## **1. 数据库能够做什么**

* 存储大量数据，方便检索和访问
* 保持数据信息的一致、完整
* 共享和安全
* 通过组合分析，产生新的有用信息

## **2. 数据库的基本概念**

### **2.1 实体**

只要是在客观世界存在的、可以被描述出来的都是实体

### **2.2 数据库(DB)**

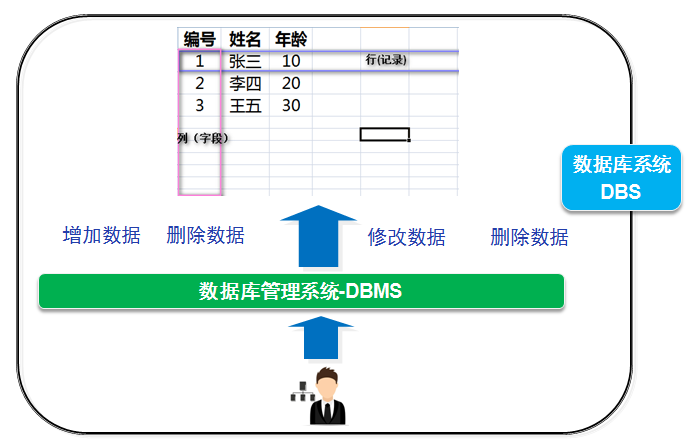
* 数据库就是数据的仓库，可以存放结构化的数据

### **2.3 数据库管理系统(DBMS)**

是一种系统软件，提供操作数据库的环境，可以通过数据库管理系统对数据进行插入、修改、删除和查询等操作。

### **2.4 SQL**

结构化查询语言 专门用来和数据库进行交流的语言,几乎所有的DBMS都支持SQL

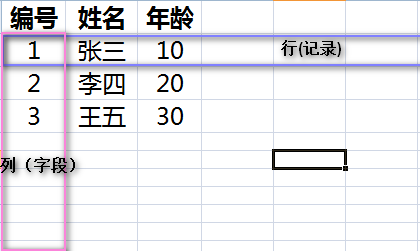


### **2.5 SQL规范**

1. SQL语句不区分大小写，建议SQL关键字大写，表名和列表小写
2. 命令用分号结尾
3. 命令可以缩进和换行，一种类型的关键字放在一行
4. 可以写单行和多行注释 , #和--是单行注释，/\*\*\*/多行注释

## **3. 数据表**

* 表是数据库中包含所有数据的数据库对象，也是其它对象的基础。
* 表定义是一个列的集合，数据在表中是按行和列的格式组织的，用来存放数据
* 行也称为记录用来存放一个个实体，列称为字段用来描述实体的某一个属性 学生管理系统



## **4.MYSQL简介**

### **4.1 MYSQL特点**

* 开源免费
* 性能高
* 安装使用都简单

### **4.2 MYSQL安装**

* [mysql下载](https://dev.mysql.com/downloads/mysql/5.5.html)
* 安装MYSQL

### **4.3 MYSQL配置**

C:\Program Files\MySQL\MySQL Server 5.5\my.ini

* port 端口号
* basedir 安装目录
* datadir 数据存放访目录
* character\_set\_server 字符集
* default-storage-engine 存储引擎
* sql-mode 语法模式
* max-connections 最大连接数

### **4.4 MYSQL启动和停止**

net start MySQL

net stop MySQL

### **4.5 通过命令行连接MYSQL**

mysql -h 127.0.0.1 -P 3306 -uroot -p123456

exit

### **4.6 切换数据库**

use test;

### **4.7 显示有哪些表**

show tables;

show tables **from** mysql;

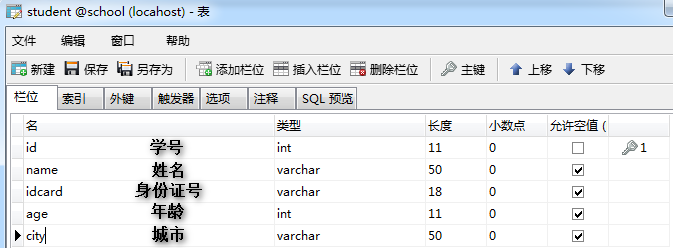
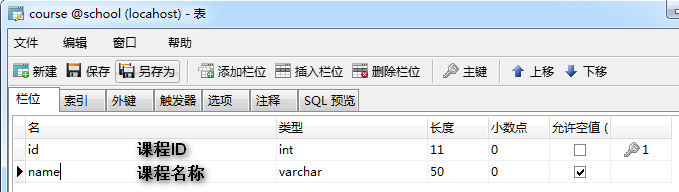
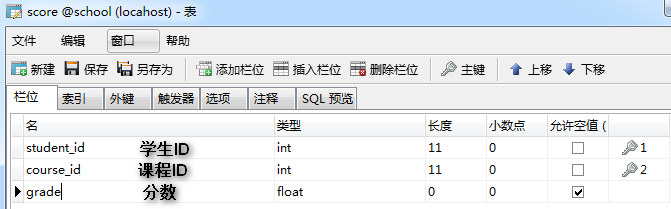
### **4.8 显示当前数据库**

select database();

