**增(insert) 删(delete) 改(update) 查((select)**

1. 增: insert into
2. 删: delete from
3. 改:update table
4. 查:select \* from

**--创建数据库;(若不写character set utf8则默认用的lantil编码集，而这种编码集在插入中文的时候会出现乱码)**

**CREATE DATABASE IT CHARACTER SET UTF8;**

**--创建表;**

**CREATE TABLE `user`(**

**username VARCHAR（50）,**

**`password` VARCHAR（50）,**

**sex CHAR (10),**

**age INT,**

**reg\_date TIMESTAMP**

**);**

**--修改表的名字 (表的重命名)**

**ALTER TABLE `user` RENAME person;**

**--DML(增、删、改)**

**--增**

**--添加数据**

**(NOW)--获取系统时间**

**数据的插入:insert into 表明称 value([插入的数据])**

**INSERT INTO person VALUES("小明","123455","男",22,NOW());**

**--删除表(DROP)**

**DROP TABLE person;**

**--删除数据库**

**DROP DATABASE IT;**

**插入指定数据:insert into person([指定插入数据的列名]) value([插入的数据])**

**INSERT INTO person (username,password,reg\_date) VALUES（"小红","123456",NOW（））；**

**--其他插入数据方式**

**INSERT INTO person SELECT \* FROM person;**

**--删除(delete)**

**--删除年龄为25**

**DELETE FROM person WHERE age=25;**

**--删除年龄在25和60之间的**

**DELETE FROM person WHERE age>2 AND age>0;**

**--删除字符(一下划线表示一个字符)**

**DELETE FROM person WHERE username LIKE"\_"；**

**--删除前面带有模糊的字符(%属于模糊查询)**

**DELETE FROM person WHERE username LIKE"模糊%";**

**--删除中间带有模糊字符的**

**DELETE FROM person WHERE username LIKE"%模糊%";**

**--删除后面带有模糊的字符**

**DELETE FROM person WHERE username LIKE"%模糊";**

**--改(update : update 表名 set 字段=值，字段=值 where 条件)**

**将小明年龄增加十岁**

**UPDATE person set age=age+10 WHERE username = "小明";**

**将小明的年龄增加十岁，性别改为女**

**UPDATE person set age=age+10，sex="女" WHERE username = "小明";**

**int:整型**

**double:浮点型 例如double(5,2)表示最多五位，其中必须有2位小数,即最大值为999.99**

**decimal:浮点型，在单表钱方面使用该类型，因为不会出现精度缺失问题**

**char:固定长度字符串类型:char(255)**

**varchar:可变长度字符串类型 :varchar(65535),**

**text(clob) :字符串类型**

**很小:tinytext2^8-1B**

**小:text 2^16-1B**

**中:mediumtest 2^24-1B**

**大:longtext 2^32-1B**

**blob:字节类型**

**很小:tinytext2^8-1B**

**小:text 2^16-1B**

**中:mediumtest 2^24-1B**

**大:longtext 2^32-1B**

**date:日期类型，格式为:YYYY-MM-DD**

**time:时间类型，格式为: hh:mm:ss**

**timestamp: 时间戳类型: YYYY-MM-DD hh:mm:ss**

**升序**

**select \* from 表 where 条件 order by 字段 asc;**

**降序**

**select \* from 表 where 条件 order by 字段 desc;**

**1、查询出部门编号为30的员工**

**SELECT \* FROM t\_employess WHERE deptno = 30;**

**2、所有销售员的姓名、编号和部门编号**

**SELECT ename,empno,deptno FROM t\_employess WHERE job = "salesman";**

**3、找出奖金高于工资的员工**

**SELECT \* FROM t\_employess WHERE comm>sal;**

**4、找出奖金高于工资60%的员工**

**SELECT \* FROM t\_employess WHERE comm>sal\*0.6;**

**5、找出部门编号为10中所有经理和部门编号为20中所有销售员的详细资料**

**SELECT \* FROM t\_employess WHERE ( deptno = 10 AND job="manger") OR (deptno = 20 AND job = "salesman");**

**6、找出部门编号为10中所有经理，部门编号为20中所有销售员，还有既不是经理又不是销售员但其工资大或等于20000的所有员工详细资料**

**SELECT \* FROM t\_employess WHERE (deptno = 10 AND job = "manger") OR (deptno = 20 AND job = "salesman")**

**OR (job NOT IN ("manger","salesman") AND sal >=20000);**

**7、无奖金或者奖金低于1000的员工**

**SELECT \* FROM t\_emlopyess WHERE comm