# 尚马教育 JAVA 基础课程

# MySQL数据库

文档编号：A18

创建日期： 2017-04-12

最后修改日期：2019-09-23

版 本 号：V3.0

电子版文件名：尚马教育-第一阶段-18.MySQL数据库专题课程.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-01 | 王绍成 | Java基础版本更新 | V2.0 |
| 2019-08-09 | 徐丽莎 | Java基础版本更新 | V3.0 |

**主讲人：**

**徐丽莎**

目录

[尚马教育 JAVA 基础课程 1](#_Toc11217)

[MySQL数据库 1](#_Toc2186)

[1. 数据库概述(Database) 3](#_Toc5380)

[1.1. 数据库分类 3](#_Toc25493)

[1.2. 常用关系型数据库 3](#_Toc18458)

[2. MySQL数据库 4](#_Toc25939)

[2.1. 优势 4](#_Toc15104)

[2.2. MySQL安装配置 4](#_Toc18653)

[3. SQL概述 5](#_Toc28104)

[3.1. SQL是什么 5](#_Toc25076)

[3.2. SQL语法 5](#_Toc12632)

[3.3. SQL分类 5](#_Toc357)

[4. DDL 6](#_Toc20674)

[4.1. 相关的指令 6](#_Toc30091)

[4.2. 基本操作(库) 6](#_Toc6724)

[4.3. 数据类型 7](#_Toc20432)

[4.3.1. 整型 7](#_Toc27955)

[4.3.2. 小数 7](#_Toc8038)

[4.3.3. 字符串 8](#_Toc23654)

[4.3.4. 日期类型 8](#_Toc4748)

[4.4. 基本操作(表) 8](#_Toc12509)

[5. DML 9](#_Toc15118)

[6. 约束 10](#_Toc14007)

[6.1. 非空约束(not null) 10](#_Toc22521)

[6.2. 唯一性约束(unique) 11](#_Toc20145)

[6.3. 默认约束(default) 11](#_Toc21893)

[6.4. 主键约束(primary key) 11](#_Toc32205)

[6.5. 外键约束(foreign key) 12](#_Toc28828)

[7. 数据库引擎 13](#_Toc26573)

[8. DQL 14](#_Toc6928)

[8.1. 语法 14](#_Toc24503)

[8.2. 准备表 14](#_Toc15267)

[8.3. 基础查询 17](#_Toc6825)

[8.4. 条件查询 18](#_Toc5091)

[8.5. 模糊查询(like) 18](#_Toc22850)

[8.6. 字段控制查询 19](#_Toc23090)

[8.6.1. 去除重复的记录 19](#_Toc30632)

[8.6.2. Null值运算 19](#_Toc17447)

[8.6.3. 别名查询 19](#_Toc7933)

[8.7. 排序(order by) 19](#_Toc177)

[8.8. 分组(聚合)函数 20](#_Toc11102)

[8.9. 分组查询(group by) 20](#_Toc2213)

[8.10. Having子句 21](#_Toc13017)

[8.11. 关联查询(关联的是多种张表) 21](#_Toc8281)

[8.11.1. 等值连接：连接的条件是一个等号 21](#_Toc15605)

[8.11.2. 不等值连接：连接条件不是等号 21](#_Toc31610)

[8.11.3. 外连接 21](#_Toc30674)

[8.11.4. 自连接 22](#_Toc15430)

[8.11.5. 子查询 22](#_Toc24293)

[8.11.6. 集合运算 22](#_Toc4551)

[8.12. 分页查询(limit) 22](#_Toc10449)

[9. 常用函数 22](#_Toc5325)

[9.1. 字符串函数 22](#_Toc15615)

[9.2. 数值函数 23](#_Toc16811)

[9.3. 日期和时间函数 24](#_Toc14033)

## 数据库概述(Database)

### 数据库分类

* 生活中，也有很多库，比如车库，存储各种各样品牌的车。
* 同理，数据库就是存储各式各样的数据。(是长期存储在计算机内有组织、有共享、统一管理的数据集合)
* 从存储位置：
  + 基于磁盘的数据库 mysql sqlserver oracle。数据存储文件中。IO 读写 效率偏慢 数据持久(永久保存)
  + 基于内存的数据库 redis 效率比较快 数据有可能丢失(不影响)
* 从关系上：
  + 关系型数据库 mysql sqlserver oracle 表与表 字段与字段之间 数据与数据之间有关系。
  + 非关系型数据库 redis

