5”结尾用户的姓名，性别和电话
SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users
WHERE uphone REGEXP '5$';
--查询 Users 表中 uPhone 以“16,17,18”开头用户的姓名，性别和电话
SELECT uName AS 用户姓名,usex AS 性别,uPhone AS 手机号 FROM users
WHERE uphone REGEXP '^16|17|18';
```

```
--数据排序
ORDER BY {列名 | 表达式 | 正整数} [ASC | DESC] [,…n]
--查询 Goods 表 tID 为 1 的商品编号、名称和价格，并按价格升序排列
SELECT gdCode AS 商品编号 ,gdName AS 商品名称,gdPrice AS 商品价格 FROM goods WHERE tID=1 ORDER BY gdPrice;
--查询 Goods 表 tID 为 1 的商品编号、名称、价格和销售量，并先按销售量降序，再价格升序排列
```

```
SELECT gdCode AS 商品编号 ,gdName AS 商品名称,gdPrice AS 商品价格,gdSaleQty AS 销售量 FROM goods WHERE tID=1 ORDER BY gdSaleQty DESC,gdPrice;


--使用 LIMIT 限制结果集返回的行数
--SELECT 语句中使用 LIMIT 子句来指定查询结果从哪一条记录开始以及一共查询多少行记录
--查询 Goods 表前 3 行记录的商品编号、名称和价格。
SELECT gdCode AS 商品编号,gdName AS 商品名称,gdPrice AS 商品价格 FROM goods LIMIT 3;


--聚合函数
-- 函数名        说明
SUM             返回表达式中所有值的和
MAX             返回表达式中的最大值
COUNT           返回组中的项数
AVG             返回组中各值的平均值
MIN             返回表达式中的最小值
GROUP_CONCAT    返回一个字符串结果，该结果由分组中的值连接组合而成
-- 查询 Goods 表，统计所有商品的总销售量。
SELECT SUM(gdSaleQty) FROM goods;
```

```
-- 查询 Goods 表，显示商品的最高价格。
SELECT MAX(gdprice) FROM goods;
-- 查询 Orders 表，显示购买过商品的用户人数。
SELECT COUNT(DISTINCT uID) FROM orders;


--数据分组统计 GROUP BY 子句
GROUP BY [ ALL ] 列名 1,列名 2 [ ,...n ] [ WITH ROLLUP] [HAVING 条件表
达式]
--GROUP BY 和 GROUP_CONCAT 函数一起使用
--DISTINCT 可以排除重复值
GROUP_CONCAT([DISTINCT] 表达式 [ORDER BY 列名] [SEPARATOR 分隔符])

-- 查询 Users 表，按 uCity 列进行分组
SELECT uID,uName,uSex,uCity FROM users GROUP BY uCity;
-- 查询 Users 表，统计各城市的用户人数。
SELECT uCity,COUNT(*) FROM users GROUP BY uCity;
-- 查询 Users 表，将同一城市的 uID 值用逗号“,”连接起来，列名为 uIDs。
SELECT uCity,GROUP_CONCAT(uID) AS uIDs FROM users GROUP BY uCity;
-- 查询 Users 表，将同一城市的 uID 值用下划线“_”连接起来，列名为 uIDs。
SELECT uCity,GROUP_CONCAT(uid ORDER BY uid SEPARATOR '_')  AS uids
FROM users GROUP BY ucity;

--GROUP BY 和 WITH ROLLUP 一起使用，可以输出每一类分组的汇总值
-- 查询 Users 表，统计“上海”和“长沙”两地用户人数。
```

```sql
SELECT uCity,COUNT(*) FROM users WHERE uCity IN ('长沙','上海') GROUP
BY uCity WITH ROLLUP;


--GROUP BY 和 HAVING 一起使用
--HAVING 关键字和 WHERE 关键字都用于设置条件表达式对查询结果集进行筛选
--HAVING 关键字后可以跟聚合函数，且只能跟 GROUP BY 一起使用
-- 查询 Users 表，统计各城市的用户人数，显示人数在 3 人以上的城市。
SELECT uCity,COUNT(*) FROM users GROUP BY uCity HAVING COUNT(*)>=3;


-- 视图操作
-- 插入视图
CREATE VIEW v_student
AS
SELECT sno,sname,ssex FROM student;


-- 插入一条语句
INSERT INTO v_student
VALUE('22140339','xmy','男');


-- 查询视图
SELECT * FROM v_student WHERE sno = '22140339';


-- 删除视图
-- 删除单个
```

```sql
DROP VIEW v_student;
-- 删除多个
DROP VIEW v_student,v_app,v_sst;
-- 删除视图指定内容
DELETE FROM v_student WHERE sno = '2007050122';

-- 修改视图
ALTER VIEW v_student
AS
SELECT sno,sname FROM student;

-- 查询 20070304 班选修大学英语的课程且成绩在 80-90 分的学生姓名，学号，班级号及成绩
CREATE VIEW v_stu2
AS
SELECT sname,A.sno,classno,degree FROM
student A, sc B, course C
WHERE A.sno = B.sno AND B.cno = C.cno
AND classno = '20070304'
AND cname = '大学英语'
AND degree BETWEEN 80 AND 90;
```

```
-- 创建一个名为 sc_view1 的视图，从数据库 gradem 的 sc 表中查询出成绩大于 90 分的所有学生选修课程成绩的信息。
CREATE VIEW sc_view1
AS
SELECT * FROM sc WHERE degree > 90;

-- 创建一个名为 sc_view2 的视图，从数据库 gradem 的 sc 表中查询出成绩小于 80 分的所有学生的学号、课程号、成绩等信息。
CREATE VIEW sc_view2
AS
SELECT sno,cno,degree FROM sc WHERE degree <80;

-- 创建一个名为 sc_view3 的视图，由数据库 gradem 的 student、course、sc 表创建一个显示“20070304”班学生选修课程（包括学生姓名、课程名称、成绩等信息）的视图。
CREATE VIEW sc_view3
AS
SELECT sname,cname,degree
FROM student a ,sc b,course c
WHERE a.sno = b.sno
AND b.cno = c.cno
AND classno = '20070304';
```

```sql
-- 创建一个从视图 sc_view1 中查询出课程号“c01”的所有学生的视图。
CREATE VIEW view_new
AS
SELECT * FROM sc_view1 WHERE cno='c01';


CREATE INDEX u_1 ON users (uname);
SHOW INDEX FROM users;


-- 索引操作
-- 显示索引
SHOW INDEX FROM student;

-- 为商品类别表的名称字段建立普通索引。
CREATE INDEX i_1 ON goodstype (tname);

-- 为用户表的用户名字段建立唯一索引 UNIQUE。
CREATE UNIQUE INDEX u_1 ON users(uname);

-- 为商品表的商品编号和商品名称创建普通的复合索引。
CREATE INDEX id_1 ON goods (gdid,gdname);

-- 为用户表的电子邮箱建立索引，降序。
CREATE INDEX e_1 ON users(uEmail DESC);
```

```sql
-- 删除以上索引。
DROP INDEX i_1 ON goodstype;
DROP INDEX id_1 ON goods;
DROP INDEX e_1 ON users;

-- 事务处理操作
-- 开启事务
START TRANSACTION;

-- 将 Alex 用户的 500 元钱转给 bill 用户，转账
-- 失败 ROLLBACK
START TRANSACTION;
UPDATE sh_user SET money = money - 500 WHERE NAME = 'Alex';
UPDATE sh_user SET money = money + 500 WHERE NAME = 'Bill';
ROLLBACK;

-- 成功 commit;
START TRANSACTION;
UPDATE sh_user SET money = money - 50 WHERE NAME = 'Alex';
UPDATE sh_user SET money = money + 50 WHERE NAME = 'Bill';
COMMIT;

-- 开启事务
```

```
START TRANSACTION;

-- 录入订单信息 sh_order_goods，购买 120 台笔记本电脑，id 为 4
INSERT INTO sh_order_goods(id,goods_id,goods_num)
VALUE (1,4,120);

-- 更新商品表 sh_goods，看看库存够不够
UPDATE sh_goods SET stock = stock - 120 WHERE id = 4;

-- 确认事务（确认/回滚）
ROLLBACK;

-- 存储过程操作
-- 创建存储过程(//)($$)
DELIMITER//
CREATE PROCEDURE aaa()
BEGIN
SELECT sno,sname,classno,saddress FROM student WHERE saddress LIKE
'%青岛%';
END//

-- 调用存储过程
CALL aaa;
```

```sql
-- rj347x 中获取名字相同人的信息

DELIMITER//

CREATE PROCEDURE bbb()

BEGIN

SELECT 学号,姓名,城市 FROM rj347x

WHERE 姓名 IN (

SELECT 姓名 FROM rj347x

GROUP BY 姓名

HAVING COUNT(*)>1

);

END//


CALL bbb;


-- 创建一个存储过程 T1，统计某位同学的考试门数（参数 in out）

-- in

DELIMITER //

CREATE PROCEDURE t1( IN T1_sno CHAR(10))

