技能二：数据库（MySQL）

考试分值约占 20%（16 分）

教程说明：

红色字体：核心高频（必需熟记）

绿色字体：重点内容（必需熟悉）

黄色字体：难点内容（尽量掌握）

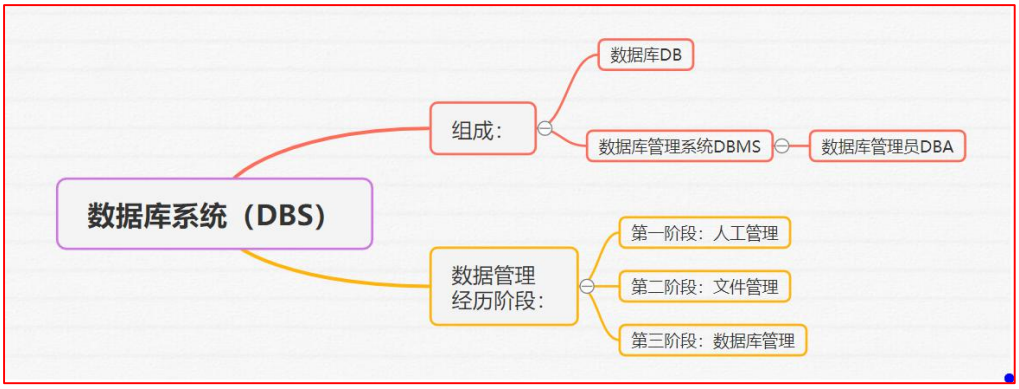
黑色字体：常规内容（必需了解）

蓝等字体：代码关键字等相关内容

1．数据库概述

1.1 根据系统需求绘制 E-R 图

1、数据库系统



2、**表**

（1）表是一种有结构化的文件。

（2）表（table）：某种特定类型数据的结构化清单。

（3）在数据库中可以创建多个文件名，且文件名不能重复，是唯一的标识。

（4）模式（schema）： 关于数据库和表的布局及特性的信息

（5）关系可以表示为一个二维的表。

（6）表有记录（元组）、列（属性、码、字段）等

3、**列**

（1）**表由列组成。列中存储着表中某部分的信息。**

（2）**列（column）是表中的一个字段。**

（3）**所有表都是由一个或多个列组成的。**

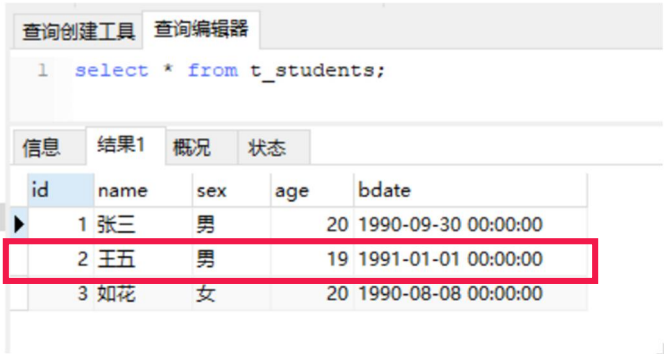
（4）**可以使用 SELECT（投影）显示列。**

4、**行**

（1）行（row） 表中的一个记录

（2）一个表中可以有 0 或多条记录。数据量越大，查询的速度就会慢，所以可以对索引的使用、使用合适的数据类型来等提升或优化查询速度。

（3）可以通过 WHERE（选择）语句将记录筛选出来。



5、主键

（1）表中每一行都应该有可以唯一标识自己的一列（或一组列）

（2）一个表只能有一个主键。主键（主码、主关键字等）可以是一列或多列组成。

（3）主键使用的 SQL 中 primary key（大小写不区分）来定义标识。

（4）作用：主键实现唯一标识实体,主键实现实体完整性。

（5）比如学号、教师编号、区号、产品编号等

6、外键

（1）某表的一个主键,出现在另一个表中,那么这个主键在另一个表中的定义为外键（外码）。

（2）作用：保持数据一致性，完整性，即数据参照完整性。

（3）外键使用的 SQL 中 foreign key（大小写不区分）来定义标识。

（4）一个表中可以有多个外键。

（5）比如成绩表中：教师编号、学号、课程号等者是成绩表的外键。

7、数据类型

（1）数据库中每个列都有相应的数据类型。

（2）数据类型定义列可以存储的数据种类。

（3）例如，如果列中存储的为数字（或许是订单中的物品数），则相应的数据类型应该为数值类型。如果列中存储的是日期、文本、注释、金额等，则应该用恰当的数据类型规定出来。

（4）数据类型（datatype） 所容许的数据的类型。

（5）每个表列都有相应的数据类型，它限制（或容许）该列中存储的数据。

8、**数据库设计阶段**

9、**E-R 图**

（1）**E-R 图也称实体-联系图(Entity Relationship Diagram)，提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。**

（2）**联系的类型**：

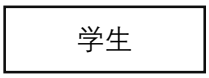
①、**1:1（一对一）**：比如身份证信息与人之间关系，一个人只能有一个身份证信息。

②、**1:n（一对多）**：比如宿舍与学生之间关系；一个宿舍可以住多个学生居住。

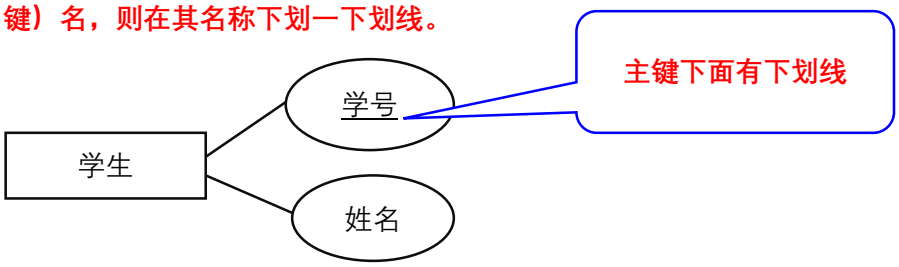
③、**m:n（多对多）**：比如学生与课程之间关系；一个学生可以选择多门课，一门课可以被多个学生选择

