**##数据库的好处**

**1.持久化数据**

**2.结构化查询**

**##数据库相关概念**

**1.DB：数据库，保存一组有组织的数据的容器**

**2.DBMS：数据库管理系统(数据库软件)，用于管理DB中的数据**

**3.SQL：结构化查询语言，用于和DBMS通信的语言**

**##配置文件**

**my.ini:**

**# SERVER SECTION**

**1.默认端口：3306**

**2.默认字符集：utf-8**

**3.默认存储引擎：INNODB**

**##MySQL常用命令**

**1.查看当前所有的数据库**

**show databases ;**

**2.打开指定的数据库**

**use 库名;**

**3.查看当前库中所有表**

**show tables;**

**4.查看其他库中所有表**

**show tables from 库名;**

**5.查看数据库版本**

**select version();**

**##MySQL语法规范**

**1.不区分大小写，建议关键字大写，表名、列名小写**

**2.建议分号结尾**

**3.根据需要可以缩进、换行**

**##MySQL查询**

**1.查询出来的结果是一张虚拟表**

**2.字段起别名**

**as关键字(可省略)**

**select name (as) ‘姓名’ from student;**

**3.去重**

**distinct关键字**

**select distinct department\_id from employee;**

**4.mysql中的+号**

**Java中的+号：①运算符 ②字符串连接符**

**Mysql中的+号：仅运算符**

**①都为数值型，则做加法运算**

**②一方为字符型，尝试将字符型转换成数值型**

**如果转换成功，则继续做加法运算，否则将字符型数值转换为0**

**③只要其中一方为null，则结果为null**

**5.mysql中的字符串拼接**

**concat()函数**

**select concat(‘a’,’b’,’c’) as 结果 from table;**

**6.条件表达式**

**(1)条件运算符：> < = !=/<> >= <=**

**(2)逻辑运算符：and/&& or/|| not/!**

**(3)模糊查询：like between and in is null**

**7.like关键字**

**通配符：**

**①%：任意多个字符，包括0个字符**

**②\_：任意一个字符**

**③：%和\_可以组合使用**

**8.between and关键字**

**①包含临界值**

**②两个临界值不要调换顺序**

**9.is null/is not null关键字**

**注意：①=或<>不能判断null值，只能使用is null**

**②is只能用于null，不能用于其他值**

**10.排序**

**order by 字段名1 asc/desc,字段名2 asc/desc...;**

**##MySQL函数**

**1.字符函数**

**(1)length()：获取字段的【字节】个数，字母占一个字节，汉字占3个字节(utf-8)**

**select \* from student where LENGTH(name) = 6;**

**(2)concat()：拼接字符串**

**(3)upper()：转换大写**

**lower()：转换小写**

**select upper(name) from student;**

**(4)substr()：截取**

**注意：mysql索引从1开始，该方法有多个重载**

**select substr(name,3) from student;#从第3个字符开始截取到结尾**

**select substr(name,1,3) from student;#从第1个字符开始截取3个【字符】**

**(5)instr()：返回子串第一次出现的索引，如果找不到返回0**

**(6)trim()：去除前后空格**

**trim(‘a’ from 字段名)：去除前后指定字符**

**(7)lpad()：左填充指定字符直至指定长度**

**rpad()：右填充指定字符直至指定长度**

**select lpad(name,3,'a') from student;**

**(8)replace()：替换**

**select replace(name,’张三’,’李四’) from student;**

**2.数学函数**

**(1)round()：四舍五入(去掉第n+1位后四舍五入)**

**select round(1.57777) from menu;#n=0，->2**

**select round(1.57777,2) from menu;#n=2，保留2位四舍五入->1.58**

**(2)ceil()：向上取整(返回>=当前数的最小整数)**

**floor()：向下取证(返回<=当前数的最大整数)**

**(3)truncate()：截断(只舍不进)**

**select truncate(1.699999,1);#->1.6**

**(4)mod()：取余**

**select mod(10,3);#->3**

**等效于select 10%3;**

**3.