BEGIN

SELECT COUNT(*) FROM sc WHERE sno = T1_sno;

END //


CALL t1('2007010101');
```

```
-- out
DELIMITER //
CREATE PROCEDURE t2( IN t_sno CHAR(10),OUT t_num INT)
BEGIN
SELECT COUNT(*) INTO t_num FROM sc WHERE sno = t_sno;
END //

CALL t2('2007010101',@num);
SELECT @num;


-- 创建触发器 T_1,当向 orderdetail 表插入一条记录时，order 表对应的 ototal 的值增加，增加的值为订单详细中对应商品的数量
 DELIMITER//
 CREATE TRIGGER T_1
 AFTER INSERT ON orderdetail FOR EACH ROW
 BEGIN
 UPDATE orders SET oTotal = oTotal + new.odnum
 WHERE orders.oid = new.oid;
 END//

 INSERT INTO orderdetail
 VALUES (13,1,1,1,'sdfsdf',NOW());

-- 变量
```

```
--查看全局变量的值
--例：查看当前的版本信息
SELECT @@version AS '当前 MySQL',CURRENT_DATE;
-- 查看全局变量
SHOW GLOBAL variables;
SHOW  variables WHERE variable_name LIKE 'collation%';


-- 用户变量
SET @变量名 =表达式;
SET @变量名 :=表达式;
SELECT @变量名 :=表达式;



-- 局部变量，必须在存储过程中
-- 定义一个整型局部变量 iAge 和可变长字符型局部变量 vAddress，并分别赋值 20
和“中国山东”，最后输出变量的值
DELIMITER//
CREATE PROCEDURE bb()
BEGIN
```

```
DECLARE iAge INT;
DECLARE vAddress LONGTEXT;
SET iAge =20;
SET vAddress = '中国山东';
SELECT iAge,vAddress;
END//

CALL bb();


-- 循环语句
-- WHILE 语句
-- 使用 WHILE 语句求 1～100 之和
DELIMITER//
CREATE PROCEDURE aa()
BEGIN
SET @i:=1,@num:=0;
WHILE @i<=100 DO
BEGIN
SET @num=@num+@i;
SET @i=@i+1;
END;
END WHILE;
SELECT @num;
END//
```

```
CALL aa();


-- REPEAT 语句
-- 使用 REPEAT 语句求 1～100 之和
DELIMITER//
CREATE PROCEDURE bb()
BEGIN
SET @i=1,@num=0;
REPEAT
BEGIN
SET @num=@num+@i;
SET @i=@i+1;
END;
UNTIL @i>100
END REPEAT;
SELECT @num;
END//

CALL bb();

-- if,elseif,else 语句
--根据结果是否为空，分别处理
 IF ( ) THEN
```

```
BEGIN

END;

ELSEIF THEN

BEGIN

 END;

ELSE

BEGIN

 END;

 END IF;


-- 利用 SQL 条件语句，在 student 表中查找“李艳”同学的信息，若找到，则显示该生的学号、姓名、班级名称及班主任，否则显示“查无此人”。

DELIMITER//

CREATE PROCEDURE dd()

BEGIN

IF(SELECT sno FROM student WHERE sname='李艳')IS NOT NULL

THEN

BEGIN

SELECT sno,sname,classno FROM student WHERE sname = '李艳';

END;

SELECT '查无此人';

END IF;

END//
```

```
CALL dd();
```

```
-- 插入新用户
CREATE USER
'xmy'@'localhost' IDENTIFIED BY '123456789',
'mm'@'localhost' IDENTIFIED BY '987654321';

-- 插入表方法创建新用户
-- 使用 INSERT 语句创建一个新用户 student，主机名为 localhost，密码为
infomation。
INSERT INTO mysql.user
(HOST,USER,authentication_string,ssl_cipher,x509_issuer,x509_subject
)
VALUES ('localhost','student',MD5('infomation'),'','','');

-- 刷新
FLUSH PRIVILEGES;
```

```
-- 删除用户
-- drop 方法
DROP USER 'username1'@'hostname1','username2'@'hostname2';
-- delete 方法
DELETE FROM mysql.user
WHERE HOST='hostname' AND USER='username';


-- 修改用户名称
-- 将用户 king1 和 king2 的名字分别修改为 ken1 和 ken2
RENAME USER
'king1'@'localhost' TO 'ken1'@'localhost',
'king2'@'localhost' TO 'ken2'@'localhost';


-- 修改密码
-- 使用 mysqladmin 命令
mysqladmin -u username -h localhost -p
-- 使用 mysqladmin 命令将 root 用户的密码修改为"rootpwd
mysqladmin -u root -p PASSWORD "rootpwd";
-- 刷新
FLUSH PRIVILEGES;


-- 使用 update 语句修改 mysql 数据库中的 user 表
UPDATE  mysql.user
```

```
   SET  authentication_string=MD5('newpassword')
   WHERE  USER='username'AND HOST='localhost';

-- 使用 set password 语句修改密码
SET PASSWORD FOR 'username'@'localhost'='newpassword';


-- 权限管理
-- 授权
--使用 GRANT 语句为用户 ken1 赋权，使其对所有的数据有查询、插入权限，并授予
GRANT 权限
GRANT SELECT,INSERT on *.* TO 'ken1'@'localhost'
WITH GRANT OPTION;

--收回所有权限
REVOKE  ALL PRIVILEGES,GRANT OPTION
FROM 'username'@'hostname';

--收回指定权限
--收回 test1 用户对 gradem 数据库中 student 表的 cterm 列的 UPDATE 权限
REVOKE UPDATE(cterm) on gradem.sc
FROM 'test1'@'localhost';

-- 查看权限
SHOW GRANTS FOR 'username'@'localhost';
```

```
-- 备份数据库或表。mysqldump 备份数据库或表的基本语法格式如下
mysqldump –u user –h host –
ppassword  dbname[tbname,[tbname…]]>filename.sql;
--备份数据库 gradem(注意不需要;结尾)
mysqldump -u root -h root -p123456 gradem>gradem1.sql
--备份数据库 gradem 的 student 表和 sc 表(表与表之间使用空格隔开)
mysqldump -u root -h root -p123456 gradem student sc>gradem2.sql
-- --databases 参数，多数据库备份
--备份数据库 gradem 和数据库 mydb
mysqldump -u root -h root -p123456 --databases gradem
mydb>gradem3.sql

-- 数据恢复
mysql –u user –p [dbname]<filename.sql;
--将备份文件 gradem5 恢复到数据库 gradem 中
mysql –u root –p123456 gradem <d:\bak\gradem5.sql
-- 使用 source 命令（注意需要使用对应数据库后才能使用）
USE gradem;
SOURCE d:\bak\gradembak.sql

-- ----------------------------------------------------------------
----------------------------------------------
```

```sql
-- 1 查询所有学生的基本信息、所有课程的基本信息和所有学生的成绩信息（用三条SQL 语句）
SELECT * FROM student;
SELECT * FROM course;
SELECT * FROM score;

-- 2 查询所有学生的学号、姓名、性别和出生日期
SELECT 学号,姓名,性别, 出生日期 FROM student;

-- 3 查询所有课程的课程名称
SELECT DISTINCT 课程名称 FROM course;

-- 4 查询前 10 门课程的课号及课程名称
SELECT 课号,课程名称 FROM course LIMIT 10;

-- 5 查询所有学生的姓名及年龄
SELECT 姓名,YEAR(now())-YEAR(出生日期) AS 年龄 FROM student;

-- 6 查询所有年龄大于 18 岁的女生的学号和姓名
SELECT 姓名,学号 FROM student WHERE YEAR(NOW())-YEAR(出生日期)>18 AND 性别='女';

-- 7 查询所有男生的信息
SELECT * FROM student WHERE 性别='男';
```

```sql
-- 8 查询所有任课教师的姓名和所在系别
SELECT 教师姓名,所在系别 FROM teacher;

-- 9 查询“电子商务”专业的学生姓名、性别和出生日期
SELECT 姓名, 性别,出生日期 FROM student WHERE 专业='电子商务';

-- 10 查询 Student 表中的所有系名
SELECT DISTINCT 系名 FROM student;

-- 11 查询“C01”课程的开课学期
SELECT 开课学期 FROM teaching WHERE 序号='C01';

-- 12 查询成绩在 80～90 分之间的学生学号及课号
SELECT 学生学号,课号 FROM score WHERE 成绩 BETWEEN 80 AND 90;

-- 13 查询在 1970 年 1 月 1 日之前出生的男教师信息
SELECT * FROM teacher WHERE 性别='男' AND 出生日期<'1970-01-01';

-- 14 输出有成绩的学生学号
SELECT 学生学号 FROM score WHERE 成绩 IS NOT NULL;

-- 15 查询所有姓“刘”的学生信息
SELECT * FROM student WHERE 姓名 LIKE '%刘%';
```

```sql
-- 16 查询生源地不是山东省的学生信息
SELECT * FROM student WHERE 生源地 NOT LIKE '%山东省%';