### 常用关系型数据库

* MySQL(学习)
* Oracle
* SQLServer
* 数据库表由横向的行和纵向的列组成：
  + 行称为记录，是组织数据的单位；
  + 列称为字段，每一列表示记录的一个属性；
* 数据表每条记录的列属性都具有相应的描述信息，如数据类型和数据宽度等。

|  |
| --- |
|  |

## MySQL数据库

### 优势

* 速度：运行速度快
* 价格：社区版免费
* 容易使用：相对来说复杂程度低，易于学习
* 可移植性：能工作在不同系统平台上
* 接口丰富：提供多种语言的API接口
* 支持查询语言：可以利用标准sql语法和ODBC(开放式数据库连接)的应用程序
* 安全性和连接性：
* 安全灵活的权限和密码系统，允许主机验证，服务器的密码连接加密；
* 网络化提高数据共享的效率

### MySQL安装配置

* 安装MySQL服务端应用程序：目前最新版本：8.\* (学习使用5.7)
* 下载路径：<https://dev.mysql.com/downloads/windows/installer/5.7.html>
* 在本地安装服务端程序，进行相关配置(用户名都是：root)
* 安装成功之后，就可以执行SQL语句的编程。
* Mysql的服务里面包含：
  + 很多数据库-->很多表(table)-->表字段，表字段数据类型，约束，数据

## SQL概述

### SQL是什么

* SQL(Structured Query Language)是“结构化查询语言”，它是对关系型数据库的操作语言。它可以应用到所有关系型数据库中。SQL标准（ANSI/ISO）有：
  + SQL-92：1992年发布的SQL语言标准；
  + SQL:1999：1999年发布的SQL语言标签；
  + SQL:2003：2003年发布的SQL语言标签；
* 这些标准就与JDK的版本一样，在新的版本中总要有一些语法的变化。不同时期的数据库对不同标准做了实现。
* 虽然SQL可以用在所有关系型数据库中，但很多数据库还都有标准之后的一些语法，我们可以称之为“方言”。例如MySQL中的LIMIT语句就是MySQL独有的方言，其它数据库都不支持！当然，Oracle或SQL Server都有自己的方言。

### SQL语法

* SQL语句可以单行或多行书写，以分号结尾；
* 可以用空格和缩进来来增强语句的可读性；
* 关键字不区别大小写，建议使用大写；

### SQL分类

* DDL(Data Definition Language)：数据定义语言，用来定义数据库对象：库、表、列等；
* DML(Data Manipulation Language)：数据操作语言，用来定义数据库记录（数据）；
* DCL(Data Control Language)：数据控制语言，用来定义访问权限和安全级别；
* DQL(Data Query Language)：数据查询语言，用来查询记录 select

## DDL

### 相关的指令

* 连接指定计算机上mysql的server:
  + mysql -uroot [-hip] -p
  + mysql的server目前不支持连接自动分配ip。localhost/127.0.0.1
  + mysql -uroot -h127.0.0.1 -p

|  |
| --- |
| 1. 查看mysql的server里面有哪些数据库 show databases；  2. 选择使用特定的数据库 use 数据库的名称;  3. 查看指定数据库下所有的表： show tables；  4. 查看表里面所有的数据: select \* from 表名;  5. 查看表的表结构: desc 表名; |

### 基本操作(库)

|  |
| --- |
| 1. 创建数据库:  create database [if not exists]数据库名; (数据库名称不可更改)  2. 删除数据库  drop database 数据库名；(连着表/数据都一块删除，数据无法回滚)   1. 查看创建数据库基本信息   show create database 数据库名;  4.创建表  create table 表名(  字段1 数据类型 [约束],  字段2 数据类型 [约束],  ...  字段n 数据类型 [约束]  ); |

### 数据类型

* MySQL与Java一样，也有数据类型。MySQL中数据类型主要应用在列(字段)上，
* 常用的数据类型：

#### 整型

|  |
| --- |
|  |
| m表示SELECT查询结果集中的显示宽度，与取值范围无关。 m最大值11  如果加入无符号unsigned属性: age tinyint(2) unsigned tinyint(1) 0 1 java 对应的是boolean  tinyint 无符号的范围是0到255. smallint 0-65535 mediumint 0-16777215 int 0-4294967295 bigint 0-18446744073709551615  如果加入zerodfill属性: 00100 m=5 00010 |