### **4.9 查询表结构**

DESC user;

## **5. 创建表**

## **6. 数据完整性**

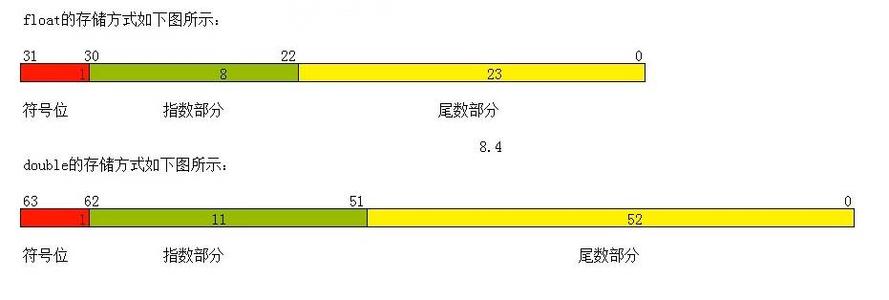
* 为了实现数据完整性，需要检验数据库表中的每行和每列数据是否符合要求
* 在创建表的时候，应该保证以后的数据输入是正确的，错误的数据不允许输入

### **6.1 域完整性**

不同的字段需要设置为各种合适的类型，比如年龄就是整数类型







### **6.2 默认值**

默认值是指如果用户没有指定值的情况下会记录的此字段指定一个提供一个预先设定的值

可以把居住地默认值设置为北京

### **6.3 非空约束**

我们可以指定某个字段不能不输入，必须提供一个非空的值

姓名字段不能为空

## **7.实体完整性**

### **7.1 主键约束**

* 表中一列或者几列组合的值能用来唯一标识表中的每一行，这样的列或者列组合称为表的主键，主键表的数据不同重复。
* 如果两列或者多列组合起来唯一标识表中的每一行，则该主键又称为"组合键"

****主键的选择标准****

1. 最少性 尽量选择单个键作为主键
2. 稳定性 ，由于主键是用来在两个表间建立联接的，所以不能经常更新，最好就不更新

### **7.2 外键**

成绩表中的学生ID应该在学生表中是存在的 我们应该让成绩表中的ID只能引用学生表中的ID，它们的值应该是一一对应的，也就是说成绩表中的ID是成绩表中的外键，对应学生表的主键 ，这样就可以保证数据的引用完整性

### **7.3 唯一约束**

唯一约束是指某个字段值是唯一的，在所有的记录中不能有重复的值.

学生的身份证号可以设置为唯一约束

### **7.4 标识列**

* 当表中没有合适的列作为主键时可以考虑增加标识列，标识列是一个无实际业务含义的列，仅仅用来区分每条记录。
* 标识列的值是自动生成的，不能在该列上输入数据

### **7.5 外键约束**

一个表的外键必须引用另一个表的主键,比如成绩表中的学生ID会引用学生表的主键，课程ID会引用成绩表的主键

* 主表没有记录，子表中不能添加相应的记录
* 修改和删除主表记录不能让子表记录孤立，必须相应修改和删除

## **8. 数据操作**

### **8.1 创建学生表**

**CREATE** **TABLE** `student` (`id` int(11) **NOT** NULL PRIMARY **KEY** AUTO\_INCREMENT ,`name` varchar(50) **NOT** NULL ,`age` int(11) NULL **DEFAULT** NULL ,`city` varchar(50) **DEFAULT** '北京' ,

)

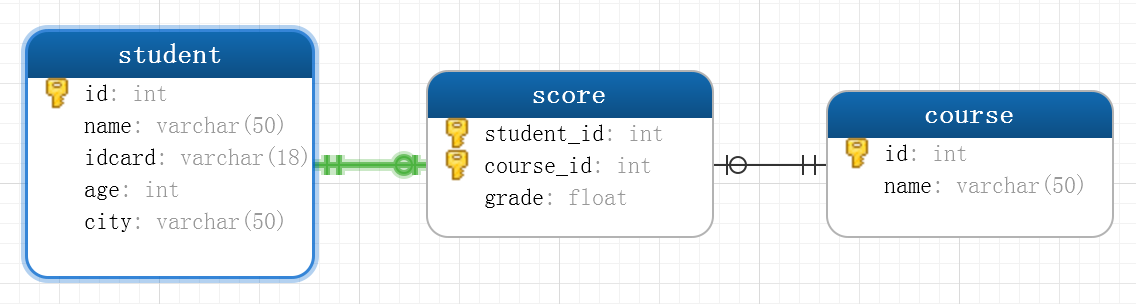
### **8.2 增加身份证号字段**

**ALTER** **TABLE** `student` **ADD** **COLUMN** `idcard` varchar(15) NULL **AFTER** `city`; --增加身份证字段**ALTER** **TABLE** `student` **MODIFY** **COLUMN** `idcard` varchar(18) **DEFAULT** NULL **AFTER** `name`; --修改身份证字段**ALTER** **TABLE** `student`**DROP** **COLUMN** `idcard`; --删除身份证字段