-- 17 查询成绩为 79 分、89 分或 99 分的记录
SELECT * FROM score WHERE 成绩 = 79 OR 成绩= 89 OR 成绩= 99;

-- 18 查询名字中第二个字是“小”字的男生的学生姓名和地址
SELECT 姓名,地址 FROM student WHERE sname 姓名 '_小%' AND 性别 = '男';

-- 19 查询名称以“计算机”开头的课程名称
SELECT 课程名称 FROM course WHERE 课程名称 LIKE '^计算机';

-- 20 查询计算机工程系和软件工程系的学生信息
SELECT * FROM student WHERE 系别='计算机工程系' OR 系别='软件工程系';


-- 多表操作
-- 查询 goods 表中商品类别为服饰的商品 id，名称，价格，销售数量
SELECT gdid,gdName,gdPrice,gdSaleQty,tname FROM
goods JOIN goodstype
ON goods.tid=goodstype.tid
WHERE goodstype.tid = 1;

-- 查询段湘林购买订单的总价格
```

```sql
SELECT uname AS 购买人,SUM(oTotal) FROM
orders JOIN users
ON orders.uid = users.uid
WHERE uname = '段湘林';

-- 查询谁没有买过东西
SELECT * FROM users
WHERE uid NOT IN(SELECT DISTINCT uid FROM orders);

-- 查询与用户在同一城市的 uname 和 uphone
SELECT uName,uPhone,ucity FROM
users WHERE ucity =
(SELECT ucity FROM users WHERE uname = '蔡准');

SELECT A.uname,A.uphone,A.ucity FROM
 users A JOIN users B
ON A.ucity = B.ucity
WHERE B.uname = '蔡准';

-- 根据 taobao.sql 提供的数据，查询服饰类的商品卖出总数
SELECT SUM(gdSaleQty) AS 商品卖出总数 FROM
goods JOIN goodstype
ON goods.tid=goodstype.tid
WHERE goodstype.tname = '服饰';
```

```sql
-- 平均年龄
SELECT AVG(YEAR(CURDATE())-YEAR(sbirthday)) FROM student;

-- 年龄大于平均年龄的学生姓名
SELECT sname FROM student
WHERE YEAR(CURDATE())-YEAR(sbirthday)
> (SELECT AVG(YEAR(CURDATE())-YEAR(sbirthday)) FROM student);

-- 查询“LED 小台灯"的 2017 年全年销售量
SELECT SUM(odNum) AS 全年销售量 FROM
orderdetail JOIN goods
ON orderdetail.gdid = goods.gdid
WHERE goods.gdname = 'LED 小台灯';

-- 查询“LED 小台灯"的 2017 年全年销售量
SELECT SUM(odnum) AS 全年销售量 FROM
orders,orderdetail,goods
WHERE orders.oID= orderdetail.oID AND goods.gdID= orderdetail.gdID
AND gdname='LED 小台灯' AND YEAR(otime)=2017;

-- 查询购买过“LED 小台灯"的用户姓名、联系电话、电子邮箱
SELECT uname AS 用户姓名,uphone AS 联系电话,uemail AS 电子邮箱
FROM users,goods,orderdetail,orders
```

```
WHERE goods.gdID=orderdetail.gdID
AND orderdetail.oID=orders.oID
AND users.uID=orders.uID
AND gdname='LED 小台灯';

-- 查询用户段湘林的所有订单信息
SELECT * FROM orders
WHERE uid IN(
SELECT uid FROM users WHERE uname = '段湘林');

-- 查询 2017 年全年购买金额在 1000 元以上用户信息，显示用户名、性别和联系电话
SELECT uname AS 用户名,usex AS 性别,uphone AS 联系电话,SUM(ototal) AS 购买金额
FROM users,orders
WHERE users.uid = orders.uid
AND YEAR(otime) = 2017
GROUP BY orders.uID
HAVING SUM(ototal)>1000;

-- 按性别统计，2017 年全年男性和女性分别购买商品的订单总价
SELECT usex AS 性别,SUM(ototal) AS 商品订单总价
FROM users JOIN orders
```

```
ON users.uID=orders.uID
WHERE YEAR(otime)=2017 GROUP BY usex;

SELECT usex,SUM(ototal)
FROM users,orders
WHERE users.uID=orders.uID
AND YEAR(otime)=2017
GROUP BY usex;

-- 统计各种类别商品 2017 年全年卖出的总数，显示类别名称和数量，并按数量从高到低排序
SELECT tname AS 类别名称,SUM(odnum) AS 卖出总数量
FROM goods,goodstype,orders,orderdetail
WHERE goods.tID=goodstype.tID
AND orders.oID=orderdetail.oID
AND goods.gdID=orderdetail.gdID
AND YEAR(otime)=2017
GROUP BY goods.tID
ORDER BY SUM(odnum) DESC;

-- 多表操作 2
-- (1) 查询电子工程系女学生的学生学号、姓名及考试成绩。
SELECT a.sno AS 学号,sname AS 姓名,degree as 成绩 FROM
student a JOIN sc b ON a.sno = b.sno
```

```
WHERE  ssex = '女'
AND sdept = '电子工程系';


--
SELECT a.sno,sname,SUM(degree) FROM
student a JOIN sc b ON a.sno = b.sno
WHERE  ssex = '女'
AND sdept = '电子工程系'
GROUP BY sname;

--查询计算机工程系女生的学生学号、姓名及考试成绩:SELECT A.sno 学号,sname
姓名,degree 成绩
SELECT A.sno 学号,A.sname 姓名,B.degree 考试成绩
FROM student A,sc B
WHERE A.sno=B.sno AND ssex='女' AND sdept='计算机工程系';

-- (2) 查询“闫旭光”同学所选课程的成绩。(不考虑重名)
SELECT cname AS 姓名,degree AS 成绩 FROM
student a JOIN sc b ON a.sno = b.sno
  JOIN course c ON c.cno = b.cno
  WHERE sname = '闫旭光';

--查询“自己”所选课程的名称、成绩。
SELECT sname 姓名,cno 课程,degree 成绩
```

```
FROM student A,sc B
WHERE A.sno=B.sno AND sname='吴兵';

-- (3) 查询“李新”老师所授课程的课程名称。
SELECT cname AS 课程名称 FROM
course a JOIN teaching b ON a.cno = b.cno
    JOIN teacher c ON c.tno = b.tno
WHERE Tname = '李新';

-- 查询“李新”教师所授课程的课程名称。
SELECT tname 姓名,cname 课程
FROM teacher A,course B,teaching C
WHERE A.tno=C.tno AND B.cno=C.cno AND tname='李新';

-- (4) 查询女教师所授课程的课程号及课程名称。
SELECT a.cno AS 课程号,cname AS 课程名称 FROM
course a JOIN teaching b ON a.cno = b.cno
    JOIN teacher c ON c.tno = b.tno
WHERE Tsex = '女';

-- 查询女教师所授课程的课程号及课程名称:
SELECT tname 姓名,cname 课程,B.cno 课程号
FROM teacher A,course B,teaching C
WHERE A.tno=C.tno AND B.cno=C.cno AND tsex='女';
```

```sql
-- (5) 查询至少选修一门课程的女学生姓名。
SELECT sname AS 姓名 FROM
student a JOIN sc b ON a.sno = b.sno
WHERE ssex = '女'
GROUP BY a.sno
HAVING COUNT(cno)>1;

-- 查询至少选修了一门课程的女生姓名，
SELECT sname
FROM student A,sc B
WHERE A.sno=B.sno AND ssex='女'
GROUP BY A.sno
HAVING COUNT(cno)>=1;

-- (6) 查询姓“王”的学生所学的课程名称。
SELECT a.sname AS 姓名,cname AS 课程名称 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN course c ON c.cno = b.cno
WHERE sname LIKE '王%';

-- 查询姓“王”的学生所学的课称名称。
SELECT sname 姓名,cname 课程
FROM student A,course B,sc C
```

```
WHERE A.sno=C.sno AND B.cno=C.cno AND sname LIKE '%王%';


-- (7) 查询选修“高等数学”课程且成绩在 80～90 分之间的学生学号及成绩。
-- (8) 查询选修“高等数学”课程且成绩在 80～90 分之间的学生姓名及成绩。
SELECT a.sno AS 学号,degree AS 成绩 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN course c ON c.cno = b.cno
WHERE cname = '高等数学'
HAVING degree BETWEEN 80 AND 90;


-- 查询选修“数据库”课程且成绩在 80-90 分的学生学号及成绩
SELECT sno 学号,degree 成绩
FROM sc A,course B
WHERE  A.cno=B.cno AND B.cname LIKE '%数据库%'
HAVING degree BETWEEN 80 AND 90;


-- (9) 查询课程成绩及格的男同学的姓名及课程号与成绩。
SELECT sname AS 姓名,c.cno AS 课程号,degree AS 成绩 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN course c ON c.cno = b.cno
WHERE ssex = '男'
AND degree>60;