（3）**图形表示**：

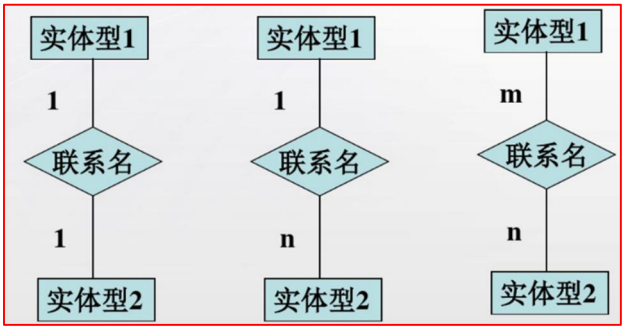
①、**矩形框**：**表示实体型，矩形框内写明实体名称**；



②、**椭圆图框：表示实体的属性，并用“实心线段”将其与相应关系的“实体型”连接起来，对于主属性（主键）名，则在其名称下划一下划线。**



③、**菱形框：表示实体型之间的联系成因，在菱形框内写明联系名，并用”实心线段“分别与有关实体型连接起来，同时在”实心线段“旁标上联系的类型**



（4）画 E-R 图的常用软件或工具：**visio、rose、word(插入形状)**等。

10、绘制 E-R 步骤

（1）首先要找出实体，以及实体的相应属于

（2）其次找到实体间存在的关系

（3）最后按照 E-R 图模型规则来绘制

11、一对一 E-R 图举例

例 1：两个实体集之间的一对一的联系的绘制方法。假设某学院有若干个系，每个系只有一名主任。则主任和系的属性分别如下：

主任—编号，姓名，年龄，学历

系—系编号，系名

主任和系之间是一个管理关系，体现主任一个任职时间。如一对一 E-R 图的示。

12、一对多 E-R 图举例

例 2：假设在某仓库管理系统中，有两个实体集：他库和商品。仓库用来存放商品，且规定一类商品只能存放在一个仓库中，一个仓库可以存放多件商品。

仓库和商品之间是一对多的联系。

仓库—仓库号，地点，面积联单

商品—商品号，商品名，价格

在存放联系中要反映出存放商品的数量。如一对多 E-R 图的示。

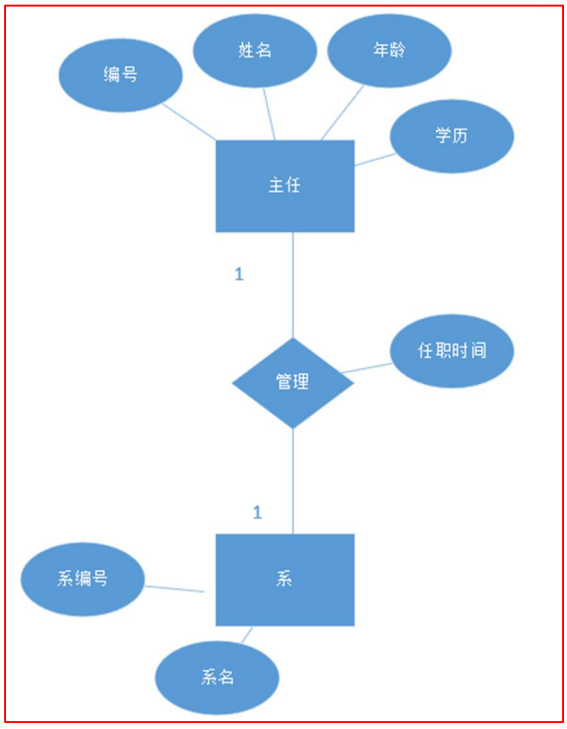
13、多对多 E-R 图举例

例 3：假设在某教务管理系统中，一个教师可以上多门课，一门课也可以由多个老师去上。教师和课程之间是多对多的联系。教师和课程可用以下属性来描述：

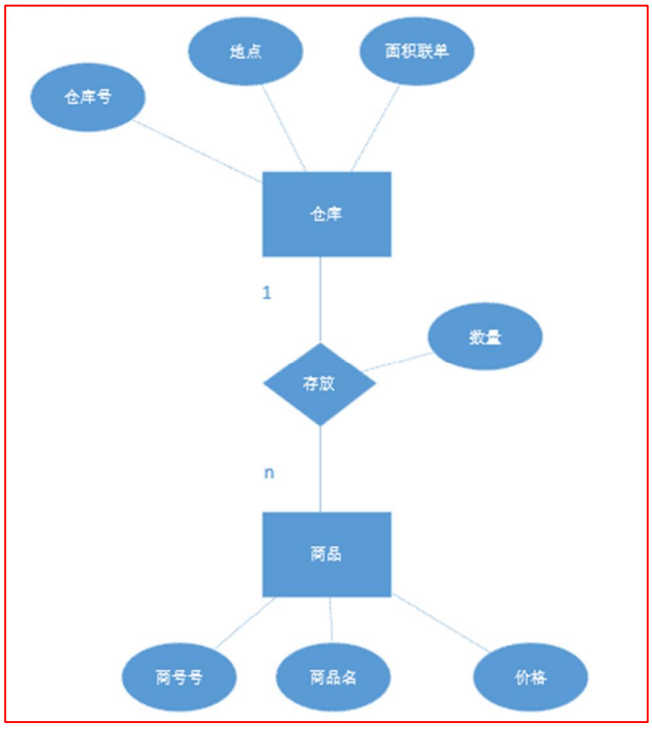
教师—教师号，教师名，职称

课程—课程号，课程名，班级

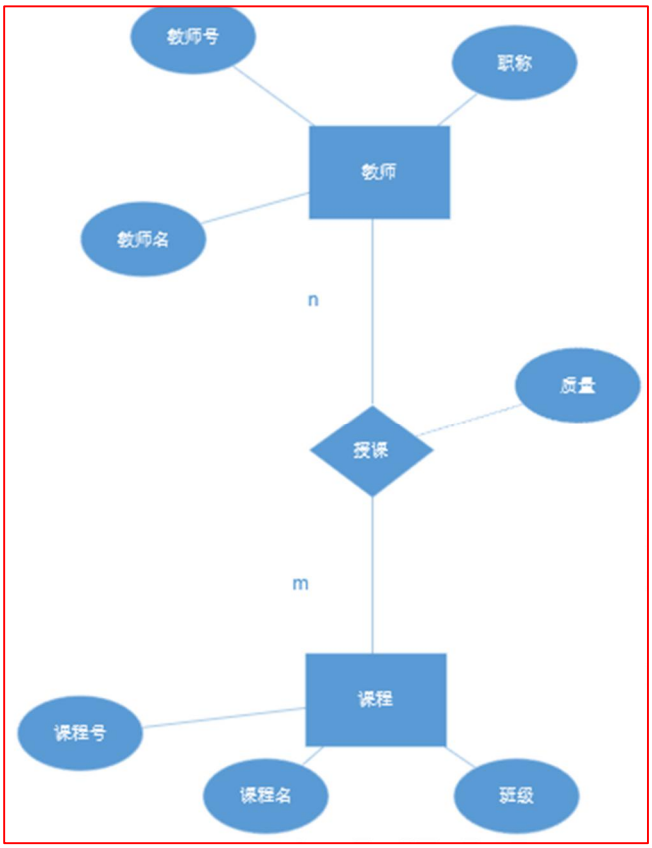
在“讲授”联系中应能反映出教师的授课程质量。如多对多 E-R 图的示。



一对一 E-R 图



一对多 E-R 图



多对多 E-R 图

2 创建数据库和表

2.1 创建配置数据库

1、MySQL 基本知识

（1）**MySQL 是一种关系型数据库管理系统**。

（2）目前主流关系型数据库管理系统之一。

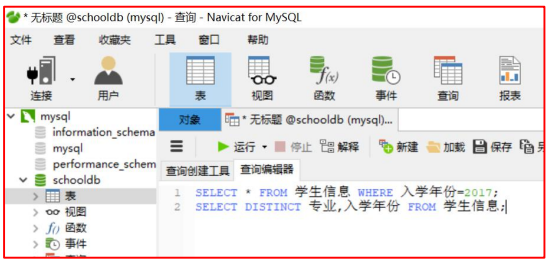
（3）**使用的 SQL（结构化查询语言）语言是用于访问数据库的最常用标准化语言**。

（4）**MySQL 有着体积小、速度快、总体拥有成本低等优点。**

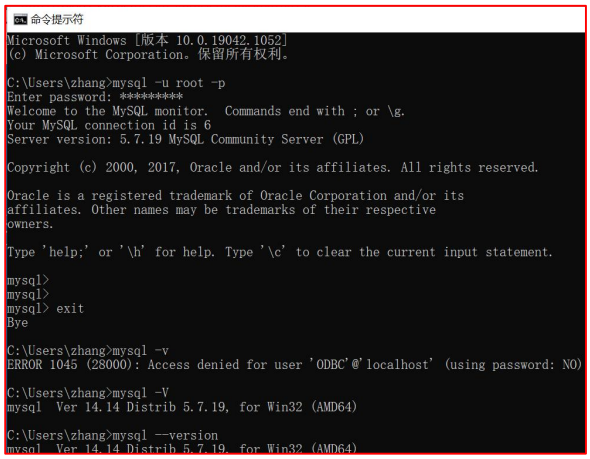
（5）**MySQL 是开源软件（自由软件）**。

（6）使用 MySQL 前需要先安装，可以使用界面工具（安装 Navicat for MySQL）进行操作它，也可以通过系统命令行窗口进行操作。重点是要掌握 SQL 各种语句的操作来实现数据库的应用。

软件界面：



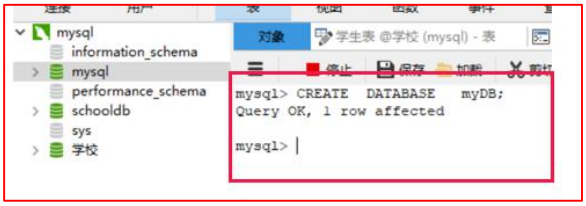
命令行界面（注意环境变量中加入 mysql 的安装路径）：



2、**创建数据库（重点是语句创建）**

（1）语法：**CREATE DATABASE** 数据库名;

（2）举例：**CREATE DATABASE** myDB;



（3）**如果是 cmd 命令行创建数据库，最好指定字符集。不然容易后面报建表结构时，有默认值为中文时会报错。**例如：

sex VARCHAR(2) DEFAULT '男',
创建表中性别一项：sex VARCHAR(2) DEFAULT '男',
运行错误提示：ERROR 1067 (42000): Invalid default value for 'sex'; 此时就需要修改字符集配置信息。