#### 小数

|  |
| --- |
| Decimal(m,d) 定点数 参数m<65 是总个数，d<30且 d<m 是小数位。 BigDecimal |
| 设一个字段定义为float(6,3)，如果插入一个数123.45678,实际数据库里存的是123.457，但总个数还以实际为准，即6位。整数部分最大是3位，如果插入数12.123456，存储的是12.1234，如果插入12.12，存储的是12.1200.  浮点数有误差，对货币等对精度敏感的数据，应该用定点数表示或存储 |

#### 字符串

|  |
| --- |
|  |
| char和varchar区别：  1.char(n) 若存入字符数小于n，则以空格补于其后，查询之时再将空格去掉。所以char类型存储的字符串末尾不能有空格，varchar不限于此。  2.char(n) 固定长度，char(4)不管是存入几个字符，都将占用4个字节，varchar是存入的实际字符数+1个字节（n<=255）或2个字节(n>255)，所以varchar(4),存入3个字符将占用4个字节。  3.char类型的字符串select检索速度要比varchar类型的快很多。 |

#### 日期类型

|  |
| --- |
| Year(2/4)两位或四位格式的年。默认是四位格式yyyy 2019 yy 19 |
| 一个字段为timestamp，这个字段里的时间数据会随其他字段修改的时候自动刷新，所以这个数据类型的字段可以存放这条记录最后被修改的时间. |
| 尤其是在大数据量测试环境下，指定数据类型的时候一般是采用从小原则，比如能用TINY INT的最好就不用INT，能用FLOAT类型的就不用DOUBLE类型，这样会对MYSQL在运行效率上提高很大， |

### 基本操作(表)

|  |
| --- |
| 1. 创建学生信息表   create table tb\_student(  sid int(4),  sname varchar(20),  gender char(1),  age tinyint(2) unsigned,  score float(4,1),  birthday date,  createtime datetime,  updatetime datetime  );   1. 表结构 alter(了解)   5.1 新增新的字段  alter table tb\_student add age tinyint(2) unsigned;  5.2 删除字段  alter table tb\_student drop age;  5.3 修改字段名称  alter table tb\_student change age sage tinyint(2) unsigned;  5.4 修改字段的数据类型  alter table tb\_student change sage sage int(2) unsigned;  alter table tb\_student modify sage tinyint(2) unsigned;  5.5 修改表名  alter table tb\_student rename to t\_student; |

## DML

* 对表数据进行操作.

|  |
| --- |
| 1.insert  1.1 对所有的字段新增数据  insert into 表名 values (数据1,数据2...数据n),(数据1,数据2...数据n); ''  insert into tb\_student values  (1,'jim','M',80.5,'2019-01-01','2019-01-01 12:00:00' ,'2019-01-01 12:00:00',20);  1.2 对指定的字段新增数据(推荐)  insert into 表名 （字段1,字段2...字段n） values (数据1,数据2...数据n);  insert into tb\_student (sid,sname,gender,score,createtime) values  (2,'tom','m',100,now());  1.3 编码处理  不能添加中文数据：需要更改mysql服务的编码格式。  mysql的server: 默认的编码格式 latin 改成utf8  mysql的核心配置文件: my.ini  66 default-character-set=utf8  100 character-set-server=utf8  2.delete  delete from 表名 [where 条件 ]; 清空表数据 多条记录受影响  delete from tb\_student where sid =1;  3.update  update 表名 set 字段名1 = 值1,字段名2 = 值2 [where 条件]；多条记录受影响  update tb\_student set sage = 20 where sname = 'jim1' ; |

## 约束

* 约束是为了表的数据的正确性！如果数据不正确，那么一开始就不能添加到表中。

### 非空约束(not null)

* 指定非空约束的列不能没有值，也就是说在插入记录时，对添加了非空约束的列一定要给值；在修改记录时，不能把非空列的值设置为NULL

|  |
| --- |
| create TABLE a(  id int not null,  name varchar(20)  );  insert into a values (1,'jim');  insert into a (id) values (2);  Field 'id' doesn't have a default value |

### 唯一性约束(unique)

* 当为字段指定唯一约束后，那么字段的值必须是唯一的。(null值除外)

|  |
| --- |
| create table b(  id int not null UNIQUE,  name varchar(20) unique  );  insert into b values (4,null);  -- Duplicate entry 'jim' for key 'name |