-- 查询课程成绩及格的男生的学生信息、课程号与成绩。
```

```sql
SELECT A.*,cno 课程号,degree 成绩
FROM student A,sc B
WHERE A.sno=B.sno AND degree>60 AND ssex='男';

-- (10) 查询选修“c04”课程的学生的平均年龄。
SELECT AVG(YEAR(NOW())-YEAR(sbirthday)) AS 平均年龄 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN course c ON c.cno = b.cno
WHERE c.cno = 'c04';

-- 查询选修“C04”课程的学生的平均年龄。
SELECT AVG(YEAR(CURDATE())-YEAR(sbirthday)) 平均年龄
FROM student a,sc b
WHERE a.sno=b.sno AND cno='c04';

-- (11) 查询学习课程名为“大学英语”的学生学号和姓名。
SELECT a.sno AS 学号,sname AS 姓名 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN course c ON c.cno = b.cno
WHERE cname = '大学英语';

-- 查询选修课程名为“数学”的学生学号和姓名。
SELECT A.sno 学号,sname 姓名
FROM student A,sc B,course C
```

```
WHERE A.sno=B.sno AND B.cno=C.cno AND cname LIKE '%数学';

-- (12) 查询“钱军”教师任课的课程号，选修其课程的学生的学号和成绩。
SELECT c.tno AS 课程号 ,a.sno AS 学号,degree AS 成绩 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN teaching c ON c.cno = b.cno
JOIN teacher d ON d.tno = c.tno
WHERE tname = '钱军';

-- 查询“钱军”教师任课的课程号、选修其课程的学生的学号和和成绩。
SELECT B.cno 课程号,cname 课程名,sno 学号,degree 成绩
FROM sc A,course B,teacher C,teaching D
WHERE A.cno=B.cno AND B.cno=D.cno AND C.tno=D.tno
AND tname='钱军';

-- (13)查询“钱军”教师任课的课程号，选修其课程的学生的姓名。
SELECT c.tno AS 课程号 ,sname AS 学生姓名 FROM
student a JOIN sc b ON a.sno = b.sno
JOIN teaching c ON c.cno = b.cno
JOIN teacher d ON d.tno = c.tno
WHERE tname = '钱军';

-- (14)查询在第 1 学期所开课程的课程名称及成绩。
SELECT cname AS 课程名称,degree AS 成绩 FROM
```

```
teaching a JOIN sc b ON  a.cno = b.cno
JOIN course c ON c.cno = b.cno
WHERE cterm ='1';

-- 查询在第 1 学期所开课程的课程名称及学生的成绩。
SELECT degree,cname
FROM teaching a,sc b,course c
WHERE a.cno=b.cno AND b.cno=c.cno
AND cterm='1'
GROUP BY sno;

-- (15)查询“c02”号课程不及格的学生信息。
SELECT * FROM
sc a JOIN student b ON a.sno=b.sno
WHERE cno='C02'
HAVING degree<60;

-- 查询“C02”课程不及格的学生信息。
SELECT *
FROM sc a,student b
WHERE a.sno=b.sno AND cno='C02'
HAVING degree<60;

-- (16)查询软件工程系成绩在 90 分以上的学生姓名、性别和课程名称。
```

```sql
SELECT sname,ssex,cname,deptname,degree FROM
department a JOIN class b ON a.deptno = b.deptno
JOIN student c ON c.classno = b.classno
JOIN sc d ON d.sno = c.sno
JOIN course e ON e.cno = d.cno
WHERE degree>90
 AND deptname = '软件工程系';


--
SELECT sname,ssex,cname FROM
student a JOIN sc b ON a.sno = b.sno
 JOIN course c ON c.cno = b.cno
 WHERE degree>90
 AND sdept = '软件工程系';

 -- 查询信息工程系成绩在 90 分以上的学生姓名、性别和课程名称。
SELECT sname 姓名,ssex 性别,cname 课程名称
FROM sc a,student b,course c
WHERE a.sno=b.sno AND a.cno=c.cno
AND sdept='信息工程系'
AND degree>90;

 -- (17) 查询同时选修了“c04”和“c02”课程的学生姓名。
SELECT sname AS 姓名 FROM
```

```
student a JOIN sc b ON a.sno=b.sno
JOIN sc c ON c.sno = b.sno
WHERE b.cno='C04' AND c.cno='C02'
GROUP BY sname;

-- 查询同时选修了“C04”和“C02”课程的学生姓名和成绩。
SELECT sname 姓名,b.degree 成绩,c.degree 成绩
FROM student a,sc b,sc c
WHERE a.sno=b.sno AND b.sno=c.sno
AND b.cno='C04' AND c.cno='C02'
GROUP BY sname;

-- 查询所有平均年龄大于平均年龄的学生姓名和年龄 gradem1
SELECT sname AS 姓名, YEAR(NOW()) - YEAR(sbirthday) AS 年龄 FROM
student
WHERE YEAR(NOW()) - YEAR(sbirthday) >
(SELECT AVG(YEAR(NOW())-YEAR(sbirthday)) FROM student);

-- 查询没有选修高等数学的学生学号和姓名 gradem1
SELECT sno,sname FROM student
WHERE sno NOT IN
(SELECT sno FROM sc WHERE cno IN(
SELECT cno FROM course WHERE cname = '高等数学'
));
```

```
-- 查询其他系中比计算机工程系某一学生年龄小的学生姓名和年龄 gradem
SELECT sname,YEAR(NOW())-YEAR(sbirthday) AS 年龄 FROM student
WHERE YEAR(NOW())-YEAR(sbirthday)< ANY(
SELECT YEAR(NOW())-YEAR(sbirthday) FROM student WHERE sdept = '计算机
工程系')
AND sdept <> '计算机工程系';

-- 子查询操作
-- 查询所有选修了 c01 号课程的学生姓名 gradem1
SELECT sname FROM student
WHERE EXISTS (
SELECT * FROM sc WHERE sno = student.sno
AND cno ='c01'
);

-- 查询选修了全部课程的学生姓名 gradem1
SELECT sname FROM student
WHERE NOT EXISTS(SELECT * FROM course WHERE NOT EXISTS(
SELECT * FROM sc WHERE sno=student.sno AND cno=course.cno)
);

-- 查询张忠同学所选课程的成绩 gradem
SELECT degree FROM sc
```

```
WHERE sno IN (
SELECT sno FROM student WHERE sname = '张忠'
);

-- 子查询“李新”老师所授课程的课程名称 gradem1
SELECT cname FROM course
WHERE cno IN (SELECT cno FROM teaching WHERE tno IN(SELECT tno FROM
teacher WHERE tname = '李新')
);

-- 子查询方法，查询女教师所授课程的课程号及课程名称
SELECT cno,cname FROM course
WHERE cno IN(SELECT cno FROM teaching
WHERE tno IN(SELECT tno FROM teacher WHERE tsex='女'));

-- （1）查询“李佳丽”同学所选课程的成绩
SELECT degree FROM sc
WHERE sno IN (
SELECT sno FROM student WHERE sname = '李佳丽'
);

-- （2）查询“李新”老师所授课程的课程名称。
SELECT cname FROM course
```

```
WHERE cno IN (SELECT cno FROM teaching WHERE tno IN(SELECT tno FROM
teacher WHERE tname = '李新')
);

-- （3）查询女教师所授课程的课程号及课程名称。
SELECT cno,cname FROM course
WHERE cno IN(SELECT cno FROM teaching
WHERE tno IN(SELECT tno FROM teacher WHERE tsex='女'));

-- （4）查询姓“王”的学生所学的课程名称。
SELECT cname FROM course
WHERE cno IN (SELECT cno FROM sc
WHERE sno IN(SELECT sno FROM student
WHERE sname LIKE '王%'));

-- （5）查询“C01”课程不及格的学生信息。
SELECT * FROM student
WHERE sno IN (SELECT sno FROM sc WHERE degree <60 AND cno='c01');

-- （6）查询选修“高等数学”课程且成绩在 80～90 分的学生学号及成绩。
SELECT sno,degree FROM sc
WHERE cno IN(SELECT cno FROM course
WHERE cname='高等数学') AND degree BETWEEN 80 AND 90;
```

```
-- （7）查询选修“C04”课程的学生的平均年龄。
SELECT AVG(YEAR(NOW())-YEAR(sbirthday)) AS 平均年龄 FROM
student WHERE sno IN(SELECT sno FROM sc WHERE cno = 'c04');

-- （8）查询选修课程名为“高等数学”的学生学号和姓名。
SELECT sno,sname FROM student
WHERE sno IN(SELECT sno FROM sc
WHERE cno IN(SELECT cno FROM course WHERE cname='高等数学'));

-- （9）查询“钱军”教师任课的课程号，选修其课程的学生的学号和成绩。
SELECT tno,sno,degree FROM sc,teacher
WHERE cno IN(SELECT cno FROM teaching
WHERE tno IN(SELECT tno FROM teacher WHERE tname='钱军'));