日期函数**

**(1)now()：返回当前系统日期+时间**

**select now();**

**(2)curdate()：只返回当前系统日期**

**curtime()：只返回当前系统时间**

**(3)year()/month()/day()/hour()/minute()/second()：**

**只返回指定字段或常量值的年/月/日/时/分/秒**

**如果是字段就加上from表名，如果是常量值则不需要**

**(4)str\_to\_date()：字符串通过指定格式转换成日期**

**select str\_to\_date('1998/03/02/18/47/03','%Y/%m/%d/%H/%i/%s');**

**#->1998-03-02 18:47:03**

**(5)date\_format()：日期通过指定格式转换成字符串**

**select date\_format('1998-03-02 18:47:03','%Y/%m/%d/%H/%i/%s');**

**#->1998/03/02/18/47/03**

**4.流程控制函数**

**(1)if函数**

**select if (fee is not null,fee,’免费’) from menu;**

**(2)case函数**

**①类似于switch-case的用法：**

**case 字段**

**when 常量1 then 要显示的值1或语句1**

**when 常量2 then 要显示的值2或语句2**

**...**

**else 要显示的值n或语句n;**

**end**

**select salary 原始工资**

**case department\_id**

**when 30 then salary\*1.1**

**when 40 then salary\*1.2**

**when 50 then salary\*1.3**

**end as ‘新工资’ from employee;**

**②类似于多重if-else的用法：**

**case**

**when 表达式1 then 要显示的值1或语句1**

**when 表达式2 then 要显示的值2或语句2**

**...**

**else 要显示的值n或语句n;**

**end**

**select salary,**

**case**

**when salary >20000 then ‘A’**

**when salary >15000 then ‘B’**

**when salary >10000 then ‘C’**

**else ‘D’**

**end as ‘工资级别’ from employee;**

**6.分组函数(用于统计)**

**(1)sum()：求和，一般数值型，忽略null值**

**(2)avg()：平均值，一般数值型，忽略null值**

**(3)max()：最大值，任何类型，忽略null值**

**(4)min()：最小值，任何类型，忽略null值**

**(5)count()：总数，任何类型，忽略null值**

**select sum(salary),avg(salary),max(salary),min(salary),count(salary) from employees;**

**分组函数都支持和distinct关键字一起使用**

**select count(distinct salary) from employees;**

**count(\*)/count(1)：总行数**

**注意：**

**(1)聚合函数和group by一起使用时，统计的是结果集中每一组的数据。聚合函数 单独使用时，统计的是结果集中的所有数据。**

**(2)和分组函数一同查询的字段要求是group by后面的字段。因为查询出 来的结果集中只有每组的第一条，而只有分组的字段才是组内所有行都相同的。**

**##分组查询：**

**1. group by关键字和having关键字**

**select count(\*),department\_id from employee**

**group by department\_id having count(\*)>2;#查询员工人数大于2的部门**

**group by后面也可以跟函数(不常用)，例如根据姓名长度进行分组。**

**2.having关键字和where关键字的区别：**

**(1)where放在group by之前，having放在group by之后。**

**(2)where在group by之前执行，having在group by之后执行，对分组结果进行过 滤。**

**3.按多个字段进行分组：**

**select count(\*),department\_id,job\_id from employees**

**group by department\_id,job\_id;#department\_id和job\_id都相同的被分为一组，没 有顺序要求。**

**4.分组后排序：**

**order by放在group by和having之后，在group by和having之后执行，对分组过 滤结果进行排序。**

**注意：having后面只能跟group by或者select后面出现的的字段，否则会报错。**

**##多表查询**

1. **笛卡尔乘积现象(交叉连接)：没有有效的连接条件，查询结果为l\*n行**