### **8.3 添加约束**

-- 主键约束**ALTER** **TABLE** `student` **ADD** PRIMARY **KEY** (`id`);-- 唯一约束**ALTER** **TABLE** `student` **ADD** **UNIQUE** **INDEX** `uq\_idcard` (`idcard`) ;-- 默认约束**ALTER** **TABLE** `student` **MODIFY** **COLUMN** `city` varchar(50) **DEFAULT** '北京' **AFTER** `age`;-- 外键约束**ALTER** **TABLE** `score` **ADD** **CONSTRAINT** `fk\_stuid` **FOREIGN** **KEY** (`student\_id`) **REFERENCES** `student` (`id`);-- 删除约束**ALTER** **TABLE** `score` **DROP** **FOREIGN** **KEY** `fk\_stuid`;

### **8.4 准备数据**



**CREATE** **TABLE** `student` (

`id` int(11) **NOT** NULL AUTO\_INCREMENT,

`name` varchar(50) **NOT** NULL,

`idcard` varchar(18) **DEFAULT** NULL,

`age` int(11) **DEFAULT** NULL,

`city` varchar(50) **DEFAULT** '',

PRIMARY **KEY** (`id`)

);

**CREATE** **TABLE** `course` (

`id` int(11) **NOT** NULL AUTO\_INCREMENT,

`name` varchar(50) **DEFAULT** NULL,

PRIMARY **KEY** (`id`)

);

**CREATE** **TABLE** `score` (

`student\_id` int(11) **NOT** NULL **DEFAULT** '0',

`course\_id` int(11) **NOT** NULL **DEFAULT** '0',

`grade` float **DEFAULT** NULL,

PRIMARY **KEY** (`student\_id`,`course\_id`),

**KEY** `fk\_courseid` (`course\_id`),

**CONSTRAINT** `fk\_courseid` **FOREIGN** **KEY** (`course\_id`) **REFERENCES** `course` (`id`),

**CONSTRAINT** `fk\_stuid` **FOREIGN** **KEY** (`student\_id`) **REFERENCES** `student` (`id`)

) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=utf8;

## **9. SQL**

### **9.1 SQL是什么**

Structured Query Language：结构化查询语言

### **9.2 为什么要用SQL**

1. 使用界面操作数据库不方便
2. 我们需要通过应用程序去操作数据库

### **9.3 SQL组成**

#### **9.3.1 DDL（data definition language）是数据定义语言**

主要的命令有CREATE、ALTER、DROP等，****DDL****主要是用在定义或改变表（TABLE）的结构，数据类型，表之间的链接和约束等初始化工作上，他们大多在建立表时使用

#### **9.3.2 DML（data manipulation language）是数据操纵语言**

它们是SELECT、UPDATE、INSERT、DELETE`，就象它的名字一样，这4条命令是用来对数据库里的数据进行操作的语言

#### **9.3.3 DCL（DataControlLanguage）是数据库控制语言**

是用来设置或更改数据库用户或角色权限的语句，包括（grant,revoke等）语句

### **9.4 SQL运算符**

是一种符号，它是用来进行列间或者变量之间的比较和数学运算的

### **9.4.1 算术运算符**

| **运算符** | **说明** |
| --- | --- |
| + | 加运算，求两个数或表达式相加的和，如1+1 |
| - | 减少减运算，求两个数或表达式相减的差，如4-1 |
| \* | 乘运算，求两个数或表达式相乘的积，如2\*2 |
| / | 除运算，求两个数或表达式相除的商，如6/4的值为1 |
| % | 取模运算，求两个数或表达式相除的余数，如：6%4的值为2 |