-- （10）查询在第 3 学期所开课程的课程名称及学生的成绩。
SELECT cname,degree FROM sc,course
WHERE sno IN(SELECT sno FROM teaching WHERE
cno IN(SELECT cno FROM course WHERE teaching.cterm = 3));

-- （11）查询与“李佳丽”同一个系的同学姓名。
SELECT sname FROM student
WHERE sdept IN(SELECT sdept FROM student WHERE sname='李佳丽');

-- （12）查询学号比“李佳丽”同学大，而出生日期比她小的学生姓名。
```

```
SELECT sname FROM student
WHERE YEAR(NOW())-YEAR(sbirthday)< ALL(SELECT YEAR(NOW())-
YEAR(sbirthday) FROM student
WHERE sname='李佳丽') AND sno>ALL (SELECT sno FROM student WHERE
sname='李佳丽');

-- （13）查询出生日期大于所有女同学出生日期的男同学的姓名及系别。
SELECT sname,sdept FROM student
WHERE sbirthday > ALL (SELECT sbirthday FROM student WHERE ssex='女
')
AND ssex='男';

-- （14）查询成绩比该课程平均成绩高的学生的学号及成绩。
SELECT sno,degree FROM sc a
WHERE degree > ALL (SELECT AVG(degree) FROM sc b WHERE a.sno =
b.sno);

-- （15）查询不讲授“C01”课的教师姓名。
SELECT tname FROM teacher
WHERE tno NOT IN (SELECT tno FROM teaching
WHERE cno='c01');

-- （16）查询没有选修“C02”课程的学生学号及姓名。
SELECT sno,sname FROM student
```

```
WHERE sno NOT IN (SELECT sno FROM sc WHERE cno='C02');


-- （17）查询选修了“高等数学”课程的学生学号、姓名及系别。
SELECT sno,sname,sdept FROM student
WHERE sno IN (SELECT sno FROM sc
WHERE cno IN(SELECT cno FROM course WHERE cname='高等数学'));


--
-- 把平均成绩大于 80 分的学生的学号和平均成绩存入另一个已知的基本表
s_grade(sno,avg_grade)中
INSERT INTO s_grade(sno,avg_grade)
SELECT sno,AVG(degree) FROM sc
GROUP BY sno
HAVING AVG(degree)>80;

-- 挑选出全班的女生信息，放在新表 girls
CREATE TABLE girls LIKE student;

INSERT INTO girls
SELECT * FROM student
WHERE ssex = '女';

-- 将电子工程系全体学生的成绩设置为 0
UPDATE sc SET degree=0
```

```
WHERE sno IN(SELECT sno FROM student WHERE sdept = '电子工程系');

-- 删除王小龙的信息
DELETE FROM student
WHERE sname = '王小龙';



-- （1）向 student 表中插入记录（"2005010203"，"张静"，"女"，"1981-3-
21"，"软件工程系"，"软件技术"）
INSERT INTO student(sno,sname,ssex,sbirthday,sdept,speciality) VALUE
("2005010203","张静","女","1981-3-21","软件工程系","软件技术");

-- （2）插入学号为“2005010302”、姓名为“李四”的学生信息
INSERT INTO student(sno,sname)
VALUE ('2005010302','李四');

-- （3）把电子工程系的学生记录保存到表 TS 中（TS 表已存在，表结构与 student
表相同）
CREATE TABLE ts LIKE student;

INSERT INTO ts
SELECT * FROM student
WHERE sdept = '电子工程系';
```

```
-- （4）将学号为“2005010202”的学生姓名改为“张华”，系别改为“电子工程系”，
专业改为“电子应用技术”。

UPDATE student SET sname='张华',sdept='电子工程系',speciality='电子应
用技术'
WHERE sno ='2005010202';

-- （5）将“李勇”同学的专业改为“计算机信息管理”。
UPDATE student SET speciality='计算机信息管理'
WHERE sname ='李勇';

-- （6）删除学号为“2005010302”的学生记录。
DELETE FROM student
WHERE sno ='2005010302';

-- （7）删除“计算机工程系”所有学生的选课记录。
DELETE FROM student
WHERE sno IN(SELECT sno FROM student WHERE sdept = '计算机工程系');

-- （8）删除 sc 表中尚无成绩的选课记录。
```

```
DELETE FROM sc
WHERE degree IS NULL;


-- （9）把“刘晨”同学的选修记录全部删除。
DELETE FROM student
WHERE sname = '刘晨';


-- 视图
-- 1.创建视图 v_goods，显示商品 ID，商品名称，商品价格，库存数量。并显示视图中数据。
CREATE VIEW v_goods
AS
SELECT gdID,gdName,gdPrice,gdQuantity FROM goods;

-- 2.创建视图 v_orders，显示订单编号、会员姓名、商品名称、总金额信息。并显示视图中数据。
CREATE OR REPLACE VIEW v_orders(订单编号,会员姓名,商品名称,总金额信息)
AS
SELECT c.oid,uName,gdName,oTotal FROM
users a,goods b,orders c,orderdetail d
WHERE a.uid = c.uid
AND c.oid = d.oid
AND b.gdid = d.gdid;
```

```
SELECT * FROM v_orders;

-- 3.使用 UPDATE 语句更新视图 v_goods，将所有商品单价提升 10%。并显示视图中数据。

UPDATE v_goods SET gdPrice = gdPrice*1.1;

SELECT * FROM v_goods;

-- 4.使用 DELETE 语句，更新视图 v_goods，删除 goods 表中最后一行。并显示视图中数据
DELETE FROM v_goods ORDER BY gdid DESC LIMIT 1;

SELECT * FROM v_goods;

-- 5.创建视图 v3，显示没有购买商品的编号，用户名、电话信息。并显示视图中数据
CREATE OR REPLACE VIEW v3(编号,用户名,电话信息)
AS
SELECT uid,uname,uphone FROM users
WHERE uid NOT IN(
SELECT uid FROM orders);
```

```sql
SELECT * FROM v3;


-- 6.删除以上创建的所有视图
DROP VIEW v_goods;
DROP VIEW v_orders;
DROP VIEW v3;


-- 储存过程
CREATE TABLE w(name1 CHAR(3),score INT);
INSERT w VALUE('xxx',82);

DELIMITER $$
CREATE PROCEDURE pw(IN mz CHAR(3),IN fen INT)
BEGIN
INSERT w VALUE(mz,fen);
SELECT * FROM w;
END $$

CALL pw('bbb',100);

-- rj347x 中获取名字相同人的信息
SELECT 学号,姓名,城市 FROM rj347x
WHERE 姓名 IN (
```

```
SELECT 姓名 FROM rj347x
GROUP BY 姓名
HAVING COUNT(*)>1
);

-- 查询李珍珍购买的商品
DELIMITER//
CREATE PROCEDURE ccc()
BEGIN
SELECT gdname FROM users a,scar b,goods c
WHERE uname = '李珍珍'
AND a.uid = b.uid
AND b.gdid =c.gdid;
END//

CALL ccc;

-- 创建一个从 SC 表中查询某一门课程考试成绩总分的存储过程 K2。
DELIMITER //
CREATE PROCEDURE k2(IN K2_cno CHAR(3))
BEGIN
SELECT SUM(degree) FROM sc WHERE cno = K2_cno;
END //
```

```
CALL k2('c01');
```

-- 创建存储过程 k3,从 student 表查询通过输入班级号参数传递给存储过程，查看结果全班同学信息

```
DELIMITER //
CREATE PROCEDURE k3(IN k3_classno CHAR(10))
BEGIN
SELECT * FROM student WHERE classno = k3_classno;
END //

CALL k3('20070101');
```

-- 创建一个从 student 表查询班级号为“20070301”班的学生资料的存储过程 proc_1，其中包括学号、姓名、性别、出生年月等。调用 proc_1 存储过程，观察执行结果。

```
DELIMITER //
CREATE PROCEDURE proc_1()
BEGIN
SELECT sno,sname,ssex,sbirthday FROM student WHERE classno =
'20070301';
END //

CALL proc_1();
```

```
-- 在 gradem 数据库中创建存储过程 proc_2，要求实现如下功能：存在不及格情况的学生选课情况列表，其中包括学号、姓名、性别、课程号、课程名、成绩、系别等。调用 proc_2 存储过程，观察执行结果。
DELIMITER //
CREATE PROCEDURE proc_2()
BEGIN
SELECT a.sno,sname,ssex,cno,degree,sdept FROM student a,sc b
WHERE degree<60
AND a.sno = b.sno;
END //

CALL proc_2();

-- 创建存储过程 spGetgoodsbygdID，根据商品 ID 查询指定的商品信息，显示 gdCode，gdName，gdPrice，gdCity。
DELIMITER//
CREATE PROCEDURE spGetgoodsbygdID(IN goods_id INT)
BEGIN
SELECT gdCode,gdName,gdPrice,gdCity FROM goods WHERE gdid =
goods_id;
END //