**语法：select 查询字段 from 表1 别名 [连接类型] join 表2 别名 on 连接条件**

**(1)内连接：inner**

**应用场景：用于查询两个表的交集部分。**

**n表连接，至少需要n-1个连接条件**

**①等值连接**

**select name,job from employees e inner join jobs j on e.job\_id = j.job\_id**

**where e.name like ‘%e%’;#查询名字中包含e的员工名和工作岗位名**

**②非等值连接**

**select salary,grade\_level from employees e inner join grade g on e.salary**

**between g.lowest\_salary and g.highest\_salary;#查询员工工资及工资级别**

**③自连接(将一张表拆成两张表进行连接)**

**select e.employee\_id,e.name,m.employee,m.name**

**from employees e,employees m on e.manager\_id = m.employee\_id;#查询员工 姓名和上级姓名**

**(1)外连接**

**应用场景：用于查询一个表中有，另一个表中没有的记录。**

**特点：外连接的查询结果为主表中的所有记录，如果从表中有和它匹配的，则显 示匹配的值，如果从表中没有和它匹配的，则显示null**

**外连接查询结果=内连接结果+主表有而从表没有的记录**

**左外连接(left)：左表为主表**

**右外连接(right)：右表为主表**

**左外和右外交换两个表的顺序，可以实现相同的效果**

**select d.department\_id from department d left join employees e**

**on d.department\_id = e.department\_id where e.employee\_id is null;**

**#查询哪个部门没有员工。部门表为主表，显示所有数据。员工表为从表，如 果部门中没有员工则外连接后的记录的employee\_id字段为null**

**全外连接(full)：MySQL不支持全外连接**

**全外连接查询结果=内连接结果+表1有而表2没有的记录+表2有而表1没 有的记录。**

**##子查询**

**概念：出现在其他语句内部的select语句，称为子查询**

**分类：**

**(1)按子查询出现的位置不同：**

**①select后面：仅仅支持标量子查询**

**②from后面：支持表子查询**

**③where或having后面(\*)：支持标量子查询(\*)、列子查询(\*)、行子查询**

**④exists后面：支持表子查询**

**(2)按子查询结果集的行列数不同：**

**①标量子查询：一行一列(子查询的结果对应一个值，如1)**

**②列子查询：一列多行(子查询的结果对应一个列表，如(1,2,3))**

**③行子查询：一行多列**

**④表子查询：多行多列(子查询的结果对应一张中间表)**

**特点：**

**①子查询放在小括号内，可以加上缩进提高可读性。**

**②子查询一般放在条件的右侧**

**③子查询的执行优先于主查询执行，主查询的条件用到了子查询的结果**

**④子查询可以嵌套子查询**

**⑤标量子查询，一般搭配单行操作符使用> < >= <= <>**

**⑥列子查询，一般搭配多行操作符使用in/not in、any/some、all**

**a > any(10,20,30)：表示a>括号中任意一个值即可，可以使用min函数代替。**

**a > all(10,20,30)：表示a>括号中所有值，可以使用max函数代替。**

**1.where或having后面的子查询：**

**#标量子查询：查询工资比Abel工资高的员工id**

**select employee\_id from employees where salary > (**

**select salary from employees where name = ‘Abel’**

**);**

**#列子查询：查询上海或北京的部门中的所有员工姓名**

**select name from employees where department\_id in(**

**select department\_id from department where city in(‘上海’,’北京’)**

**)**

**2.from后面的子查询：**

**#查询每个部门的平均工资对应的工资等级**

**select a.department\_id, g.grade\_level from(**

**select department\_id,avg(salary) from employees group by department\_id**

**) a inner join job\_grades g on avg(salary) between lowest\_salary and highest\_salary;**

**3.exists后面的子查询：**

**语法：exists(完整的查询语句)**

**结果：1或者0**

**#查询有员工的部门名**

**select department\_name from departments d where exists(**

**select \* from employee e where d.department\_id = e.department\_id**

**);**

**分析：先执行外查询，查询出所有部门记录(注意：此时只执行了from departments， 还没执行select department\_name，所以子查询中可以使用外查询结果的全部字 段)。然后将外查询的每一条记录作为子查询的条件进行匹配。如果部门中存在员 工则子查询返回个数大于0则保留这条外查询记录。如果部门中不存在员工子查 询返回的个数等于0则丢弃掉这条外查询记录。最后执行select department\_name 获取符合条件的记录的字段。**