### 默认约束(default)

|  |
| --- |
| create table c(  id int not null UNIQUE,  name varchar(20) not null DEFAULT '无名氏',  age TINYINT(2) not null  );  insert into c (id,age) values (2,20); |

### 主键约束(primary key)

* 当某一列添加了主键约束后，那么这一列的数据就不能重复出现。这样每行记录中其主键列的值就是这一行的唯一标识。主键列的值不能为NULL，也不能重复！
* 任意类型的字段都可以充当主键，一张表有且只有一个主键。
* 如果主键列是整数类型的，一般不会手动给值，都是结合mysql的server自动维护主键列的数据。(自增特性，主键自增长 auto\_increment)

|  |
| --- |
| create table d(  id int primary key auto\_increment,  name varchar(20) not null UNIQUE,  age tinyint(2)  );  insert into d (name,age) values ('张三',20);  修改自增的初始值:  alter table 表名 auto\_increment = 初始值; |

* 如果主键列是字符串类型，一般不会手动给值，都是结合mysql的server自动维护主键列的数据

|  |
| --- |
| create table a(  id varchar(64) primary key ,  name varchar(20) not null UNIQUE,  age tinyint(2)  );  insert into a (id,name,age) values (uuid(),'张三',20); |

### 外键约束(foreign key)

* 主外键是构成表与表关联的唯一途径！
* 外键是另一张表的主键！例如员工表与部门表之间就存在关联关系，其中员工表中的部门编号字段就是外键，是相对部门表的外键。
* 创建表，描述： 一个商品一个类型。

|  |
| --- |
| create table tb\_type(  id int primary key auto\_increment,  typename varchar(20) not null unique,  desc varchar(100),  createtime datetime,  updatetime datetime  ); |
| create table tb\_good(  id int primary key auto\_increment,  gname varchar(20) not null unique,  price float(10,2) not null,  createtime datetime,  updatetime datetime，  ); |

* 需要将商品表与类型表进行关联，一对一的关联关系，
* 商品表(从表/子表)typeid 的数据要严格参照类型表(主表) 主键id的数据。
* typeid: 外键列

|  |
| --- |
| Alter table tb\_good add typeid int(2);  alter table tb\_good add CONSTRAINT fk\_typeid  FOREIGN key (typeid ) REFERENCES tb\_type (id);  RESTRICT： 删除，修改都是受限。  CASCADE: 级联操作。 (不能使用)  SET NULL: 前提:外键列可以为null。(可以使用)  外键不推荐使用，弱化外键。(所有的关联都在逻辑层面进行处理) |

## 数据库引擎

INNODB 默认使用 VS MYISAM 早期使用

|  |  |  |
| --- | --- | --- |
|  | INNODB | MYISAM |
| 事务支持 | 支持 | 不支持 |
| 数据行锁定 | 支持(行锁) | 不支持(表锁) |
| 外键约束 | 支持 | 不支持 |
| 全文索引 | 支持(5.6 必须是英文) | 支持 |
| 表空间大小 | 较小 | 较大，约2倍 |

优势:

MYISAM: 节约空间，速度较快(小而快)

INNODB: 安全性高，事务的处理，多表多用户的操作。

在物理文件空间上的表示

INNODB: 在数据库表中只有一个\*.frm文件，以及上级目录的ibdata1文件。

MYISAM:

\*.frm ---表结构文件

\*.MYD ---数据文件(data)

\*.MYI ---索引文件(index)

## DQL

* DQL就是数据查询 select 语言，数据库执行DQL语句不会对数据进行改变，而是让数据库发送结果集给客户端。

### 语法

|  |
| --- |
| SELECT selection\_list /\*要查询的列名称\*/  FROM table\_list /\*要查询的表名称\*/  [WHERE condition /\*筛选数据行的条件\*/  GROUP BY grouping\_columns /\*对结果分组\*/  HAVING condition /\*分组后的筛选行的条件\*/  ORDER BY sorting\_columns /\*对结果排序\*/  LIMIT offset\_start, row\_count /\*结果限定\*/] |