CALL spGetgoodsbygdID(4);
```

```
-- 创建存储过程 spGetuIDbyuName，根据用户名返回用户 ID（创建和调用带输入输出参数的存储过程）。
DELIMITER //
CREATE PROCEDURE spGetuIDbyuName(IN user_name VARCHAR(20),OUT user_id INT)
BEGIN
SELECT uid INTO user_id FROM users WHERE uName = user_name;
END//

CALL spGetuIDbyuName('段湘林',@uid);
SELECT @uid;

-- 为表 sc 创建一个插入触发器 student_sc_insert，当向表 sc 插入数据时，必须保证插入的学号有效地存在于 student 表中，
 -- 如果插入的学号在 student 表中不存在，插入新学号到 student 表。
 DELIMITER //
 CREATE TRIGGER student_sc_insert
 BEFORE INSERT ON sc FOR EACH ROW
 BEGIN
 IF(SELECT sno FROM student WHERE sno = new.sno) IS NULL
 THEN INSERT INTO student(sno) VALUES(new.sno);
 END IF;
 END//
```

```
 INSERT INTO sc VALUES('2007010199','a01',100);


-- 声明变量，并查询为软件工程系的学生信息
 SET @sdept = '软件工程系';
SELECT * FROM student WHERE sdept = @sdept;


-- 局部变量
DELIMITER//
CREATE PROCEDURE aa()
BEGIN
DECLARE myvar INT DEFAULT 10;
SET myvar =100;
SELECT myvar;
END//


CALL aa();


--将全局系统变量 sort_buffer_size 的值改为 40000，执行命令如下。
SET @@global.sort_buffer_size=40000;


--创建用户变量 user1 并赋值为 1，user2 赋值为 2，user3 赋值为 3。
--1
SET @user1=1, @user2=2,@user3=3;
```

```
--2
SELECT @user1:=1, @user2:=2,@user3:=3;

-- 利用存储过程或函数，求 1～100 的偶数和。
DELIMITER//
CREATE PROCEDURE cc()
BEGIN
SET @i:=2,@num:=0;
WHILE @i<=100 DO
BEGIN
SET @num=@num+@i;
SET @i=@i+2;
END;
END WHILE;
SELECT @num;
END//

CALL cc();

--
DELIMITER//
CREATE PROCEDURE ee()
BEGIN
SET @i=2,@num=0;
```

```
REPEAT
BEGIN
SET @num=@num+@i;
SET @i=@i+2;
END;
UNTIL @i>100
END REPEAT;
SELECT @num;
END//

CALL ee();


-- 创建 exam1 用户，初始密码设置为 123456。让该用户对所有数据库拥有
SELECT、CREATE、DROP、SUPER 和 GRANT 权限
CREATE USER 'exam1'@'localhost' IDENTIFIED BY '123456';
GRANT SELECT,CREATE,DROP,SUPER ON *.* TO 'exam1'@'localhost' WITH
GRANT OPTION;


-- 创建 exam2 用户，该用户没有初始密码
CREATE USER 'exam2'@'localhost';

-- 查看 exam1 用户和 exam2 用户的权限信息
SHOW GRANTS FOR 'exam1'@'localhost';
```

```
SHOW GRANTS FOR 'exam2'@'localhost';


-- 用 root 用户登录，收回 exam1 用户和 exam2 用户的所有权限
REVOKE ALL PRIVILEGES,GRANT OPTION FROM
'exam1'@'localhost','exam2'@'localhost';


--
REVOKE ALL ON *.* FROM 'exam1'@'localhost','exam2'@'localhost';


-- 删除 exam1 用户和 exam2 用户
DROP USER 'exam1'@'localhost','exam2'@'localhost';


-- 使用 GRANT 语句将 gradem 数据库中 sc 表的 degree 列和 cterm 列的 UPDATE 权
限授予用户 test1
GRANT UPDATE(degree,cterm) on gradem.sc
to 'test1'@'localhost';


-- 使用 GRANT 语句将 gradem 数据库中 student 表的 DELETE 权限授予用户 ken1。
GRANT DELETE on gradem.student
TO 'ken1'@'localhost';


-- 使用 REVOKE 语句收回 ken1 用户的所有权限，包括 GRANT 权限。
REVOKE ALL PRIVILEGES,GRANT OPTION
```

```
FROM 'ken1'@'localhost';


-- 用户管理
-- 创建数据库用户“user1”“user2”,密码为“123”。
CREATE USER 'user1'@'localhost' IDENTIFIED BY '123',
'user2'@'localhost' IDENTIFIED BY '123';

CREATE USER 'user2'@'localhost' IDENTIFIED BY '123';
-- 将用户“user2”的名称改为“user3”。
RENAME USER 'user2'@'localhost' TO 'user3'@'localhost';


-- 将用户“user3”的密码改为“123456”。
SET PASSWORD FOR 'user3'@'localhost' = '123456';


-- 删除用户“user3”。
DROP USER 'user3'@'localhost';

--
DROP USER 'user1'@'localhost';


-- 权限管理
```

```
-- 授子用户“user1”对 LibraryDB 数据库中读者表的“SELECT”操作权限；
GRANT SELECT ON librarydb.读者表 TO 'user1'@'localhost';

SHOW GRANTS FOR 'user1'@'localhost';

-- 授予用户“user1”对 LibraryDB 数据库中借阅表的插入、修改、删除操作权限；
GRANT INSERT,UPDATE,DELETE ON librarydb.借阅表 TO
'user1'@'localhost';

-- 授予用户“user1”对 LibraryDB 数据库拥有所有操作权限；
GRANT ALL PRIVILEGES ON librarydb.* TO 'user1'@'localhost';

-- 授予用户“user2”对 LibraryDB 数据库中的库存表有“SELECT”操作权限，并许其
将该权限授予其他用户；
GRANT SELECT ON librarydb.库存表 TO 'user2'@'localhost' WITH GRANT
OPTION;

SHOW GRANTS FOR 'user2'@'localhost';