**#使用in替代**

**select department\_name from departments d where department\_id in(**

**select department\_id from employees;**

**)**

**分析：先执行内查询，查询出所有有员工的部门的id。再执行外查询，通过in关 键字进行过滤。**

**##分页查询**

**limit offset,size;**

**offset：要显示数据记录的起始索引(从0开始，如果是0则可以省略)**

**size：要显示的数据记录个数**

**limit 0,5：从第一条开始，显示5条**

**page：页码数**

**size：每页显示数量**

**limit (page-1)\*size,size;**

**##联合查询(union关键字)**

**合并：将多条查询语句的结果合并成一个结果**

**#查询姓王或者姓李的员工**

**select \* from employee where name like ‘%王%’ or name like ‘%李%’;**

**->**

**select \* from employee where name like %王%’ union**

**select \* from employee where name like %李%’;**

**应用场景：要查询的结果来自于多张表，且多张表没有直接的连接关系，但查询的信 息一致时。**

**#**

**select china\_student\_id,china\_student\_name from china\_student**

**union**

**select america\_student\_id,america\_student\_name from america\_student**

**结果集中的字段只有china\_student\_id,china\_student\_name，**

**结果集中的数据包括china\_student表和america\_student。**

**注意：**

1. **多条查询语句查询列数必须相同**
2. **多条查询语句查询的每一列的类型和顺序最好一致**
3. **使用union联合查询默认会去重，如果不需要可以使用union all替代**