### 准备表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 学生表：stu  |  |  |  | | --- | --- | --- | | **字段名称** | **字段类型** | **说明** | | **sid** | char(6) | 学生学号 | | **sname** | varchar(50) | 学生姓名 | | **age** | int | 学生年龄 | | **gender** | varchar(50) | 学生性别 | |
| CREATE TABLE stu (  sid CHAR(6),  sname VARCHAR(50),  age INT,  gender VARCHAR(50)  ); |
| INSERT INTO stu VALUES('S\_1001', 'liuYi', 35, 'male');  INSERT INTO stu VALUES('S\_1002', 'chenEr', 15, 'female');  INSERT INTO stu VALUES('S\_1003', 'zhangSan', 95, 'male');  INSERT INTO stu VALUES('S\_1004', 'liSi', 65, 'female');  INSERT INTO stu VALUES('S\_1005', 'wangWu', 55, 'male');  INSERT INTO stu VALUES('S\_1006', 'zhaoLiu', 75, 'female');  INSERT INTO stu VALUES('S\_1007', 'sunQi', 25, 'male');  INSERT INTO stu VALUES('S\_1008', 'zhouBa', 45, 'female');  INSERT INTO stu VALUES('S\_1009', 'wuJiu', 85, 'male');  INSERT INTO stu VALUES('S\_1010', 'zhengShi', 5, 'female');  INSERT INTO stu VALUES('S\_1011', 'xxx', NULL, NULL); |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 员工表emp  |  |  |  | | --- | --- | --- | | **字段名称** | **字段类型** | **说明** | | **empno** | int | 员工编号 | | **ename** | varchar(50) | 员工姓名 | | **job** | varchar(50) | 员工工作 | | **mgr** | int | 领导编号 | | **hiredate** | date | 入职日期 | | **sal** | decimal(7,2) | 月薪 | | **comm** | decimal(7,2) | 奖金 | | **deptno** | int | 部门编号 | |
| CREATE TABLE emp(  empno INT,  ename VARCHAR(50),  job VARCHAR(50),  mgr INT,  hiredate DATE,  sal DECIMAL(7,2),  comm decimal(7,2),  deptno INT  ) ; |
| INSERT INTO emp values(7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);  INSERT INTO emp values(7499,'ALLEN','SALESMAN',7698,'1981-02-20',1600,300,30);  INSERT INTO emp values(7521,'WARD','SALESMAN',7698,'1981-02-22',1250,500,30);  INSERT INTO emp values(7566,'JONES','MANAGER',7839,'1981-04-02',2975,NULL,20);  INSERT INTO emp values(7654,'MARTIN','SALESMAN',7698,'1981-09-28',1250,1400,30);  INSERT INTO emp values(7698,'BLAKE','MANAGER',7839,'1981-05-01',2850,NULL,30);  INSERT INTO emp values(7782,'CLARK','MANAGER',7839,'1981-06-09',2450,NULL,10);  INSERT INTO emp values(7788,'SCOTT','ANALYST',7566,'1987-04-19',3000,NULL,20);  INSERT INTO emp values(7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);  INSERT INTO emp values(7844,'TURNER','SALESMAN',7698,'1981-09-08',1500,0,30);  INSERT INTO emp values(7876,'ADAMS','CLERK',7788,'1987-05-23',1100,NULL,20);  INSERT INTO emp values(7900,'JAMES','CLERK',7698,'1981-12-03',950,NULL,30);  INSERT INTO emp values(7902,'FORD','ANALYST',7566,'1981-12-03',3000,NULL,20);  INSERT INTO emp values(7934,'MILLER','CLERK',7782,'1982-01-23',1300,NULL,10); |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 部门表：dept  |  |  |  | | --- | --- | --- | | **字段名称** | **字段类型** | **说明** | | **deptno** | int | 部门分编码 | | **dname** | varchar(50) | 部门名称 | | **loc** | varchar(50) | 部门所在地点 | |
| CREATE TABLE dept(  deptno INT,  dname varchar(14),  loc varchar(13)  ); |
| INSERT INTO dept values(10, 'ACCOUNTING', 'NEW YORK');  INSERT INTO dept values(20, 'RESEARCH', 'DALLAS');  INSERT INTO dept values(30, 'SALES', 'CHICAGO');  INSERT INTO dept values(40, 'OPERATIONS', 'BOSTON'); |

|  |
| --- |
| 3.工资等级表：salgrade  CREATE TABLE `salgrade` (  `GRADE` int(11) NOT NULL DEFAULT '0' COMMENT '工资的等级',  `LowSAL` decimal(7,2) DEFAULT NULL COMMENT '此等级的最低工资',  `HISAL` decimal(7,2) DEFAULT NULL COMMENT '此等级的最高工资'  ) ;  INSERT INTO `salgrade` VALUES  (1,700.00,1200.00),(2,1201.00,1400.00),  (3,1401.00,2000.00),(4,2001.00,3000.00),(5,3001.00,9999.00); |