IS NULL OR comm<1000;**

**8、查询名字由三个字组成的员工**

**SELECT \* FROM t\_emlopyess WHERE ename LIKE"\_\_\_";**

**9、查询2000年入职的员工**

**SELECT \* FROM t\_employess WHERE hiredate LIKE"2000%";**

**10、查询所有员工详细信息，用编号升序排序**

**SELECT \* FROM t\_employess ORDER BY empno ASC;**

**11、查询所有员工详细信息，用工资降序排序，如果工资相同使用入职日期升序排序**

**SELECT \* FROM t\_employess ORDER BY sal DESC,hiredate ASC;**

**12、查询每个部门的平均工资 group by 字段 avg:平均值(字段)**

**SELECT deptno,AVG(sal) FROM t\_employess GROUP BY deptno;**

**13、查询每个部门的雇员数量 count:数值()字段**

**SELECT deptno,COUNT(empno) FROM t\_employess GROUP BY deptno;**

**14、查询每种工种的最高工资max（）、最低工资min()、人数count（）**

**SELECT job,MAX(sal),MIN(sal),count(\*) FROM t\_employess GROUP BY job;**

1. **查询男生人数**

**select count(\*) as 男生人数 from student where sex=’男’;**

**关系型数据库:mysql Oracle、DB2、PosetgreSQL、Microsoft SQLServer、Microsoft Access、MySql、浪潮K-DB等**

**非关系型数据库:NoSql、Cloudant、MongoDB、redis、HBase**

**数据库储存数据的优点：**

**1、可大量储存数据**

**2、方便检索**

**3、保持数据的一致性、完整性**

**4、安全，可共享**

**5、通过组合分析，可产生新数据**

**数据库的发展历程。**

1. **没有数据库，使用磁盘文件储存数据**
2. **层次结构模型数据库**
3. **网状结构模型数据库**
4. **关系结构模型数据库:使用二维表格来存储数据**
5. **关系对象模型数据库**

**数据库核心:RDBMS（DBMS）：数据库管理系统**

**理解数据库。**

**RDBMS = 数据库(DB) +数据库管理员(manager)**

**DB = N个表**

**表 = N条记录(数据)组合**

**SQL语句分类**

1. **DDL（Data Definition language）:数据定义语言，用来定义数据库对象:库、表、列等**

**创建、删除、修改:库、表结构!!!**

1. **DML(Data Manipalation Language):数据操作语言，用来定义数据库记录(数据):**

**增、删、改:表记录**

1. **DCL(Data Control Language) 数据控制语言，用来定义访问权限和安全级别:**
2. **DQL(Data Query Language):数据查询语言，用来查询记录(数据)。**

**---修改本机登陆密码:root**

**grant all privileges(将所有的权限) on \*.\*(所有数据库的所有表) to ‘root’@’localhost(本机用户)’ identified by ‘root’ with grant option;**

**---给其他用户分配登录密码:root，别人登录我们这台主机，需要用root登录才行**

**:grant all privileges(将所有的权限) on \*.\*(所有数据库的所有表) to ‘root’@’%(匹配所有其他用户)’ identified by ‘root’ with grant option;**

**-uroot -proot -h[别人的IP地址]**

**查看表结构:基本表信息(DESC)**

**DESC person；**

**CREATE TABLE t1(**

**sal DOUBLE(3,2) 其中3表示总共有3位数，2表示占两位小数 能表示的最大值为9.99 如实际插入数据的时候大于9.99，存储失效，会以最大值显示9.99**

**);**

**Date 日期: 2010-9-6**

**Time 时间:15:30:30**

**Timestamp 时间戳(日期+时间): 2010-9-3 15:20:15**

**查看当前数据库所有表名称**

**Show tables**

**查看指定表的创建语句**

**Show create table [表名称]**

**查看表结构**

**Desc [表名称]**

**删除表**

**Drop table [表名称]**

**添加表:**

**---添加列:alter table [表名称] add [列名] 数据类型**

**ALTER table person add age int;**

**----删除列:alter table [表名称] drop [列名]**

**ALTER table person drop age**

**-----修改列的名称:alter table [表名称] change [旧列名] [新列名] 数据类型**

**ALTER table person change age sex CHAR(20);**

**----修改列的数据类型:alter table [表名称] modify 列名 数据类型**

**ALTER table person modify sex CHAR（50）**

**----修改表的名称:alter table person rename to [新表名];**

**ALTER table person rename to KL;----将person改为KL**

**数据库的备份和导入**

**1、备份**

**在dos窗口输入mysqldump -u[用户名] -p[密码] [要备份数据库的名称] > [路径+文件名的名称]**

1. **导入**
2. **在dos窗口输入mysql -u[用户名] -p[密码] < [数据库文件]**
3. **先登录到mysql然后在mysql命令行里输入:source[数据库文件名称]**