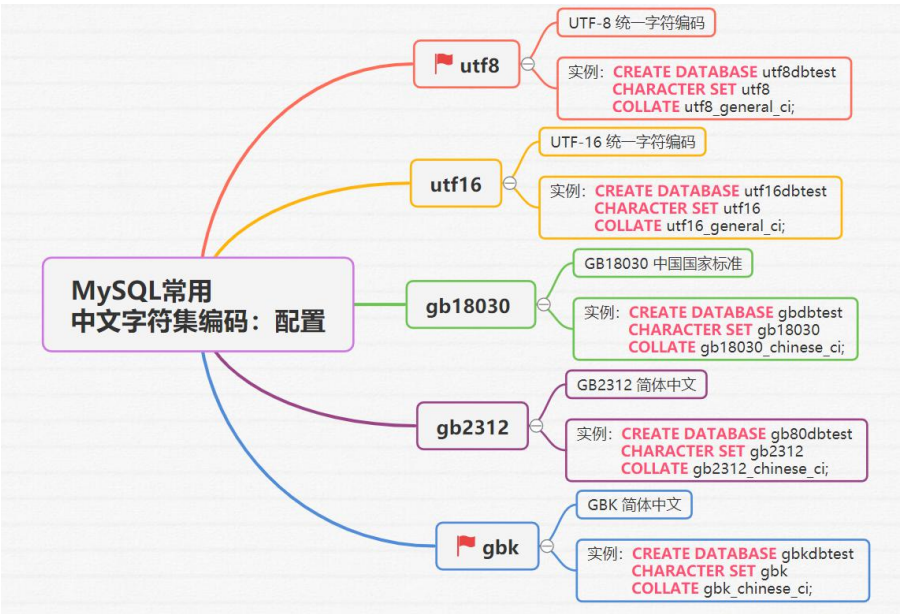
3、创建时配置数据库字符编码

（1）创建数据库时,设置数据库的编码方式:

CHARACTER SET:指定数据库采用的字符集,utf8 不能写成 utf-8

COLLATE:指定数据库字符集排序规则,utf8 的默认排序规则为 utf8_general_ci，ci 是不区分大小写。

（2）MySQL 创建数据库时**常用字符集编码**(图中都是默认排序规则):



5、查看字符集编码

（1）查看所有的字符编码

```
SHOW CHARACTER SET;
```

（2）查看创建数据库的指令并查看数据库使用的编码

```
SHOW CREATE DATABASE  test;
```

```
mysql> SHOW CREATE DATABASE  test;
+----------+-----------------------------------------------------------------+
| Database | Create Database                                                 |
+----------+-----------------------------------------------------------------+
| test     | CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET utf8 */ |
+----------+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

（3）查看数据库编码：

```
SHOW VARIABLES LIKE '%char%';
```

```
mysql> SHOW  VARIABLES LIKE '%char%';
+--------------------------+-----------------------------------------------------------+
| Variable_name            | Value                                                     |
+--------------------------+-----------------------------------------------------------+
| character_set_client     | gbk                                                       |
| character_set_connection | gbk                                                       |
| character_set_database   | latin1                                                    |
| character_set_filesystem | binary                                                    |
| character_set_results    | gbk                                                       |
| character_set_server     | latin1                                                    |
| character_set_system     | utf8                                                      |
| character_sets_dir       | C:\Program Files (x86)\MySQL\MySQL Server 5.7\share\charsets\ |
+--------------------------+-----------------------------------------------------------+
8 rows in set, 8 warnings (0.01 sec)
```

6、**修改数据库编码**

（1）语法：

ALTER DATABASE 数据库名

CHARACTER SET 字符集

COLLATE 排序规则；

（2）实例：

```
--修改 dbtest 字符集为 gbk,字符集排序为 gbk_chinese_ci
ALTER DATABASE dbtest CHARACTER SET GBK COLLATE gbk_chinese_ci;
--修改 dbtest 字符集为 utf8,字符集排序为 utf8_ general _ci
ALTER DATABASE dbtest CHARACTER SET utf8 COLLATE utf8_general_ci;
```

7、查看已经存在的数据库信息

（1）语法：**SHOW DATABASES**;

（2）实例：

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| gb1231db           |
| mysql              |
| performance_schema |
| schooldb           |
| sys                |
| test               |
| utf8dbtest         |
+--------------------+
8 rows in set (0.00 sec)
```

系统 CMD 命令行界面操作

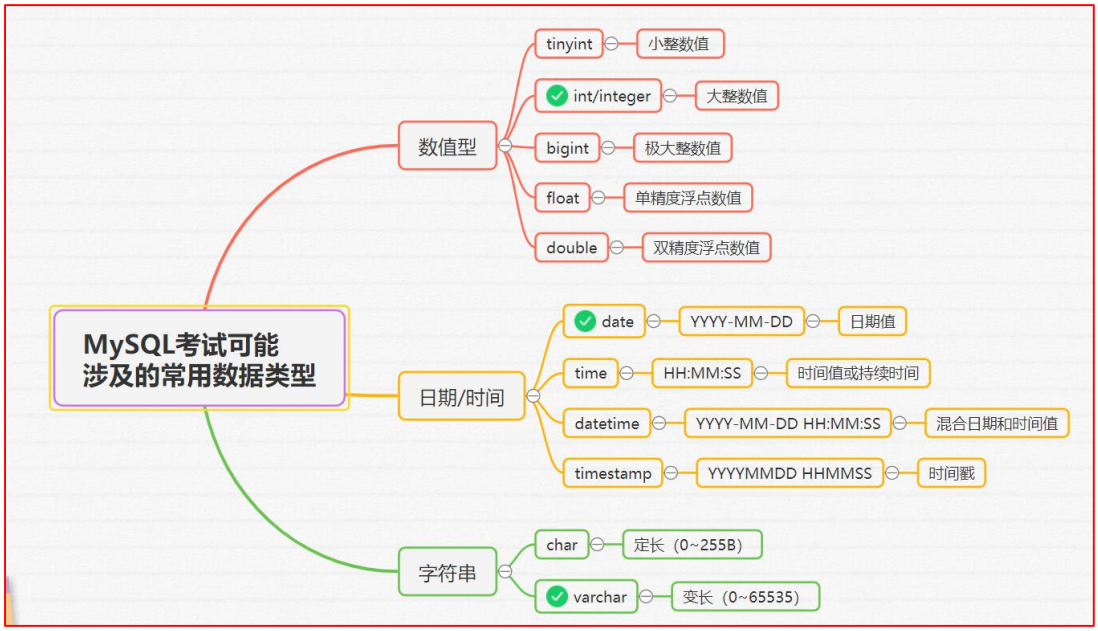
```
mysql>
mysql>
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| gb1231db           |
| mysql              |
| performance_schema |
| schooldb           |
| sys                |
| test               |
| utf8dbtest         |
+--------------------+
8 rows in set

mysql>
```

Navicat 软件中的命令行界面操作

2.2 数据类型与创建表

1、MySQL 常用数据类型



2、创建表的格式

（1）格式

```
CREATE TABLE 表名（
        属性名 1 类型   其他（默认值 ，是否为空、主键、自动增长等），
        属性名 2 类型   其他（默认值 ，是否为空、主键、自动增长等），
        ……
        [PRIMARY KEY(主键属性名)]
        [FOREIGN KEY(外键属性名)  REFERENCES 主表名（属性名）]
   )
```

（2）是否为空

是：NULL

否：NOT NULL

（3）默认值

```
sSex varchar(255) NOT NULL DEFAULT '男' COMMENT '性别'
```

（4）自动增长：AUTO_INCREMENT

（4）PRIMARY KEY(主键属性名)：多个属性之间使用逗号分隔

（5）FOREIGN KEY(外键属性名)：多个属性之间使用逗号分隔； REFERENCES：引用参照

（6）实例：

```
CREATE TABLE t_students (
  id        INT(11) NOT NULL AUTO_INCREMENT COMMENT '编号',
  sNo       INT(11) NOT NULL COMMENT '学号',
  sName     VARCHAR(255) NOT NULL COMMENT '姓名',
```

```
  sSex      VARCHAR(255) NOT NULL DEFAULT '男' COMMENT '性别',
  sDate     DATETIME NOT NULL COMMENT '出生日期',
  sMask     VARCHAR(255) DEFAULT NULL COMMENT '备注',
  sPartNo   INT(11) NOT NULL COMMENT '院系',
  PRIMARY KEY (id)
  FOREIGN KEY(sPartNo)  REFERENCES  t_department（sPart）
)
```