### 基础查询

* 查询所有列

|  |
| --- |
|  |

* 查询指定列

|  |
| --- |
|  |

### 条件查询

* 条件查询就是在查询时给出WHERE子句，在WHERE子句中可以使用如下运算符及关键字：
* =、!=、<>、<、<=、>、>=；
  + BETWEEN…AND；是否满足一个区间范围 >= <=
  + IN(set)；条件的集合
  + IS NULL；
  + AND； 连接多个条件的查询
  + OR；or 满足其中一个条件就可以
  + NOT；
* 查询学生性别为女，并且年龄50的记录
* 查询学生性别为女，并且年龄50的记录
* 查询学号为S\_1001，S\_1002，S\_1003的记录
* 查询学号不是S\_1001，S\_1002，S\_1003的记录
* 查询年龄为null的记录
* 查询年龄在20到40之间的学生记录
* 查询性别非男的学生记录
* 查询姓名不为null的学生记录

|  |
| --- |
|  |

### 模糊查询(like)

* 查询姓名由5个字母构成的学生记录
* 查询姓名以“z”开头的学生记录
* 查询姓名中第2个字母为“i”的学生记录
* 查询姓名中包含“a”字母的学生记录

|  |
| --- |
|  |

### 字段控制查询

#### 去除重复的记录

* 去除重复记录（两行或两行以上记录中系列的上的数据都相同），例如emp表中sal字段就存在相同的记录。当只查询emp表的sal字段时，那么会出现重复记录，那么想去除重复记录，需要使用DISTINCT：
* 针对于某个字段的数据。
  + 查询学生表里面所有的性别。

|  |
| --- |
|  |

#### Null值运算

* 查询员工的月薪和佣金之和。

|  |
| --- |
|  |

#### 别名查询

* 在上面查询中出现列名为sal+IFNULL(comm,0)，这很不美观，现在我们给这一列给出一个别名，为total：
* 给列起别名时，是可以省略AS关键字的
* 表名，字段名等都可以使用别名查询。

### 排序(order by)

* 根据某个或者某些字段升序(默认 asc)或者降序(desc)排序。
  + 查询所有学生记录，按年龄升序排序
  + 查询所有学生记录，按年龄降序排序
  + 查询所有雇员，按月薪降序排序，如果月薪相同时，按编号升序排序

|  |
| --- |
|  |

### 分组(聚合)函数

* 聚合函数是用来做纵向运算的函数：
* COUNT()：统计指定列不为NULL的记录行数；
* MAX()：计算指定列的最大值，如果指定列是字符串类型，那么使用字符串排序运算；
* MIN()：计算指定列的最小值，如果指定列是字符串类型，那么使用字符串排序运算；
* SUM()：计算指定列的数值和，如果指定列类型不是数值类型，那么计算结果为0；
* AVG()：计算指定列的平均值，如果指定列类型不是数值类型，那么计算结果为0；
  + 查询emp表中记录数：
  + 查询emp表中有佣金的人数：注意，因为count()函数中给出的是comm列，那么只统计comm列非NULL的行数。
  + 查询emp表中月薪大于2500的人数：
  + 统计月薪与佣金之和大于2500元的人数：

|  |
| --- |
|  |

* + 查询有佣金的人数，以及有领导的人数：
  + 查询所有雇员月薪和：
  + 查询所有雇员月薪和，以及所有雇员佣金和：
  + 查询所有雇员月薪+佣金和：
  + 统计所有员工平均工资：

|  |
| --- |
|  |

* + 查询最高工资和最低工资

|  |
| --- |
|  |

### 分组查询(group by)

* 查询每个部门的部门编号和每个部门的工资和：
* 查询每个部门的部门编号以及每个部门的人数：
* 查询每个部门的部门编号以及每个部门员工工资大于1500的人数：

|  |
| --- |
|  |

### Having子句

* Where与having作用一致，都是对不满足条件的记录进行过滤。
* 二者也是有区别的：
  + haivng是在分组后对数据进行过滤. where是在分组前对数据进行过滤
  + having后面可以使用分组函数(统计函数) where后面不可以使用分组函数。
* 查询工资总和大于9000的部门编号以及工资和：