**##增删改**

**插入：insert into**

**语法：Insert into 表名(列名...) values(值1...);**

**批量插入：Insert into 表名(列名...) values(值1...),(值2...);**

**注意：插入的值的类型要与列的类型一致或兼容(隐式转换)，日期型使用单引号。**

**可以省略列名，此时需要设置所有列的值。**

**修改：update set**

**语法：update 表名 set 列名1=值1,列名2=值2... where 筛选条件;**

**多表更新：update 表1 连接 表2 set 列名1=值1,列名2=值2... where 筛选条件;**

**#修改张无忌的女朋友的年龄为18**

**update boys b inner join girls g on b.id =g.boyfriend\_id set g.age = 18 where g.boyfriend\_name = ‘张无忌’;**

**删除：delete from**

**语法：delete from 表名 where 筛选条件;**

**多表删除：delete 要删除数据的表的别名 from 表1 连接 表2 where 筛选条件;**

**#删除张无忌的女朋友的信息**

**delete g from boys b inner join girls g on b.id = g.boyfriend\_id where g.boyfriend\_name = ‘张无忌’;**

**#删除张无忌和他女朋友的信息**

**delete b.g from boys b inner join girls g on b.id = g.boyfriend\_id where g.boyfriend\_name = ‘张无忌’;**

**清空表：truncate table 表名;**

**truncate删除和delete删除的区别【面试题】：**

**(1)truncate不能加where条件**

**(2)truncate效率高于delete from 表名;**

**(3)如果要删除的表中有自增长列，使用delete删除后再插入数据，自增 长列的值从断点开始，而使用truncate删除后再插入数据，自增长列 的值从1开始**

**(4)truncate删除没有返回值(受影响的行数)，delete删除有返回值**

**(5)truncate删除不能回滚，delete删除可以回滚**

**##DDL数据定义语言**

**#操作数据库：**

**1.创建：create database 库名;/create database if not exists 库名;**

**2.修改数据库字符集：alter database 库名 character set gbk;**

**3.删除：drop database 库名;/drop database if exists 库名;**

**#操作数据表：**

1. **创建：create table if not exists 表名(**

**列名 列的类型【(长度) 约束】,**

**列名 列的类型【(长度) 约束】,**

**...**

**列名 列的类型【(长度) 约束】**

**);**

**【】表示可省略，会有默认长度，但是varchar类型必须加长度**

1. **修改**
2. **修改列名(也可以修改列的类型或约束)：**

**alter table 表名 change column 旧列名1 新列名1 新类型【(长度) 约束】,**

**change column 旧列名2 新列名2 新类型【(长度) 约束】;**

**注意：类名不需要加引号**

1. **修改列的类型或约束：**

**alter table 表名 modify column 列名1 新类型【(长度) 约束】,**

**(modify column 列名2 新类型【(长度) 约束】...);**

1. **添加新列：**

**alter table 表名 add column 列名1 列的类型【(长度) 约束】 after 列名,**

**(add column 列名2 列的类型【(长度) 约束】...);**

1. **删除列：**

**alter table 表名 drop column 列名1**

**(,drop column 列名2...);**

1. **修改表名：**

**alter table 旧表名 rename to 新表名;**

1. **删除**
2. **删除表：drop table if exists 表名;**
3. **复制**
4. **仅仅复制表的(全部)结构**

**create table 新表名 like 旧表名;**

**新表会复制旧表全部列**

1. **复制表的(部分)结构和(部分)数据**

**create table 新表名 select 列名1,列名2... from 旧表名 where 筛选条件;**

**新表只会复制旧表指定列**

1. **仅仅复制表的(部分)结构**

**create table 新表名 select 列名1,列名2... from 旧表名 where 0;**

**0在这里代表恒不成立，非0代表恒成立，或者写where 1=2也可以。**

**##数据类型**

**字段类型的选择原则：所选择的类型越简单越好，保存数值的类型越小越好(节省空间)。**

1. **数值型**

**①整型(范围由小到大)：**

**tinyint：1个字节，有符号：-128-127/无符号：0-255**

**smallint：2个字节**

**mediumint：3个字节**

**Int/integer：4个字节**

**bigint：8个字节**

**设置无符号：创建表或修改表时在字段类型后面加unsigned关键字，如果不加 则为有符号。**

**注意：范围只由类型决定，长度代表了显示的最大宽度。**

**1.如果是整型，即使插入了大于显示宽度的值，只要该值不超过该类型的 取值范围，数值依然可以插入，而且能够正常显示出来。**

**2.如果是字符型，这个宽度才真正的用上了。不管是char还varchar，宽 度都定义了字符串允许插入的最大【字符】数。例如：varchar(5)，那 么无法插入’aaaaaa’或者’呜呜呜呜呜呜’。**

1. **浮点型(float或double)日期类型对数据的宽度没有要求，一般也不设 置，默认是0。**

**②浮点型：**

**float(M,D)：4个字节**

**double(M,D)：8个字节**

**③定点型：**

**decimal(M,D)：精度比double高，如果对精度要求较高，建议使用此类型**

**说明：**

**1.M：整数部分和小数部分的总长度。D：小数点后面保留的位数**

**2.M和D都可以省略**

**3.如果是decimal，M默认是10，D默认是0。如果是float或者double，M和 D默认都是0，此时会根据插入数值的精度来决定精度，如果设置了M或 者D则参考第一条。**

1. **字符型：**

**①较短的文本：char(M)、varchar(M)**

**char和varchar的区别：**

**char是定长，varchar是可变长度**

**例如：**

**char(10)，保存’张三’，也会保存为10个字符，耗费空间，但是性能稍高。 可以省略长度，默认为1。**

**varchar(10)，保存’张三’，保存为2个字符，节省空间，但是性能略低。**

**不能省略长度。**

**②较长的文本：text、blob(较大的二进制)**

1. **日期型：**

**日期型设置时需要使用单引号**

**date：4个字节，只包含日期：2020-05-01**

**datetime：8个字节，包含日期和时间：2020-05-01 08:00:00**

**timestamp：4个字节，时间戳**

**datetime和timestamp的区别：**

1. **timestamp占用字节数更小**
2. **datetime范围1000-9999，timestamp范围：1970-2038**
3. **datetime不受时区的影响，timestamp受时区的影响**

**补充sql命令：**

**查看当前时区：show variables like ‘time\_zone’;**

**重新设置时区：set time\_zone = ‘+9:00’#设置时区为东九区**