2.3 修改表结构

1、**修改表名**

（1）语法：**ALTER TABLE 原表名 RENAME TO 新表名；**

（2）实例：

```
ALTER TABLE t_students RENAME TO t_students1;
```

2、**修改属性**

（1）修改某个列的列名和类型。

①、语法 ： **ALTER TABLE 表名 CHANGE 原列名 新列名 新数据类型；**

②、实例：

```
ALTER TABLE t_students1 CHANGE id sId BIGINT;
```

（2）修改某个列的类型长度

①、语法 **ALTER TABLE 表名 MODIFY COLUMN 字段名 数据类型(修改后的长度)；**

②、实例：

```
ALTER TABLE t_students1 MODIFY COLUMN id INT(20);
```

3、**删除属性**

（1）语法：**ALTER TABLE 表名 DROP COLUMN 列名；**

（2）删除一列属性：删除一列时可以省写 COLUMN

```
ALTER TABLE 表名 DROP [COLUMN] 列名;
```

（3）删除多列属性:删除多列时不可以省写 COLUMN

```
ALTER TABLE 表名 DROP COLUMN 列名 1,
                 DROP COLUMN 列名 2,
                 ……
                 DROP COLUMN 列名 n;
```

4、**添加新属性**

（1）语法：**ALTER TABLE 表名 ADD COLUMN 列名 数据类型;**

（2）实例：

```
ALTER TABLE t_students1 ADD COLUMN   sMask VARCHAR(255);
```

5、**删除主键**

（1）语法：**ALTER TABLE 表名 DROP PRIMARY KEY;**

（2）实例：

```
ALTER TABLE t_students1 DROP PRIMARY KEY;
```

6、**添加主键**

（1）语法：**ALTER TABLE 表名 ADD PRIMARY KEY (列名组);**

（2）实例：

```
ALTER TABLE t_students1 ADD PRIMARY KEY (sId);
```

2.4 删除表和数据库

1、**删除表结构**（包括所有数据）

（1）语法：**DROP TABLE 表名；**

（2）实例：

```
DROP TABLE t_students1;
```

2、**批量删除表结构**（包括所有数据）

（1）语法：**DROP TABLE 表名 1,表名 2，表名 3……；**

（2）实例：

```
DROP TABLE t_students1, t_students2,t_students3;
```

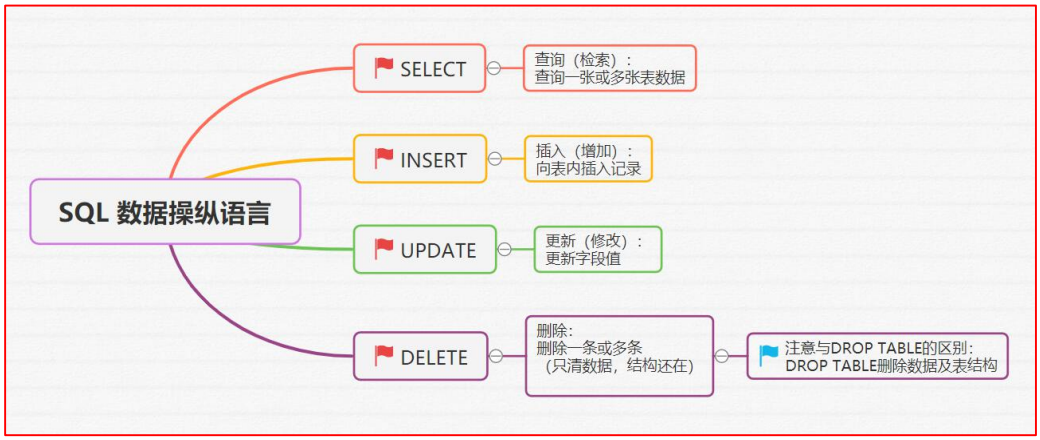
3、**删除数据库**

（1）语法：**DROP TABLE 数据库名；**

（2）实例：

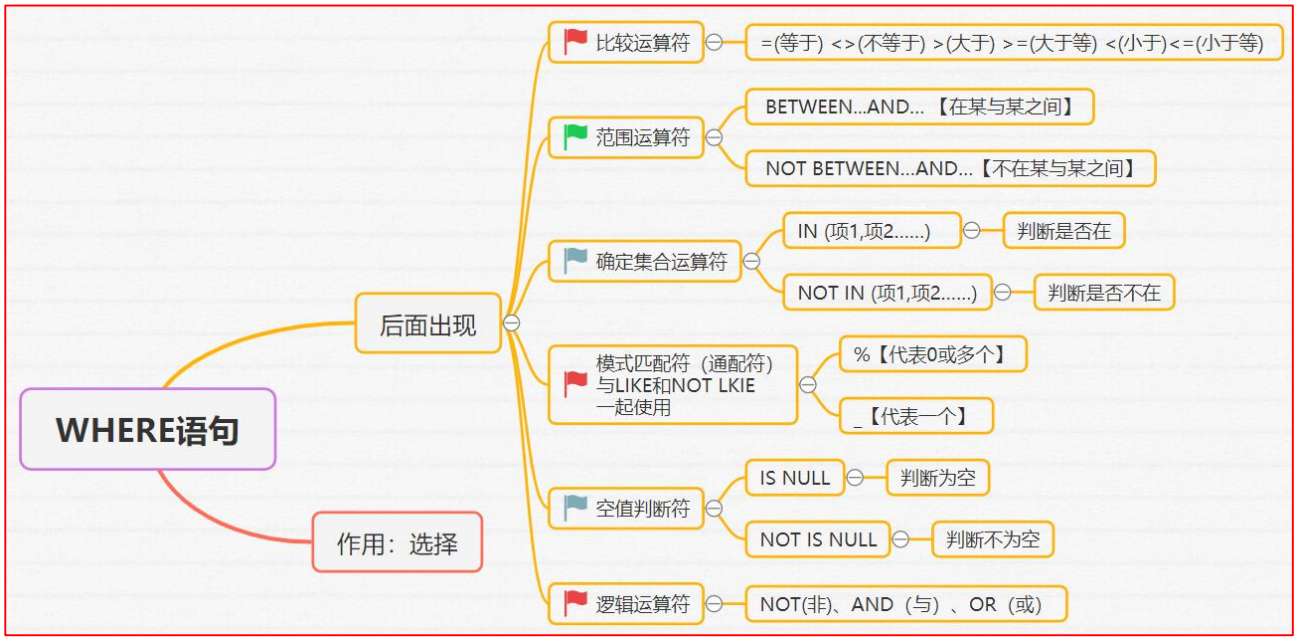
```
DROP DATABASE myDB;
```

3．SQL 数据操纵语言



3.1 数据的增删改处理

1、**WHERE 语句后的运算符**



2、**INSERT 语句（增加、插入操作）**

（1）**插入一条记录**

①、语法：：**INSERT INTO 表名（属性名列表）**

VALUES(属性值 1，属性值 2，属性值 3，.....);

②、实例：

```
INSERT INTO 学生表（id,name,sex,bdate） VALUES('S02021'，'张志虎'，'男'，'1990-9-30');
```

（2）**插入查询出来结果**：

①、语法：**INSERT INTO 表名 （属性名列表）**

SELECT * FROM 表名 1;

②、实例：

```
INSERT INTO t_tepm(name, age)
            SELECT name,age
            FROM t_students
            WHERE age>=19
```

3、DELETE 语句（删除）

（1）清空表记录（删除全部记录）

①、语法：DELETE FROM 表名;

②、实例：

```
DELETE FROM tb1;
```

（2）删除满足条件的记录

①、语法：DELETE FROM 表名 WHERE 复合条件表达式;

②、实例：

```
DELETE FROM tb1 WHERE sex= '男';
```

4、UPDATE 语句（更新、修改）

（1）更新全表中相关属性的值

①、语法：UPDATE 表名 SET 属性名 1 = 新值[，属性名 2=新值 ,…..]；

②、实例：

```
UPDATE 学生表 SET  sGrade='大三';
```

（2）更新全表中满足条件记录的相关属性的值;