-- 收回用户“user1”对 LibraryDB 数据库中读者表的“SELECT“”操作权限；
REVOKE SELECT ON librarydb.读者表 FROM 'user1'@'localhost';
```

1.在数据库中，下列说法不正确的是(D)。

A.数据库避免了一切数据的重复

B.若系统是完全可以控制的，则系统可确保更新时的一致性

C.数据库中的数据可以共享

D.数据库减少了数据冗余

2.如何构造出一个合适的数据逻辑结构是(C)主要解决的问题。

A.关系系统查询优化

B.数据字典

C.关系数据库规范化理论

D.关系数据库查询

3.下述选项中,(D)不是数据操纵语句。

A.插入 Insert

B.更新 Update

C.删除 Delete

D.创造 Create

4.CREATEUNIQUEINDEXwriter_indexON 作者信息(作者编号)语句创建了一个(A)索引

A.唯一索引 UNIQUE

B.全文索引

C.普通索引 INDEX

D.空间索引

5.SQL 使用(C)语句为用户授予系统权限或对象权限。

A.查询 SELECT

B.创造 CREATE

C.授予 GRANT

D.移除 REVOKE

6.使用 SELECT 将表中数据导出到文件，可以使用哪一子句(B)？

A.TOFILE

B.INTOFILE

C.OUTTOFILE

D.INTOOUTFILE

7.关系运算中花费时间最长的运算时(B)

A.选择

B.笛卡尔积

C.投影

D.除

8.关系数据库的规范化理论主要解决的问题是(A)。

A.如何构造合适的数据逻辑结构

B.如何构造合适的数据物理结构

C.如何构造合适的应用程序界面

D.如何控制不同用户的数据操作权限

9.关系数据库中，主键是(D)

A.创建唯一的索引，允许空值

B.只允许以表中第一字段建立

C.允许有多个主键的

D.为标识表中唯一的实体

10.下述选项中,(D)是数据定义语句。

A.插入 Insert

B.更新 Update

C.删除 Delete

D.创造 Create

11.在 SQL 中,谓词“EXIST”的含义是(A)。

A.存在量词

B.自然连接

C.全称量词

D.等值连接

12.LEFTJOIN 用于(B)。

A.显示符合条件的数据行

B.显示符合条件的数据行以及左边表中不符合条件的数据行

C.显示符合条件的数据行以及右边表中不符合条件的数据行

D.将两个表中的记录匹配成新的数据行

13.如事务 T 对数据对象 R 实行 X 封锁,则 T 对 R(C)。

A.只能读不能写

B.只能写不能读

C.即可读又可写

D.不能读也不能写

14.以下选项中,不属于关系性质的是(D)。

A.关系的列必须是同质的

B.关系中的元组的顺序可以改变

C.关系中列的顺序可以改变

D.关系中不同字段的域不能相同

15.DB、DBMS 和 DBS 三者之间的关系是(B)。

A.DB 包括 DBMS 和 DBS

B.DBS 包括 DB 和 DBMS

C.DBMS 包括 DB 和 DBS

D.不能相互包括

16.SQL 语言中,下列涉及空值的操作,不正确的是(C)

A.AGEISNULL

B.AGEISNOTNULL

C.AGE=NULL

D.NOT(AGEISNULL)

17.实现事务回滚的语句是(C)。

A.授予 GRANT

B.提交 COMMIT

C.回滚 ROLLBACK

D.移除 REVOKE

18.关系的模式和子模式分别对应 SQL 中的(C)。

A.基本表和外模式

B.基本表和表的文件

C.基本表和视图

D.视图和基本表

19.在数据库的三级模式结构中,描述数据库全局逻辑结构和特性的是(D)。

A.外模式

B.内模式

C.存储模式

D.模式

20.(A)可以看成是现实世界到机器世界的一个过渡的中间层次。

A.概念模型

B.逻辑模型

C.结构模型

D.物理模型

21.公司中有多个部门和多名职员，每个职员只能属于一个部门，

一个部门可以有多名职员，从职员到部门的联系类型是(C)

A.多对多

B.一对一

C.多对一

D.一对多

22.有关分布式结构的数据库系统说法正确的是(A)。

A.B/S 模式是一种特殊的分布式数据库系统体系结构

B.分布式结构中数据的分布式存储对于终端用户是可见的

C.C/S 模式的服务器端不是分布式数据库系统体系结构

D.分布式结构的缺点是终端数量较多,主机任务过重

23.主键的建立有(D)种方法

A.一

B.四

C.二

D.三

24.关系代数中的 π 运算符对应 SELECT 语句中的以下哪个子句？(A)

A.SELECT

B.FROM

C.WHERE

D.GROUPBY

25.数据库(DB)、数据库系统(DBS)、数据库管理系统(DBMS)之间的关系是(C)。

A.DB 包含 DBS 和 DBMS

B.DBMS 包含 DB 和 DBS

C.DBS 包含 DB 和 DBMS

D.没有任何关系

26.数据库系统的核心是(B)。

A.数据模型

B.数据库管理系统

C.数据库

D.数据操作

27.解决并发控制带来的数据不一致问题普遍采用的技术是(A)。

A.封锁

B.存取控制

C.恢复

D.协商

28.如何构造出一个合适的数据逻辑结构是(C)主要解决的问题。

A.关系系统查询优化

B.数据字典

C.关系数据库规范化理论

D.关系数据库查询

29.触发器可以创建在(A)中。

A.表

B.过程

C.数据过程

D.函数

30.将 E-R 模型转换成关系模型，属于数据库的(C)。

A.需求分析

B.概念设计

C.逻辑设计

D.物理设计

31.在关系模式 SCX(SNO,SN,TCO,CNO,SCORE)中,主键为 SNO 和 CNO,
则当学生 A 没有选取任何课程是,该关系模式存在的问题是(D)。

A.数据冗余

B.删除异常

C.更新异常

D.插入异常

32.SQL 中,ORDERBY 子句（排序）的位置是(C)。

A.SELECT 子句之后

B.WHERE 子句之后

C.最后一行

D.任意一行

34.同一个关系模型的任意两个元组值(A)。

A.不能完全相同

B.可以完全相同

C.必须完全相同

D.以上都不是

35.创建数据库使用以下哪项(D)

A.CREATE mytest

B.CREATE tablemytest

C.DATABASE mytest

D.CREATE DATABASE mytest

36.若用如下的 SQL 语句创建了一个表