* 查询姓名全称

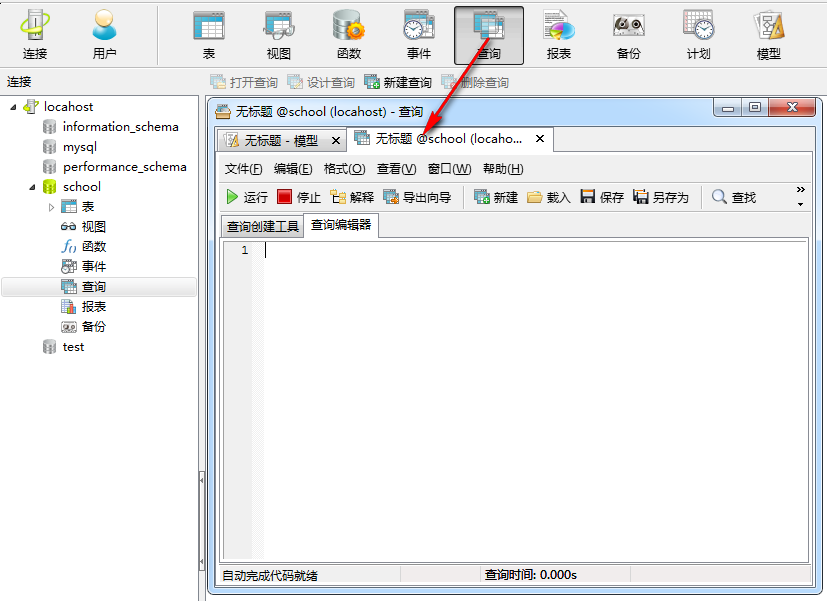
### **9.4.2 逻辑运算符**

| **运算符** | **说明** |
| --- | --- |
| AND | 当且仅当两个布尔表达式都为true时，返回TRUE |
| OR | 当且仅当两个布尔表达式都为false，返回FALSE |
| NOT | 布尔表达式的值取反 |

### **9.4.3 比较运算符**

| **运算符** | **说明** |
| --- | --- |
| = | 等于 |
| > | 大于 |
| < | 小于 |
| <> | 不等于 |
| >= | 大于等于 |
| <= | 小于等于 |
| != | 不等于 |

## **9.5 查询窗口**



## **9.6 数据操作语言**

### **9.6.1 插入数据行**

#### **9.6.1.1 语法**

**INSERT** [**INTO**] 表名 [(列名)] **VALUES** (值列表)

#### **9.6.1.2 插入记录**

向学生表插入一条记录, 姓名张三,身份证号123456,年龄30,城市北京

**INSERT** **INTO** [school].[student]

(**name**,idcard,age,city)**VALUES**

('张三','123456',30,'北京')

#### **9.6.1.3 注意事项**

* 每次插入一行数据，不能只插入一部分数据，插入的数据是否有效将按照整行的完整性要求来检验
* 每个数据值的数据类型、精度、位数必须与要应的列名精确匹配
* 不能为标识符指定值
* 如果某字段设置为不能为空，则必须插入数据
* 插入数据时还要符合检查性约束的要求
* 有缺省值的列，可以使用DEFAULT关键字来代替插入实际的值

### **9.6.2 更新数据行**

#### **9.6.2.1 语法**

**UPDATE** 表名 **SET** 列名 = 更新值 [**WHERE** 更新条件]

#### **9.6.2.2 更新记录**

* 更新ID等于7的学生年龄改为40，城市改为上海

**UPDATE** [school].[student]

**SET** age = 40,city = '上海'**WHERE** **id**=7

#### **9.6.2.3 注意**

* 多列时用逗号隔开，一定要加更新条件以免错误更新
* 多个联合条件使用 AND id=7 and idcard='410787'
* 判断某字段是否为空 email is null or email = ''

### **9.6.3 删除数据**

#### **9.6.3.1 语法**

**DELETE** [**FROM**] 表名 [**WHERE** <删除条件>]

#### **9.6.3.2 删除记录**

* 删除ID=7的学生记录

**DELETE** **FROM** [school].[student] **WHERE** **id**=7

#### **9.6.3.3 注意**

* Delete语句是对整行进行操作，因此不需要提供列名
* 如果要删除主表数据，则要先删除子表记录

### **9.6.4 TRUNCATE 截断表**

截断整个表中的数据

#### **9.6.4.1 语法**

**TRUNCATE** **TABLE** 表名

#### **9.6.4.2 截断学生表**

**TRUNCATE** **TABLE** student

数据全部清空,但表结构、列、约束等不被改动 不能用于有外键约束引用的表 标识列重新开始编号 因为要删除的数据不会写入日志，数据也不能恢复，所以工作中请尽量不要使用此命令

## **9.7 数据查询**

* 查询就是从客户端 发出查询请求数据库服务器，并从数据库返回查询结果的过程
* 每次执行查询只是从数据表中提取数据，并按表的方式呈现出来
* 查询产生的是虚拟表,并不会保存起来

### **9.7.1 查询**

#### **9.7.1.1 语法**

**SELECT** <列名> **FROM** <表名>

[**WHERE** <查询条件表达式>]

[**ORDER** **BY** <排序的列名>[**ASC**或**DESC**]]

#### **9.7.1.2 排序**

* 查询北京的学生信息，并按ID正序排列

**SELECT** **id**,**name**,idcard,age,city**FROM** student**WHERE** home= '北京'**ORDER** **BY** **id** **asc**

* 

#### **9.7.1.3 别名**

**SELECT** **id**,**name**,idcard,age,city **AS** home**FROM** student**WHERE** city= '山东'**ORDER** **BY** **id** **asc**