①、语法：

```
UPDATE 表名 SET 属性名 1 = 新值[，属性名 2=新值 ,…..]
            WHERE 属性 1=值；
```

②、实例：

```
UPDATE 学生表 SET  sGrade='大三' WHERE ID='sz00001';
```

3.2 简单查询与连接查询

1、简单查询:[ ]表示可以缺省不写

（1）查询全表记录

①、语法：SELECT * FROM 表名 [AS] 别名];

②、实例：

```
SELECT  *  FROM  t_student;
SELECT  *  FROM  t_student AS S;
```

（2）查询指定列的全部记录;

①、语法：SELECT 属性名 1 [AS] 属性 1 [AS]别名,
属性名 2 [AS] 属性名 2 别名,
.....
FROM 表名 [[AS] 别名]；

②、实例：

```
SELECT sName AS 姓名,
       sAge  年龄
```

```
FROM t_student;
```

（3）**查询满足条件的记录；**

①、语法：**SELECT * FROM 表名 [[AS] 别名] WHERE 复合条件表达式；**

②、实例：

```
SELECT * FROM t_student AS S WHERE S.ID ='SZ1000001';
SELECT * FROM t_student WHERE ID ='SZ1000001';
```

（4）**去除重复记录(DISTINCT)**

①、语法：**SELECT *DISTINCT* * FROM 表名 [[AS] 别名]**

②、实例：

```
SELECT DISTINCT * FROM t_student;
```

2、**连接查询**

3、**等值连接**

（1）语法关键字：**A 表名 INNER JOIN B 表名 ON 等值连接条件**

（2）以上关键字 **INNER JOIN ON** 也可以不写

（3）**一般通过外键来进行等值连接**

（4）实例：

```
SELECT A.id AS ID，A.name AS Name, A.age AS Age,B.name AS Grade
FROM t_students AS A,t_grade AS B
WHERE A.class = B.id;
或
SELECT A.id AS ID，A.name AS Name, A.age AS Age,B.name AS Grade
FROM t_students AS A
INNER JOIN t_grade AS B
ON A.class = B.id;
```

4、**非等值连接**

（1）语法与等值连接相同

（2）**使用非'='的其他比较运算符作为匹配条件**

（3）实例：

```
SELECT A.id AS ID，A.name AS Name, A.age AS Age,B.name AS Grade
```

```
FROM t_students AS A,t_grade AS B
WHERE A.class > B.id;
或
SELECT A.id AS ID，A.name AS Name, A.age AS Age,B.name AS Grade
FROM t_students AS A
INNER JOIN t_grade AS B
ON A.class > B.id;
```

5、**自然连接**

（1）语法关键字：A 表名 NATURAL JOIN B 表名

（2）**连接时需要有相同属性，且会删除结果中重复的属性。**

（3）实例：

```
SELECT * FROM t_students NATURAL JOIN t_grade
```

6、**自连接**

（1）**连接自己，即自连接在自连接查询过程中，一定要设置表的别名**

（2）一个表中具有上司与下属关系的字段。比如员工表中，存在一个经理字段，表示员工的上司。

（3）实例

```
SELECT A.name AS 员工名,B.name AS 经理
FROM 员工表 AS A，员工表 AS B
WHERE A.leader = B.id
```

7、**左外连接**

（1）语法关键字： A 表名 LEFT [OUTER] JOIN B 表名 ON 条件

（2）实例：

```
SELECT A.id AS 学号, A.name AS 姓名,A.age AS 年龄,B.name AS 年级
FROM 学生表 AS A
LEFT JOIN 年级表 AS B
ON A.class = B.id;
或
SELECT A.id AS 学号, A.name AS 姓名,A.age AS 年龄,B.name AS 年级
FROM 学生表 AS A
LEFT OUTER JOIN 年级表 AS B
ON A.class = B.id;
```

8、**右外连接**

（1）语法关键字：A 表名 RIGHT [OUTER] JOIN B 表名 ON 条件

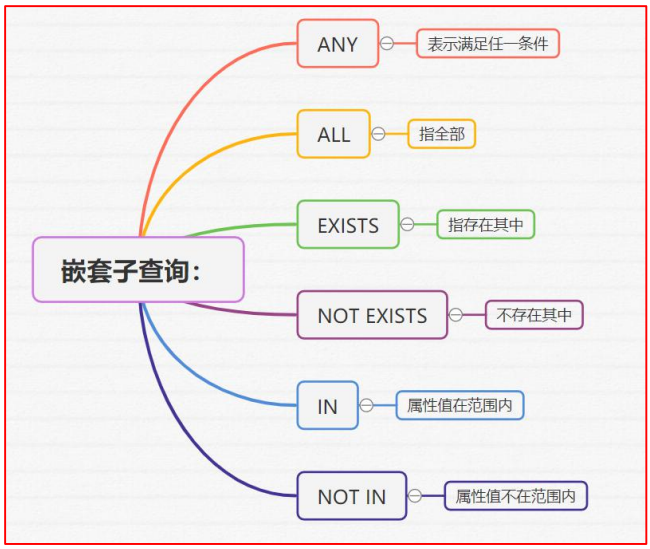
（2）实例

```
SELECT B.id AS 学号, B.name AS 姓名,B.age AS 年龄,A.name AS 年级
FROM 年级表 AS A
LEFT JOIN 学生表 AS B
ON B.class = A.id;
或
SELECT B.id AS 学号, B.name AS 姓名,B.age AS 年龄,A.name AS 年级
```

```
FROM 年级表 AS A
LEFT  OUTER  JOIN 学生表 AS B
ON   B.class = A.id;
```

3.3 嵌套子查询

1、嵌套子查询



2、ANY 关键字的子查询

（1）语法关键字：ANY

（2）实例：

```
SELECT   age
FROM tb1
WHERE age > ANY ( SELECT age   FROM   tb2);
```

3、ALL 关键字的子查询

（1）语法关键字：ALL

（2）实例：

```
SELECT   age
FROM tb1
WHERE age > ALL( SELECT age   FROM   tb2);
```

4、EXISTS 关键字的子查询

（1）语法关键字：EXISTS

（2）实例：

```
SELECT   *
FROM   tb1
WHERE EXISTS(SELECT age FROM tb1 WHERE   age   > 19 );
```

5、NOT EXISTS 关键字的子查询

（1）语法关键字：NOT EXISTS

（2）实例：

```
SELECT   *
```

```
FROM    tb1
WHERE NOT EXISTS (SELECT age FROM tb1 WHERE    age    < 19 )
```

6、IN 关键字的子查询

（1）语法关键字：IN

（2）实例：

```
SELECT    *
FROM    tb1
WHERE NOT EXISTS (SELECT age FROM tb1 WHERE    age    < 19 )
```

7、NOT IN 关键字的子查询

（1）语法关键字：NOT IN

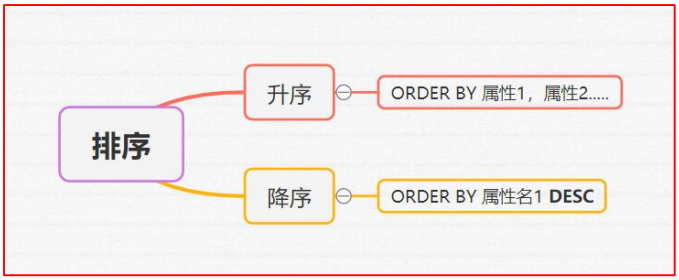
（2）实例：

```
SELECT *
FROM tb1
WHERE age NOT IN(SELECT age FROM tb2);
```

3.4 排序、计算及分组

1、查询结果的排序

（1）语法：



（2）实例：

```
--按年龄降序
SELECT * FROM tb    ORDER BY    age DESC;
--按年龄升序
SELECT * FROM tb    ORDER BY    age;
```

2、查询结果的计算

（1）语法：查询字段中定义计算字段，比如使用算术运算符来计算。

（2）实例：

```
SELECT    ST.id ,ST.`name`,ST.age,ST.cs,ST.math ,(ST.cs+ST.math) AS sum
FROM t_students AS ST ;
```