```
CREATE TABLESC(S# CHAR(6) NOTNULL, C# CHAR(3) NOTNULL, SCORE
INTEGER, NOTE CHAR(20));
```

向 SC 表插入如下行时，(B)行可以被插入。

A.(NULL, '103', 80, '选修')

B.('200823', '101', NULL, NULL)

C.('201132', NULL, 86, '')

D.(’201009’, ’111’, 60, 必修)

37.下列关于关系的理解,错误是的(C)。

A.在一个关系中,可以任意交换两个元组的顺序

B.在一个关系中,可以任意交换两个列的顺序

C.每个列必须有自己的名字且必须取自不同的域

D.关系中的每一个份量必须是一个确定的值,即属性不可再分。

38.对于分布式数据库,可以简单归纳为(B)。

A.数据逻辑上分散,物理上统一

B.数据物理上分散,逻辑上统一

C.数据在逻辑上、物理上都是分散的

D.数据在逻辑上、物理上都是统一的

39.SQL 语言中，删除一个视图的命令是(D)

A.REMOVE

B.CLEAR

C.DELETE

D.DROP

40.实现关系代数投影运算的 SQL 子句是(A)

A.SELECT

B.ORDERBY

C.FROM

D.WHERE

41.创建索引是为了(A)。

A.提高存取速度

B.减少 I/O

C.节约空间

D.减少缓冲区个数

42.设有关系 SC(sno,cname,grade)，

各属性的含义分别为学号、课程名成绩。若要将所有学生的“大学计算机基础”课程的成绩增加 3 分，能正确完成该操作的 SQL 语句是(B)grade=grade+3WHEREcname='大学计算机基础'。

A.Update

B.UPDATE SC set

C.UPDATE set

D.UPDATEd SC set

43.SQL 语言一次查询的结果是一个(D)

A.数据项

B.记录

C.元组

D.表

44.在视图上不能完成的操作是(B)

A.更新视图数据

B.在视图上定义新的基本表

C.在视图上定义新的视图

D.查询

45.有两个关系 R 和 S,R 表示为[X,Y],S 表示为[Y,Z],如果 R 中有 n 条记录,S 中有 m 条记录,则 R 和 S 的除法所得的关系中记录的数量不可能为(B)。

A.m

B.m+n

C.0

D.m-n

46.创建视图的命令是(D)

A.ALTER view

B.ALTER table

C.CREATE table

D.CREATE view

47.下述关于数据库的错误叙述是(B)。

A.文件系统的数据管理方式解决了数据的物理独立性问题

B.文件系统的数据管理方式解决了数据共享难,数据冗余问题

C.数据库系统的数据管理方式解决了数据逻辑独立性问题

D.数据库系统的数据管理方式解决了数据完整性控制问题

48.对数据库物理存储方式的描述称为(B)

A.外模式

B.内模式

C.概念模式

D.逻辑模式

49.当事务 T 更新了数据 R 为 Q,事务 P 读取 Q,

之后事务 T 因为某些原因撤销了对事务 R 的更新,则上述过程产生的问题为(B)。

A.丢失更新

B.污读

C.不可重读

D.数据冗余

50.使用 CREATE TABLE 语句的(A)子句，在创建基本表时可以启用全文本搜索

A.FULLTEXT

B.ENGINE

C.FROM

D.WHRER

51.在数据库三级模式间引入二级映象的主要作用是(A)

A.提高数据与程序的独立性

B.提高数据与程序的安全性

C.保持数据与程序的一致性

D.提高数据与程序的可移植性

52.(C)备份是在某一次完全备份的基础上，只备份其后数据的变化。

A.比较

B.检查

C.增量

D.二次

53.关于用户模式、概念模式和内模式的数量说法正确的是(C)。

A.多个用户模式和概念模式,一个内模式

B.用户模式、内模式和概念模式的数量均为一个

C.多个用户模式,一个内模式和概念模式

D.用户模式、内模式和概念模式的数量可以为多个

54.以下选项中,(C)不是关系模型的组成部分。

A.完整性约束

B.数据结构

C.数据恢复

D.数据操作

55.事务的原子性是指(B)。

A.事务一旦提交,对数据库的改变是永久的

B.事务中包括的所有操作要么都做,要么都不做

C.一个事务内部的操作及使用的数据对并发的其他事务是隔离的

D.事务必须使数据从一个一致性状态到另一个一致性状态

56.2NF 规范到 3NF 是为了消除(C)。

A.非主属性对候选键的部分函数依赖

B.主属性对候选键的部分函数依赖

C.非主属性对候选键的传递函数依赖

D.以上都不是

57.设关系 R 和 S 的属性个数分别为 r 和 s,则(R×S)操作结果的属性个数为(A)。

A.r+s

B.r-s

C.r×s

D.max(r,s)

58.视图是一个“虚表”,视图的构造基于(C)

A.基本表

B.视图

C.基本表或视图

D.数据字典

60.下列哪些语句对主键的说明正确(C)

A.主键可重复

B.主键不唯一

C.在数据表中的唯一索引

D.主键用 foreign key 修饰

61.在关系模式 R 中函数依赖 A→B 的语义是(B)。

A.在 R 的某一个关系中,若两个元组的 X 值相等,则 Y 值也相等

B.在 R 的每一个关系中,若两个元组的 X 值相等,则 Y 值也相等

C.在 R 的某一关系中,Y 值应与 X 值相等

D.在 R 的每一关系中,Y 值应与 X 值相等

62.实现关系代数选取运算的 SQL 子句是(D)。

A.SELECT

B.ORDERBY

C.FROM

D.WHERE

63.SQL 语言是(C)的缩写。

A.结构化定义语言

B.结构化控制语言

C.结构化查询语言

D.结构化操纵语言

64.日志文件是用于记录(C)。

A.程序运行过程

B.数据操作

C.对数据的所有更新操作

D.程序执行结果

65.在下列关系代数的操作中,(C)不属于专门的关系运算。

A.自然连接

B.投影

C.广义笛卡尔积

D.连接

66.以下关于相关子查询,以下说法正确的是(C)。

A.先执行子查询

B.子查询的查询条件与父查询中数据表无关

C.父查询和子查询交替执行

D.子查询执行一次

67.一张表的主键个数为(C)

A.至多 3 个

B.没有限制

C.至多 1 个

D.至多 2 个

68.一个关系只有一个(C)。

A.候选码

B.外码

C.主码

D.以上都不是

69.能够消除多值依赖引起冗余的是(C)。

A.2NF

B.3NF

C.4NF

D.BCNF

70.关系 R 与关系 S 只有 1 个公共属性,T1 是 R 与 S 等值连接的结果,T2 是 R 与 S 自然连接的结果,则(D)。

A.T1 的属性个数等于 T2 的属性个数

B.T1 的属性个数小于 T2 的属性个数

C.T1 的属性个数大于或等于 T2 的属性个数

D.T1 的属性个数大于 T2 的属性个数

71.数据的存储结构与数据逻辑结构之间的独立性称为数据的(B)。

A.结构独立性

B.物理独立性

C.逻辑独立性

D.分布独立性

72.下列不属于数据库系统的三级模式结构的是(B)。

A.外模式

B.抽象模式

C.模式

D.内模式

73.通过哪些关系运算,可以为关系增加一条记录和删除一条记录(D)。

A.并和交

B.并和笛卡尔积

C.除法和差

D.并和差

74.建议采取的数据库行为设计和结构设计的方法分别是(C)。

A.均是自顶向下

B.自底向上和自顶向下

C.自顶向下和自底向上

D.均是自底向上

75.为了保证数据的物理独立性,需要修改的是(B)。

A.模式

B.模式与内模式之间的映射

C.外模式

D.模式与外模式之间的映射

76.从 E-R 模型向关系模型转换的时候,一个 M:N 联系转为关系模式时,该关系模式的关键字是(C)。

A.M 端实体的关键字

B.N 端实体关键字

C.M 端实体关键字和 N 端实体关键字的组合

D.其它

81.(B)属于信息世界的模型,是现实世界到计算机世界的一个中间层次。

A.数据模型

B.概念模型

C.关系模型

D.层次模型

---

1. 需在存储过程中使用某一参数向存储过程传入数值,

同时该参数还存储了需要返回的计算结果,

则该参数在存储过程参数列表中的修饰符为(INOUT)

2 分布 E-R 图之间的冲突主要有(属性冲突)、(命名冲突)、(结构冲突)三种

3. 数据库的三级模式结构中,外模式/模式映像可以保证数据和应用程序间的(逻辑独立性),

模式/内模式映像可以保证数据和应用程序间的(物理独立性)

4.当创建主键的时候,数据库管理系统会自动生成(聚集型索引)

5.在一个超市销售系统中,希望每次插入数据后,都自动对当天有营业额的统计工作,在数据库中可以(触发器)通过实现该需求

6.使用 SELECT 语句查询时,要去掉查询结果中的重复记录,应该使用(DISTINCT)关键字

7.在关系模式的分解中,数据等价用(无损连接)衡量,函数依赖等价用(保持函数依赖)衡量

8.在数据库实施阶段包括两项重要工作,一项是数据的(载入),另一项是(应用程序的编码和调试)

9.局部 E-R 图合并时的冲突有(属性冲突)、(命名冲突)、(结构冲突)三种

11.关系模式 R(X,Y,Z)中,函数依赖发生在 XY 和 YZ 上,则分解{XZ,XY}的主要问题是(丢失函数依赖)

12.DFD 是数据库设计过程中(需求分析)阶段的工作。

在概念设计向逻辑设计转换时,(1 对 1)联系和(1 对多)联系通常不用转换成对应的基本表

13.(触发器)是特殊类型的存储过程，它能在任何试图改变表中由触发器保护的数据时执行

14.在数据库系统中,更换了存储数据文件的硬盘,系统的业务逻辑无需改变,这一特性被称为(物理独立性)

15.E—R 图的主要元素是(实体型)、(属性)、(联系)

16.若关系为 1NF,且它的每一个非主属性都不部分依赖于候选码,则成该关系为(2NF)

17.当将局部 E-R 图集成为全局 E-R 图时，如果同一对象在一个局部 E-R 图中作为实体，而在另一个局部 E-R 图中作为属性，则这种现象称(结构冲突)

18.在数据库管理系统中,如果规定脚本中出现的操作都是针对 ADO 数据库进行的,则在脚本的最顶部应该声明的语句是(use ADO)(即使用数据库)

19.按照转储状态,数据转储可以分为(冷备份)和(热备份)

20.一个关系模式的形式化(五元组)表示为(R(U,D,DOM,F))

21.关系规则的完整性规则包括(参照完整性)、(实体完整性)和(用户自定义)的完整性

22.关系系统的完整性控制包括(实体完整性)、(参照完整性)和(用户自定义)的完整性

23.层次模型无法表达的实体联系类型为(多对多)关系

24.从关系规范化理论的角度讲，一个只满足 1NF 的关系可能存在的四方面问题是：数据(冗余度大、插入异常、修改异常、删除异常)。

25.如果两个实体之间具有 m：n 联系，则将它们转换为关系模型的结果是(3)个表

26.事务的特征包括:(原子性、一致性、隔离性、持久性)

27.保护数据库,防止未经授权的或不合法的使用造成的数据泄露、更改破坏。这是指数据的(安全性)

28.数据库系统一般包括(数据库管理系统、应用系统、数据库管理员、用户)

29.关系代数运算中,专门的关系运算有(选择、投影、连接)

30.设有关系模式 R(U,V,X,Y),函数依赖为{UV→XY,U→Y},则 R 的候选键是 UV,它属于(1NF)范式的关系模式

31.子查询依赖于父类查询,这类查询称为(相关子查询)

32.如果采用关系数据库来实现应用，则在数据库的逻辑设计阶段需将(E-R 图)转换为关系模式

33.删除数据库的关键字是:(DROP)

34.数据库发展经历了三个阶段:(人工管理阶段、文件系统阶段、数据库系统阶段)

35.逻辑层次上的数据模型有三个要素:(数据结构、数据完整性、数据操纵)

36.视图是虚表,只在数据库中存储其(定义)

37.(BC)范式消除了主属性对主码的部分函数依赖

38.对于非规范化的关系模式,经过(所有属性不可再分)转换为 1NF,
将 1NF 经过(非主属性完全依赖于主关系键)转换为 2NF,
将 2NF 经过(非主属性不传递依赖于主关系键)转换为 3NF

39.数据库的并发操作导致的数据库不一致性包括:(丢失更新、污读、不可重读)

41.关系中属性不可再分,也称为(域关系演算)

42.数据库的安全保护功能包括:(安全性、完整性控制、并发性控制、故障恢复)等多方面

43.在 MyqlL 中，触发器的执行时间有两种：(after)和(before)

44.使用 Create View 语句产生的虚表称为(视图)

45.如果一个满足 1NF 关系的所有属性合起来组成一个关键字，

则该关系最高满足的范式是(3NF)[注:在 1NF、2NF、3NF 范围内]

46.SQL 语言提供(数据定义、数据查询、数据操纵、数据控制)等功能

47.装入测试数据并进行数据库调试工作是在(数据库实施)阶段的工作

49.数据库的备份类型按涉及的数据范围来划分有 3 种，分别是(完整备份、增量备份、差异备份)

50.判断函数依赖集 F 和 G 等价的充分必要条件是(F+=G+)

51.数据库设计的一般步骤有：

(需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库的实施、运行与维护)

52.在视图中删除或修改一条记录，相应的(基本表)也随着视图更新

53.索引可以分为(聚集型索引)和(主键索引)

54.并发控制的主要方法是封锁,基本的封锁类型有(互斥锁)和(共享锁)

55.数据独立性分为(逻辑数据独立性)和(物理数据独立性)

56.当数据的物理存储改变了,应用程序不变,而由 DBMS 处理这种改变,这是指数据的(物理独立性)

57.关系模型中三类完整性约束有(实体完整性、参照完整性、用户自定义的完整性)

58.视图建立完成后,数据字典中存放的是视图的(定义)

59.设有学生表 S[学号，姓名，班级]和学生选课表 SC[学号，课程号，成绩]，为维护数据一致性，表 S 与 SC 之间应满足(参照完整性约束)

60.在 MySQL 中，有两种基本类型的索引:(普通索引)和(唯一索引)

61.关系代数中专门的关系运算包括：(选择 where、投影 select、连接 from、除)

62.SQL 支持数据库的三级模式结构,其中
(外模式)对应于视图和部分基本表,
(模式)对应于基本表,
(内模式)对应于存储文件