#### **9.7.1.4 查询空行**

**SELECT** **id**,**name**,age,city**FROM** student**WHERE** city **is** null **or** city =''

#### **9.7.1.5 常量列**

**SELECT** **id**,**name**,age,city,'中国' **as** country**FROM** student

#### **9.7.1.6 限制返回的行数**

**SELECT** **id**,**name**,age,city,'中国' **as** country**FROM** student **limit** 2

#### **9.7.1.7 DISTINCT**

* 查询同学们一共来自哪些不同的城市

**SELECT** **id**,**name**,age,**DISTINCT** city,'中国' **as** country**FROM** student

#### **9.7.1.8 +**

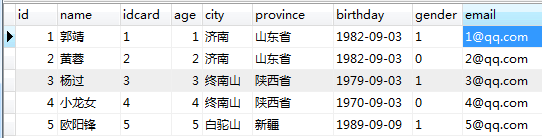
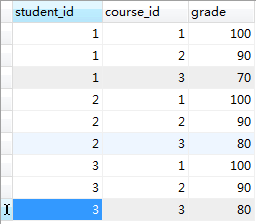
* 在MYSQL中+号只能用作于数字

**SELECT** 1+1**SELECT** 1+'1'**SELECT** 1+'zfpx' **SELECT** 1+null**SELECT** **CONCAT**(last\_name,first\_name) **FROM** **user**;

### **9.7.1.7 练习**

#### **9.7.1.7.1 准备数据**

**ALTER** **TABLE** `student`**ADD** **COLUMN** `province` varchar(50) NULL **AFTER** `city`,**ADD** **COLUMN** `birthday` date NULL **AFTER** `province`,**ADD** **COLUMN** `gender` int(11) NULL **AFTER** `birthday`,**ADD** **COLUMN** `email` varchar(50) NULL **AFTER** `gender`;

**INSERT** **INTO** `student`(**id**,**name**,gender,age,city,province,birthday,idcard,email) **VALUES** ('1', '郭靖', '1', '1', '济南', '山东省', '1982-09-03', 1, '1@qq.com');**INSERT** **INTO** `student`(**id**,**name**,gender,age,city,province,birthday,idcard,email) **VALUES** ('2', '黄蓉', '2', '2', '济南', '山东省', '1982-09-03', 0, '2@qq.com');**INSERT** **INTO** `student`(**id**,**name**,gender,age,city,province,birthday,idcard,email) **VALUES** ('3', '杨过', '3', '3', '终南山', '陕西省', '1979-09-03', 1, '3@qq.com');**INSERT** **INTO** `student`(**id**,**name**,gender,age,city,province,birthday,idcard,email) **VALUES** ('4', '小龙女', '4', '4', '终南山', '陕西省', '1970-09-03', 0, '4@qq.com');**INSERT** **INTO** `student`(**id**,**name**,gender,age,city,province,birthday,idcard,email) **VALUES** ('5', '欧阳锋', '5', '5', '白驼山', '新疆', '1989-09-09', 1, '5@qq.com');

**INSERT** **INTO** `course` **VALUES** ('1', '语文');**INSERT** **INTO** `course` **VALUES** ('2', '数学');**INSERT** **INTO** `course` **VALUES** ('3', '英语');

**INSERT** **INTO** `score` **VALUES** ('1', '1', '100');**INSERT** **INTO** `score` **VALUES** ('1', '2', '90');**INSERT** **INTO** `score` **VALUES** ('1', '3', '70');**INSERT** **INTO** `score` **VALUES** ('2', '1', '100');**INSERT** **INTO** `score` **VALUES** ('2', '2', '90');**INSERT** **INTO** `score` **VALUES** ('2', '3', '80');**INSERT** **INTO** `score` **VALUES** ('3', '1', '100');**INSERT** **INTO** `score` **VALUES** ('3', '2', '90');**INSERT** **INTO** `score` **VALUES** ('3', '3', '80');

#### **9.7.1.7.2 查询山东省的同学全部信息**

**select** \* **from** student **where** province = '山东';

#### **9.7.1.7.3 查询山东省的男同学的全部信息**

**select** \* **from** student **where** province = '山东' **and** gender=1;

#### **9.7.1.7.4 查询没有填写电子邮箱的同学的姓名和身份证号**

**SELECT** **name**,idcard **FROM** student **WHERE** email **IS** NULL

### **9.7.1.8 排序**

#### **9.7.1.8.1 把学生的分数按升序排列**

**select** \* **from** score **order** **by** grade **asc**;

#### **9.7.1.8.2 把学生的分数按降序排列**

**select** \* **from** score **order** **by** grade **desc**;

#### **9.7.1.8.3 课程编号按升序，成绩按降序**

**select** \* **from** score **order** **by** course\_id **asc**, grade **desc**;

## **9.8 函数**

### **9.8.1 字符函数**

| **函数名称** | **描述** |
| --- | --- |
| CONCAT | 字符串连接 |
| CONCAT\_WS | 使用指定的分隔符进行字符连接 |
| FORMAT | 数字格式化 |
| LOWER | 转小写字母 |
| UPPER | 转大写字母 |
| LEFT | 返回字符串s开始的最左边n个字符 |
| RIGHT | 返回字符串s开始的最左边n个字符 |