3、查询结果的分组

（1）语法：GROUP BY 属性名 1[,属性名 2,属性名 3……] [HAVING 条件]

（2）使用注意事项：

①、HAVING 只能用在 GROUP BY 之后，对分组后的结果进行筛选(即使用 HAVING 的前提条件是分组)。

②、WHERE 肯定在 GROUP BY 之前，即也在 HAVING 之前。

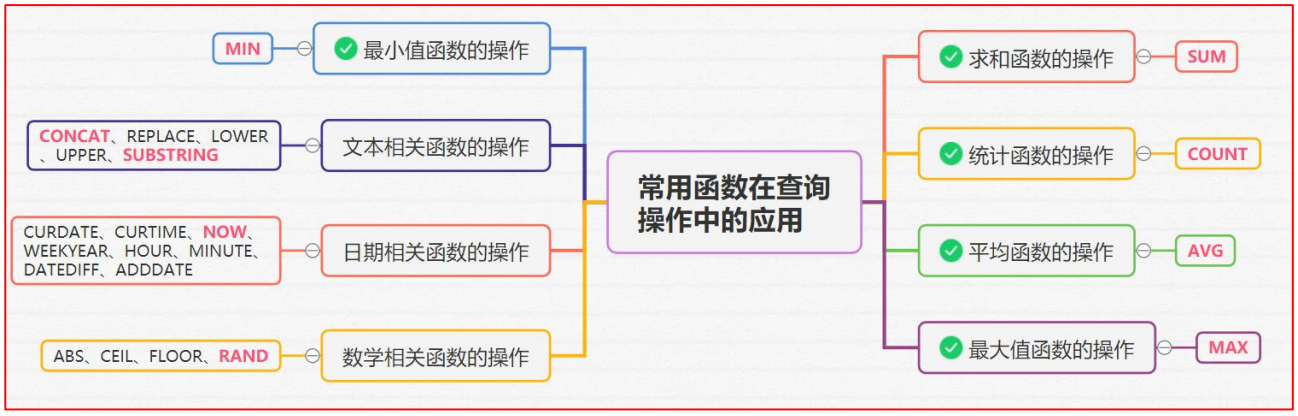
③、WHERE 后的条件表达式里不允许使用聚合函数（COUNT(),SUM(),AVG(),MAX(),MIN()），而 HAVING 可以。

（2）实例：

```
--统计按性别分组的人数
SELECT sex, COUNT(sex) AS counts
FROM t_students
GROUP BY sex;
--统计性别超过 100 人数
SELECT sex, COUNT(sex) AS counts
FROM t_students
GROUP BY sex
HAVING COUNT(sex) >100;
```

3.5 常用的函数应用

1、常用函数的在查询操作中的应用



2、求和函数的操作

（1）语法：SUM(属性名或公式) [AS 别名]

（2）功能：属于常用聚合函数，返回属性（字段）的和

（3）实例：

```
--全班数总成绩
SELECT SUM(math) AS 总成绩
FROM 成绩表 WHERE 班级号= 'cs001 ';
--计算销售额提成
SELECT SUM(销售额* 0.10) AS 提成
FROM 销售表;
```

3、统计函数的操作

（1）语法：COUNT(属性名) [AS 别名]

（2）功能：属于常用聚合函数，返回某属性（字段）统计的行数

（3）实例：

```
SELECT sex 性别,COUNT(sex) AS 人数
```

```
FROM 学生表
WHERE sex='男';
```

4、**平均函数的操作**

（1）语法：**AVG(属性名) [AS 别名]**

（2）功能：**属于常用聚合函数，返回某属性（字段）的平均值**

（3）实例：

```
--男生的数学成绩平均分
SELECT AVG(math) AS 平均成绩
FROM 学生表
WHERE sex='男';
```

5、**最大值函数的操作**

（1）语法：**MAX(属性名) [AS 别名]**

（2）功能：**属于常用聚合函数，返回某属性（字段）的最大值**

（3）实例：

```
--查询出最大年龄的学生信息
SELECT * from 学生表
WHERE AGE IN(SELECT MAX(age) FROM 学生表);
```

6、最小值函数的操作

（1）语法：**MIN(属性名) [AS 别名]**

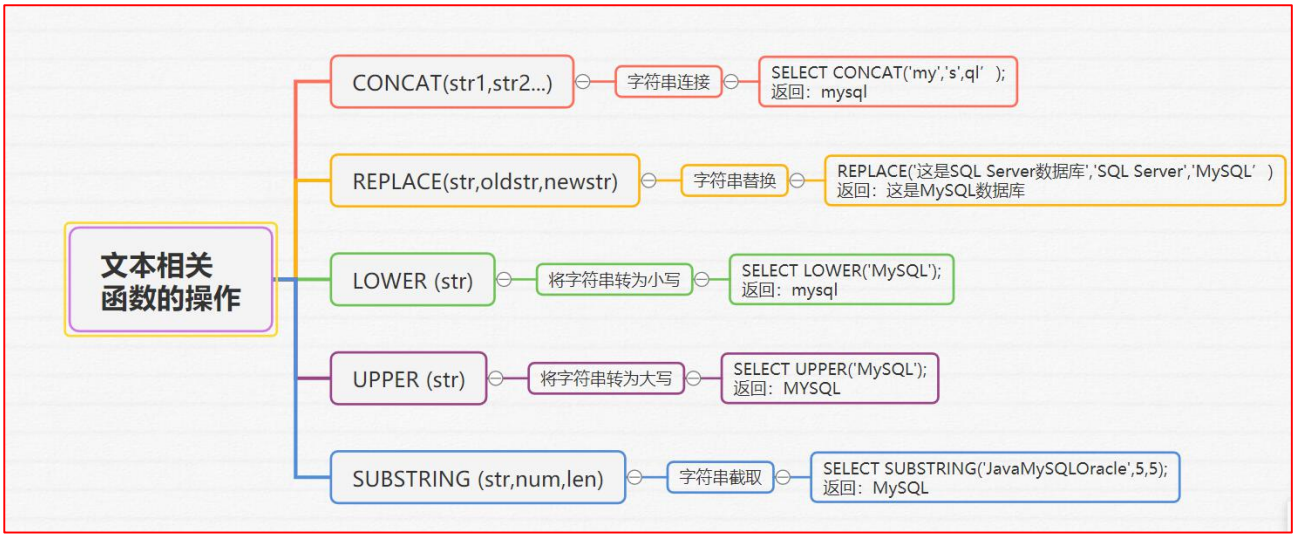
（2）功能：**属于常用聚合函数，返回某属性（字段）的最小值**

（3）实例：

```
--查询出最小的年龄
SELECT MIN(age) AS 最小年龄
FROM 学生表;
```

7、**文本相关函数的操作**

（1）文本相关函数介绍：



（2）实例：

```
1  CONCAT('my','s','ql'),REPLACE('这是SQL Server数据库','SQL Server','MySQL'),
```