|  |
| --- |
|  |

### 关联查询(关联的是多种张表)

#### 等值连接：连接的条件是一个等号

* 案例：查询员工信息，要求显示员工号，姓名，月薪，部门名称

|  |
| --- |
|  |

#### 不等值连接：连接条件不是等号

* 案例：查询员工信息，要求显示：员工号，姓名，月薪，薪水的级别

|  |
| --- |
|  |

#### 外连接

* 内连接：inner join on 等同于等值连接
* 左外连接：left join on 以左表为基准，右表没有的数据使用null值进行填充。
* 右外连接：right join on 以右表为基准，左表没有的数据使用null值进行填充
* 案例:按部门统计员工数，部门号，部门名称，人数

|  |
| --- |
|  |

#### 自连接

* 核心：通过别名，将同一张表视为多张表
* 案例：查询员工姓名和员工的老板的名称

|  |
| --- |
|  |

#### 子查询

* 子查询的作用：查询条件未知的事物.
* 查询条件已知的问题：例如：查询工资为800的员工信息
* 查询条件未知的问题：例如：查询工资为20号部门平均工资的员工信息.

|  |
| --- |
|  |

#### 集合运算

* union：二个集合中，如果都有相同的，取其一(行记录)
* union all：二个集合中，如果都有相同的，都取

### 分页查询(limit)

* 对最后的结果进行限定（mysql独有）
* Select \* from 表 limit size; 指定从第一条记录开始查询 查size条
* Select \* from 表 limit ?,size 指定从第n条记录(index)开始查询 查size条

## 常用函数

### 字符串函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **函数** | **功能** | | concat(str1,str2,…) | 连接字符串 | | insert(str,pos,len,newstr) | 字符串str从第pos位置开始的len个字符替换为新字符串newstr | | lower(str) | 转成小写 | | upper(str) | 转成大写 | | length(str) | 返回字符串str的长度 | | char\_length(str) | 返回字符串str的长度 | | lpad(str,ien,padstr) | 返回字符串str，其左边由字符串padstr填补到len字符串长度 | | rpad(str,len,padstr) | 返回字符串str，其左边由字符串padstr填补到len字符串长度 | | trim(str) | 去掉字符串str前缀和后缀的空格 | | repeat(str,count) | 返回str重复count次的结果 | | replace(str,from\_str,to\_str) | 用字符串to\_str替换字符串str中所有的字符串from\_str | | substring(str,pos,len) | 从字符串str的pos位置起len个字符长度的子串 | |

### **数值**函数

|  |
| --- |
|  |

### 日期和时间函数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **函数** | **功能** | | CURDATE() | 返回当前日期 | | CURTIME() | 返回当前时间 | | NOW() | 返回当前的日期和时间 | | WEEK(date) | 返回指定日期为一年中的第几周 | | YEAR(date) | 返回日期的年份 | | HOUR(time) | 返回time的小时值 | | MINUTE(time) | 返回time的分钟值 | | MONTHNAME(date) | 返回date的月份名 | | DATEDIFF(expr,expr2) | 返回起始时间expr和结束时间exrp2之间的天数 | | DATE\_FORMAT(date,fmt) | 返回按字符串fmt格式化日期date值 | | from\_unixtime(unix\_timestamp,”%Y-%m-%d %H:%i:%S”)常用来将毫秒数转换为时间格式 | | | |
| %M 月名字(January……December)  %W 星期名字(Sunday……Saturday)  %D 有英语前缀的月份的日期(1st, 2nd, 3rd, 等等。）  %Y 年, 数字, 4 位  %y 年, 数字, 2 位 20  %a 缩写的星期名字(Sun……Sat)  %d 月份中的天数, 数字(00……31)  %e 月份中的天数, 数字(0……31)  %m 月, 数字(01……12)  %c 月, 数字(1……12)  %b 缩写的月份名字(Jan……Dec)  %j 一年中的天数(001……366)  %H 小时(00……23)  %k 小时(0……23)  %h 小时(01……12) | %I 小时(01……12)  %l 小时(1……12)  %i 分钟, 数字(00……59)  %r 时间,12 小时(hh:mm:ss [AP]M)  %T 时间,24 小时(hh:mm:ss)  %S 秒(00……59)  %s 秒(00……59)  %p AM或PM  %w 一个星期中的天数(0=Sunday ……6=Saturday ）  %U 星期(0……52), 这里星期天是星期的第一天  %u 星期(0……52), 这里星期一是星期的第一天  %% 一个文字“%”。 |