#### **9.8.1.1 LENGTH**

**SELECT** **LENGTH**('zfpx');**SELECT** **LENGTH**('珠峰培训');**SHOW** **VARIABLES** **like** '%char%';

#### **9.8.1.2 拼接字符串 [CONCAT]**

**SELECT** **CONCAT**(last\_name,'\_',first\_name) **FROM** employees;

#### **9.8.1.3 拼接字符串 [UPPER]**

**SELECT** **UPPER**('zfpx') **FROM** employees;

#### **9.8.1.4 拼接字符串 [LOWER]**

**SELECT** **LOWER**('ZFPX') **FROM** employees;

#### **9.8.1.5 截取字符串 [SUBSTR]**

**SELECT** **SUBSTR**('zfpx',2);-- 截取从指定索引处开始的所有的字符**SELECT** **SUBSTR**('zfpx',2,3);--截取从指定索引处开始的指定数量的字符

#### **9.8.1.6 姓名的首字符大写，其它字符小写然后用\_拼接**

**SELECT** **CONCAT**(**UPPER**(**SUBSTR**(last\_name,1,1)),'\_',**LOWER**(**SUBSTR**(last\_name,2)));

#### **9.8.1.7 返回子串在原始字符串的起始索引**

**SELECT** **INSTR**('zfpx','f');

#### **9.8.1.8 去掉左右空格[TRIM]**

**SELECT** **TRIM**(' zfpx ');**SELECT** **LTRIM**(' zfpx ');**SELECT** **RTRIM**(' zfpx ');**SELECT** **TRIM**('x' **FROM** 'xxzfpxxx');

#### **9.8.1.9 用空格补齐[LPAD]**

**SELECT** **LPAD**('zfpx',10,'@');**SELECT** **LPAD**('1',8,'0');**SELECT** RPAD('1',8,'0');

#### **9.8.1.10 替换**

**SELECT** **REPLACE**('zfpx','f','q')

#### **9.8.1.11 FORMAT**

**SELECT** **FORMAT**(100000,2); 100,000.00**select** **format**(100.31111,2) 100.31**select** **format**(100.31111,0); 100**SELECT** **FORMAT**(423423234.65534453,2); 423,423,234.66

#### **9.8.1.12 LEFT RIGHT**

**select** **left**('abcde12345',5); //abcde**select** **left**('abcde12345',5); //12345

### **9.8.2 数学函数**

| **函数名称** | **描述** |
| --- | --- |
| CEIL | 向上取整 |
| FLOOR | 向下取整数 |
| DIV | 整数取 |
| MOD | 取余(取模) |
| POWER | 幂运算 |
| ROUND | 四舍五入 |
| TRUNCATE | 数字截取 |

#### **9.8.2.1 四舍五入**

**SELECT** **ROUND**(2.5);**SELECT** **ROUND**(2.555,2);

#### **9.8.2.2 向上取整**

**SELECT** **CEIL**(1.00);

#### **9.8.2.3 向下取整**

**SELECT** **FLOOR**(1.00);

#### **9.8.2.4 截断**

**SELECT** **TRUNCATE**(1.66,1);

#### **9.8.2.5 取余**

**SELECT** **mod**(10,3);

### **9.8.3 日期函数**

| **函数名称** | **描述** |
| --- | --- |
| NOW | 当前日期和时间 |
| CURDATE | 当前日期 |
| CURTIME | 当前时间 |
| DATE\_ADD | 日期变化 |
| DATEDIFF | 计算日期差 |
| DATE\_FORMAT | 日期格式化 |

#### **9.8.3.1 返回日期时间**

**SELECT** **NOW**():

#### **9.8.3.2 返回日期**

**SELECT** **CURDATE**():

#### **9.8.3.3 返回日期**

**SELECT** **CURTIME**():

#### **9.8.3.4 返回日期中指定的部分**

年 月 日 小时 分钟 秒

**SELECT** **YEAR**(**NOW**());**SELECT** **MONTH**(**NOW**());**SELECT** MONTHNAME(**NOW**());**SELECT** **DAY**(**NOW**());**SELECT** **HOUR**(**NOW**());**SELECT** **MINUTE**(**NOW**());**SELECT** **SECOND**(**NOW**());

#### **9.8.3.5 str\_to\_date**

将日期格式的字符串转成指定格式的日期

**SELECT** **STR\_TO\_DATE**('2018-09-09','%Y-%m-%d')

| **序号** | **格式符** | **功能** |
| --- | --- | --- |
| 1 | %Y | 4位的年份 |
| 2 | %y | 2位的年份 |
| 3 | %m | 月份(01,02) |
| 4 | %c | 月份(1,2) |
| 5 | %d | 日(01,02) |
| 6 | %H | 小时(24小时制) |
| 7 | %h | 小时(12小时制) |
| 8 | %i | 分钟(00,01) |
| 9 | %s | 秒(00,01) |