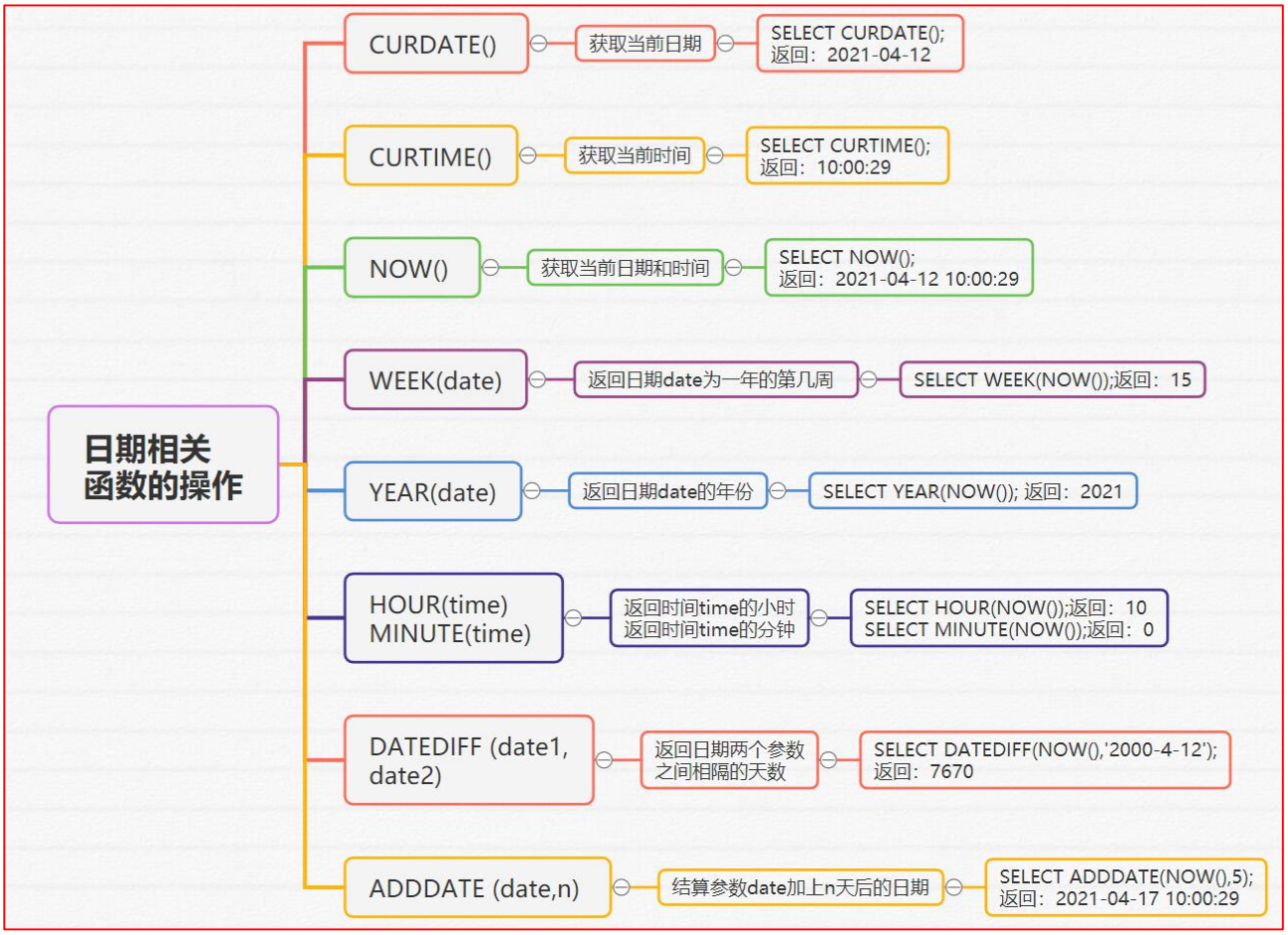
```
LOWER('MySQL'),UPPER('MySQL'),SUBSTRING('JavaMySQLOracle',5,5);
```

信息 | 结果1 | 概况 | 状态

CONCAT('my','s','ql')	REPLACE('这是SQL Server	LOWER('MySQL')	UPPER('MySQL')	SUBSTRING('JavaMySQLC
mysql	这是MySQL数据库	mysql	MYSQL	MySQL

8、**日期相关函数的操作**

（1）日期相关函数介绍



（2）实例：

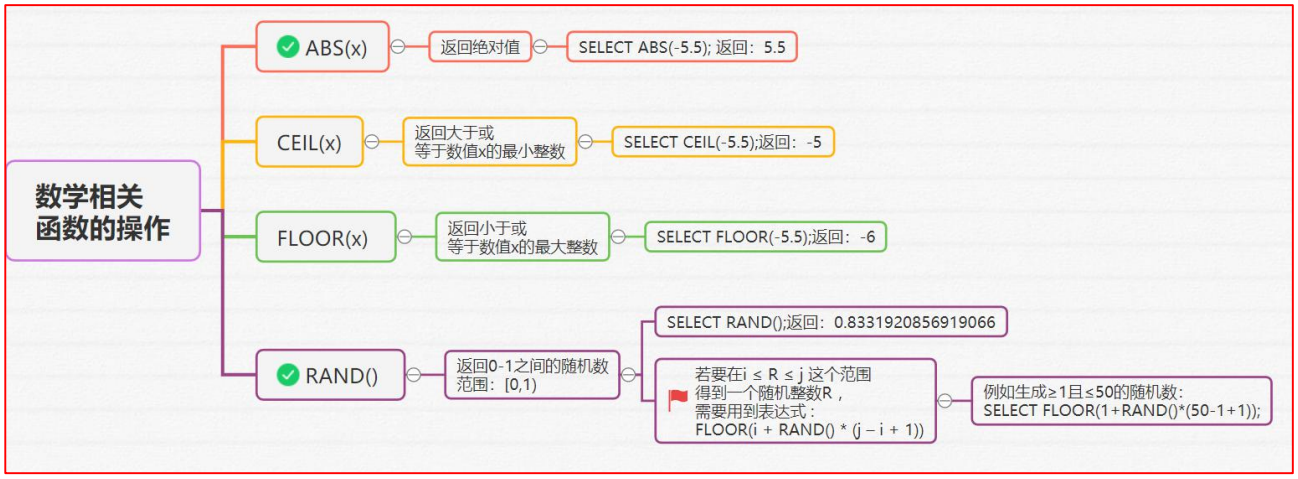
```
1  SELECT CURDATE() ,CURTIME() ,NOW(),WEEK(NOW()),YEAR(NOW()),HOUR(NOW()),
MINUTE(NOW()),DATEDIFF(NOW(),'2000-4-12'),ADDDATE(NOW(),5);
```

信息 | 结果1 | 概况 | 状态

CURDATE()	CURTIME()	NOW()	WEEK(NOW())	YEAR(NOW())	HOUR(NOW())	MINUTE(NOW())	DATEDIFF(NOW(),'2000-4	ADDDATE(NOW(),5)
2021-04-12	10:00:29	2021-04-12 10:00:29	15	2021	10	0	7670	2021-04-17 10:00:29

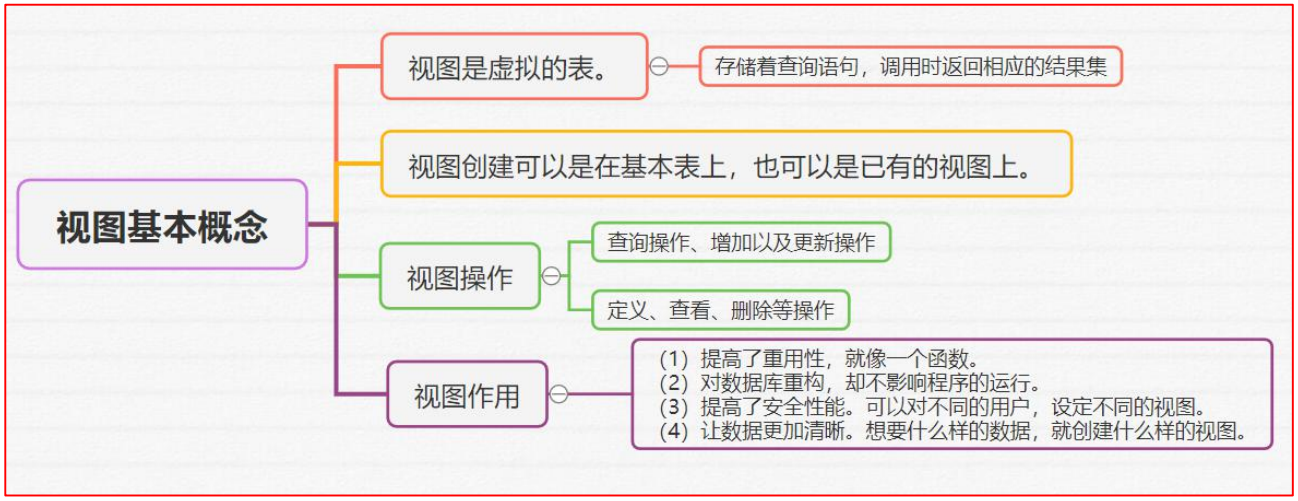
9、**数学相关函数的操作**

（1）数学相关函数介绍:重点是 **RAND 函数**



（3）实例：

4．视图和索引



4.1 创建和查看视图

1、直接在表上创建视图

（1）语法：

CREATE VIEW 视图名 AS (对表查询语句)；

（2）实例：

--将学生表与年级表进行等值连接后的查询结果集定义为 v_student1 视图。

```
1> CREATE VIEW v_student1 AS
select
a.id AS 编号,
a.name AS 姓名,
a.sex AS 性别,
a.age AS 年龄,
a.bdate AS 出生日期,
a.cs AS 计算机,
a.math AS 数学,
b.name AS 年级
from (t_students a join t_grade b)
where (a.class = b.id);
```