#### **9.8.3.6**

查询入职日期为1984-8-4的员工信息

**SELECT** \* **FROM** employees **WHERE** hiredate = **STR\_TO\_DATE**('12-30 1984','%m-%d %Y');

#### **9.8.3.7 DATE\_FORMAT**

将日期转换成指定字符串

**SELECT** **DATE\_FORMAT**(**NOW**(),'%Y年%m月%d日')

#### **9.8.3.8 DATE\_ADD**

**SELECT** **DATE\_ADD**(**NOW**(),INTERVAL 365 **DAY**);**SELECT** **DATE\_ADD**(**NOW**(),INTERVAL 1 **MONTH**);**SELECT** **DATE\_ADD**(**NOW**(),INTERVAL 1 **YEAR**);

#### **9.8.3.9 DATEDIFF**

**SELECT** **DATEDIFF**('2019-1-1',**NOW**());

### **9.8.4 其它函数**

**SELECT** CONNECTION\_ID();**SELECT** **DATABASE**();**SELECT** **VERSION**();**select** **LAST\_INSERT\_ID**();**SELECT** **USER**();

**SELECT** **MD5**('123456');//摘要算法**SELECT** **PASSWORD**('123456');//修改当前用户的密码**SELECT** **User**,**Password** **from** mysql.user;

### **9.8.5 流程控制函数**

#### **9.8.5.1 IF**

**SELECT** **IF**(1>0,'A','B');

#### **9.8.5.2 CASE函数**

CASE 要判断的字段和表达式

WHEN 常量1 then 要显示的值或语句1

WHEN 常量2 then 要显示的值或语句2

ELSE 要显示的值**END**

**SELECT** **CASE** **WHEN** grade<60 **then** '不及格'**WHEN** grade>=60 **then** '及格'**ELSE** '未知'**ENDFROM** score;

**SELECT** **CASE** **levelWHEN** 'A' **then** '优秀'**WHEN** 'B' **then** '良好'**ELSE** '未知'**ENDFROM** score;

### **9.9 自定义函数**

* 自定义函数是对MySQL的扩展，使用方式和内置函数相同
* 函数必须要有参数和返回值
* 函数可以接收任意类型的值，也可以接收这些类型的参数
* 函数体由合法的SQL语句组成
* 函数体可以是简单的SELECT语句或INSERT语句，如果是复合结构要用BEGIN...END
* 函数体也可以包含声明，循环和流程控制
* 返回值只能有一个

#### **9.9.1 语法**

**CREATE** **FUNCTION** func\_name() **RETURNS** VARCHAR(64)**body**

#### **9.9.2 znow**

* 注意函数名一定要跟着小括号表示参数

**CREATE** **FUNCTION** znow() **RETURNS** VARCHAR(30)**RETURN** **DATE\_FORMAT**(**NOW**(),'%Y年%m月%d日 %H点:%i分%s秒');

#### **9.9.3 zadd**

**CREATE** **FUNCTION** zadd(num1 INT,num2 INT) **RETURNS** INT**RETURN** num1+num2;

#### **9.9.4 addUser**

**CREATE** **TABLE** stu(**id** int PRIMARY **KEY** AUTO\_INCREMENT,**name** VARCHAR(50));**CREATE** **FUNCTION** addUser(**name** VARCHAR(50)) **RETURNS** INT**BEGIN**

**INSERT** **INTO** stu(**name**) **VALUES**(**name**);

RETURN LAST\_INSERT\_ID();**END**

**SELECT** addUser('zfpx');**DROP** **FUNCTION** addUser

### **9.10 模糊查询**

就是查询的条件是模糊的，不是特别明确的

#### **9.10.1 通配符**

代替一个或多个真正的字符，与LIKE 关键字一起使用

* \_表示一个任意字符
* %表示任意长度的字符串

#### **9.10.2 BETWEEN AND**

查询某一列在指定的规范内的记录，包括两个边界

**select** \* **from** score **where** grade **between** 80 **and** 100

#### **9.10.3 IN**

查询某一列中的值在列出的内容列表中

**select** \* **from** student **where** city **in** ('北京','上海','广东')

#### **9.10.4 IS NUL**

* 查询没有邮箱的 IS NUL
* 查询有邮箱的 IS NOT NULL

## **10. DCL**

### **10.1 创建用户**

CREATE USER 用户名@访问地址 IDENTIFIED BY '密码'

CREATE USER user1@localhost IDENTIFIED BY '123456'

### **10.2 授予权限**

GRANT 权限1,权限2,,, ON 数据库.\* TO 用户名

GRANT CREATE,DROP ON db1.\* TO user1@localhost;

### **10.3 授予权限**

REVOKE 权限1，权限2，，，ON 数据库.\* FROM 用户名;

REVOKE CREATE,DROP ONT db1.\* FROM user1@localhost;

### **10.4 查看用户权限**

SHOW GRANTS FROM 用户名

SHOW GRANTS FROM user1@localhost;

### **10.5 删除用户**

DROP USER 用户名

DROP USER user1@localhost;

### **10.6 修改密码**

UPDATE USER SET PASSWORD=PASSWORD('password') WHERE User='username' and Host='localhost';

FLUSH PRIVILEGES;

## **11. 存储过程**

* 大多数SQL语句都是针对一个或多个表的单条语句。但是并不是所有的操作都是可以用一条语句来完成的，经常有一些操作是需要多条语句配合才能完成。我们引入的存储过程（Stored Procedure）是一组为了完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给定参数（如果该存储过程带有参数）来调用执行它

### **11.1 语法**

CREATE PROCEDURE([[IN |OUT |INOUT ] 参数名 数据类形...])

### **11.2 调用**

* 因为存储过程实际上是一种函数,所以存储过程名后需要有 () 符号(即使不传递参数也需要)

CALL Avg\_Price()

### **11.3 删除**

DROP PROCEDURE IF EXISTS Avg\_Price;

### **11.4 参数**

* 存储过程并不显示结果，而是把结果返回给你指定的变量

create procedure sum (**in** a int,**in** b int,out result int)

begin

select a+b INTO result;

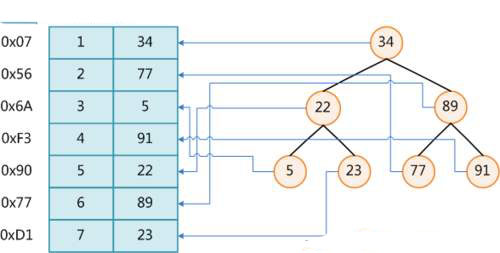
end

call sum5(1,2,@result);

select @result;

## **12. 索引**

* 索引可以提高数据库的查询速度



### **12.1 准备数据**

-- 构建一个8388608条记录的数据

-- 创建用户表

create table user(

id int,

username varchar(64),

userno int

)

-- 创建生成随机字符串的函数

CREATE FUNCTION `rand\_string`(n INT) RETURNS varchar(255)

BEGIN

DECLARE chars\_str varchar(100) DEFAULT 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';

DECLARE return\_str varchar(255) DEFAULT '';

DECLARE i INT DEFAULT 0;

WHILE i < n DO

SET return\_str = concat(return\_str,substring(chars\_str , FLOOR(1 + RAND()\*62 ),1));

SET i = i +1;

END WHILE;

RETURN return\_str;

END;

-- 产生随机数字

create **function** **rand\_num**() **returns** **int**(5) **begin** **declare** **i** **int** **default** 0; **set** **i** = **floor**(10+rand()\*500); **return** **i**; **end**

-- 创建存储过程，向用户表添加海量数据

**create** **procedure** **insert\_user**(in start int(10),**in** **max\_num** **int**(10))

**begin**

**declare** **i** **int**;

**set** **i**=0;

**while** **i**<**max\_num** **do**

**set** **i**=**i**+1;

**insert** **into** **user**(id,username,userno) **values**((start+i),**rand\_string**(6),**rand\_num**());

**end** **while**;

**end**;**call** **insert\_user**(1,8388608);

### **12.2 创建普通索引**

create table user(id int primary key,name varchar(32),email varchar(32));

alter table user add index(name);

create index idx\_name on user(name);

### **12.3 查看索引**

show index **from** 表名;

show index **from** user

### **12.4 删除索引**

alter table 表名 drop primary;

alter table 表名 drop index 索引名;

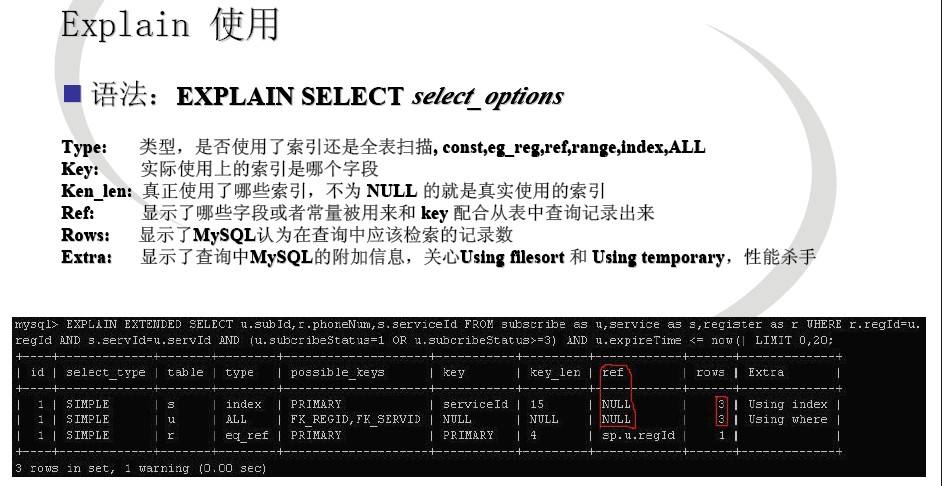
drop index 索引名 on 表名;

### **12.5 慢查询**

explain

select \* **from** users where userno=4593;

alter table users add index(userno);



### **12.6 索引创建的原则**

* 比较频繁作为查询条件的字段应该创建索引
* 唯一性太差的字段不适合单独创建索引，即使频繁作为查询条件
* 更新非常频繁的字段不适合作创建索引
* 不会出现在where子句中的字段不该创建索引