2、直接在视图上创建新视图

（1）语法：

CREATE VIEW 视图名 AS (对视图进行查询)；

（2）实例：

```
CREATE VIEW v_student2 AS
SELECT * FROM v_student1;
```

3、查看视图

（1）语法：**SHOW CREATE VIEW 视图名；**

（2）实例：

```
SHOW CREATE VIEW v_student1;
```

4、删除视图

（1）语法：**DROP VIEW 视图名；**

（2）实例：

```
--通过执行 SQL 语句实现删除 v_student2 视图的信息。
DROP VIEW v_student2;
```

4.2 用视图检索和修改基本表中数据

1、视图检索（查询）

（1）语法：**SELECT * FROM 视图名；**

（2）使用说明：使用视图就像和使用基本表一样，可以简单查询，也可以联合查询（UNION）,也可连接查询，也可以排序、分组等，**视图不能索引操作**。

（3）实例：

```
SELECT * FROM v_student1;
```

2、视图修改本表中数据

（1）**MySQL 允许一定条件下通过视图对基本表的数据修改（增加、删除、修改）操作。**

（2）**如果视图中包含以下是不被允许更新操作的：**

3、通过视图向基本表中插入操作

（1）语法：**INSERT INTO 表名（列）VALUES(值)；**

（2）实例：

```
--v_temp1 视图中插入一条记录
INSERT INTO v_tepm1(name, age)
    VALUES('吴越',19);
```

4、通过视图向基本表中修改操作

（1）语法：**UPDATE 视图名 SET 属性名=新值 [WHRER 条件]；**

（2）实例：

```
--v_temp1 视图将吴越年龄加 1
UPDATE v_tepm1
      SET age = age+1
      WHERE name =‘吴越’;
```

5、通过视图向基本表中删除操作

（1）语法：**DELETE FROM 视图名 [WHRER 条件]；**

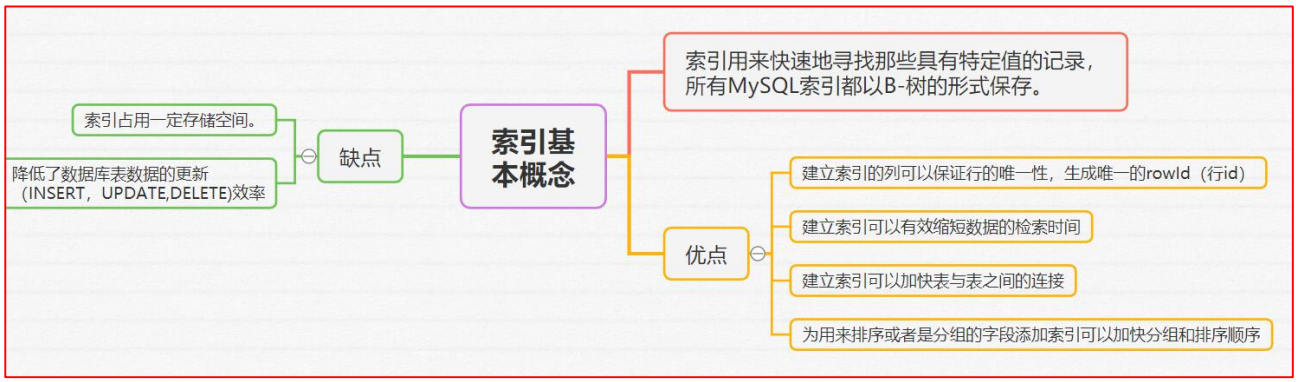
（2）实例：

```
--v_temp1 视图将姓名叫吴越的记录删除
DELETE
FROM v_tepm1
WHERE name =‘吴越’;
```

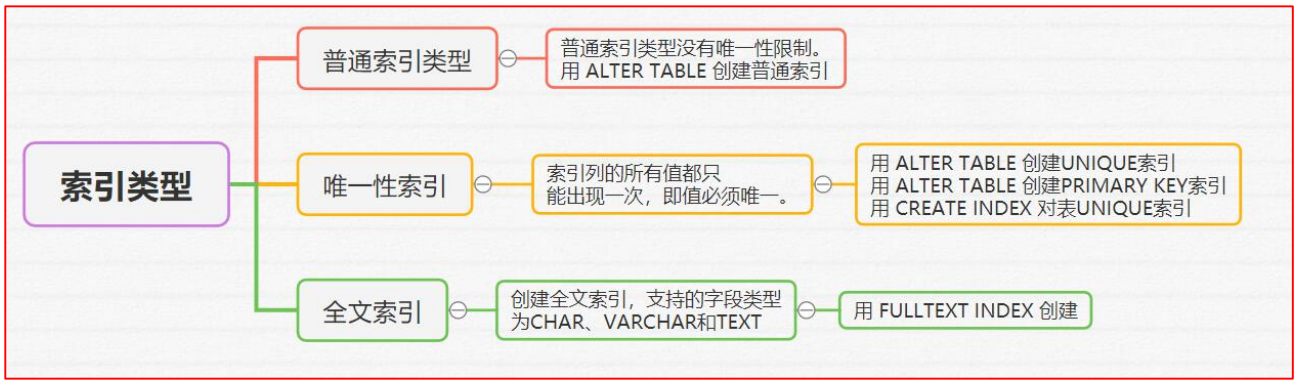
4.3 创建索引

1、索引基本概念

（1）索引及优缺点



（2）索引的类型



2、创建索引

（1）**创建表时候建索引**

```
CREATE TABLE projectfile (
    id INT AUTO_INCREMENT COMMENT '附件 id',
    fileuploadercode VARCHAR(128) COMMENT '附件上传者 code',
    projectid INT COMMENT '项目 id;此列受 project 表中的 id 列约束',
    filename VARCHAR (512) COMMENT '附件名',
    --主键索引
    PRIMARY KEY (id),
    FOREIGN KEY (projectid) REFERENCES project (id),
    --给 fileuploadercode 字段创建普通索引
    UNIQUE INDEX (projectid),
    --指定使用 INNODB 存储引擎(该引擎支持事务)、utf8 字符编码
    INDEX (fileuploadercode),
    --附件上传者 code 和附件名加上全文索引的表
    FULLTEXT INDEX (fileuploadercode,filename) WITH PARSER ngram
) ENGINE = INNODB DEFAULT CHARSET = utf8 COMMENT '项目附件表';
```

注意：**在全文索引中，n-gram 就是一段文字里面连续的 n 个字的序列**。

（2）**ALTER TABLE 创建（修改表结构时创建）**

①、创建普通索引

- 语法：

```
ALTER TABLE table_name ADD INDEX index_name (column_list);
```

- 实例：

```
ALTER TABLE t_tepm ADD INDEX idxName(name);
```

②、创建 UNIQUE 索引

- 语法：

```
ALTER TABLE table_name ADD UNIQUE (column_list);
```

- 实例：

```
ALTER TABLE t_tepm ADD UNIQUE (id);
```

（3）CREATE INDEX 创建

①、创建普通索引

- 语法：

```
CREATE INDEX index_name ON table_name (column_list);
```

- 实例：

```
CREATE INDEX idxStuName ON t_tepm(name);
```

②、创建 UINIQUE 索引

- 语法：

```
CREATE UNIQUE INDEX index_name ON table_name (column_list);
```

- 实例：

```
CREATE UNIQUE INDEX idxID    ON t_tepm(id);
```

③、创建全文索引

- 语法：

```
CREATE FULLTEXT INDEX index_name ON table_name (column_list)    WITH PARSER ngram;
```

- 实例：

```
CREATE FULLTEXT INDEX ngram_idx ON articles(title) WITH PARSER ngram;
```

3、使用索引

（1）自然语言搜索

```
SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('精神');
```

（2）BOOLEAN MODE： + 表示必须包含，-表示必须不包含

```
SELECT * FROM articles WHERE MATCH (title,body) AGAINST ('+精神' IN BOOLEAN MODE);
```

4、删除索引

（1）DROP INDEX 索引名 ON 表名

（2）ALTER TABLE 表名 DROP INDEX 索引名