**访问他人服务器:grant all privileges on \*.\* to 'bc'@'%' identified by 'root';**

**flush privileges;**

**主键的创建**

**单个主键的创建**

**Create table person(**

**字段1 类型 primary key**

**);**

**多个主键的创建**

**Ccreate table person(**

**字段1 类型**

**字段2 类型**

**Primary(字段1,字段2)**

**);**

**主键:**

**----能唯一描述数据;唯一性，描述数据**

**----插入数据的时候，primary key修饰的列插入数据不能为空，非空性**

**----可被引用**

**主键的自增长 ----int，primary kry，auto\_increment**

**Create table person(**

**字段 INT PRIMARY KEY AUTO\_INCREMENT**

**);**

**外键:**

1. **外键插入的值必须是另一张主键的值(外键引用主键)**
2. **外键可以重复的值**
3. **外键可以为空**
4. **一个表可以有多个外键**

**Create table 用户表(**

**Id VARCHAR(50) PRIMARY KEY,**

**Username VARCHAR(50),**

**);**

**Create table 商品表(**

**Sid VARCHAR(50) PRIMARY KEY,**

**Sname varchar(50)**

**);**

**Create table 订单表(**

**Id varchar(50),**

**Sid varcahr(50),**

**Num int,**

**Primary key(id,sid),**

**-------设置外键:把本表的属性作为外键来连接其他表的主键**

**FOREIGN KEY (id) REFERENCES 用户表(id)，**

**FOREIGN KEY (sid) REFERENCES 商品表(sid)**

**);**

**-----插入数据**

**INSERT INTO 用户表 VALUE(“001”,”名字”);**

**INSERT INTO 商品表 VALUE(“011”,”冰箱”);**

**INSERT INTO 订单表 VALUE(“001”,”011”,”60”);**

**学生表 教师表**

**Create table 学生表(**

**Sname char(50) primary key,**

**Sex char(20)**

**);**

**Create table 教师表(**

**Tname char(50) PRIMARY KEY,**

**Sex char(20)**

**);**

**Create table 教学表(**

**Sname char(50),**

**Tname char(20),**

**PRIMARY KEY(sname,tname),**

**FOREING KEY (sname) REFERENCES 学生表(sname),**

**FOREING KEY (tname) REFERENCES 教师表(tname)**

**);**

**子查询(嵌套查询):**

**用法:**

1. **在from后**
2. **在where后**

|  |
| --- |
| 1. **--查询工资与scott一样的其他人的姓名**   **SELECT ename FROM employess WHERE sal =(SELECT sal FROM employess WHERE ename = “scott”);**   1. **---查询工资高于平均工资的那些人**   **SELECT \* FROM employess WHERE sal > (SELECT AVG(sal) FROM employess);** |

**笛卡尔积:就是将两个或者多个集合序列重新组合成一个新的集合。**

**消除笛卡尔积:**

**将两个或者多个集合相同的部分进行关联，从而舍去无效数据**

|  |
| --- |
| **---消除笛卡尔积**  **SELECT 表名.”列名”,表名.”列名”,employess.\* FROM 表名,表名 WHERE 表名.”列名1” = 表名.”列名1”;** |

**---标准写法(join on: 连接查询，自然连接)**

**SELECT \* FROM t\_employess e JOIN t\_dept t ON e.’deptno’ = t.’deptno’;**

**---右连接:right outer**

**SELECT \* FROM t\_employess e RIGHT OUTER JOIN t\_dept t ON e.’deptno’ = t.’deptno’;**

**---右连接:将右表作为主表，若左表没有的数据，用null进行显示**

**---列出部门名称和这些部门的员工信息，同时列出那些没有员工的部门。**

**SELECT \* FROM t\_employess e RIGHT OUTER JOIN t\_dept t ON e.’deptno’ = t.’deptno’;**

**笔试题:视图和表的区别**

**数据库中的数据都是存储在表中的，而视图只是一个或多个表依照某个条件组合而成的结果集，一般来说你可以用update，insert，delete等sql语句修改表中的数据，而对视图只能进行select操作.。但是也存在可更新的视图，对于这类视图的update，insert和delete等操作最终会作用于与其相关的表中数据。因此，表是数据库中数据存储的基础，而试图只是为了满足某种查询要求而建立的一个对象。**

**简单来讲：**

**表是物理存在的，可以理解成计算机中的文件!**

**视图是虚拟的内存表，你可以理解成windows的快捷方式!**

**----i视图 : view: create view [名称] as [sql语句];**

**CREATE VIEW findAllView AS SELECT\* FROM t\_employess;**

**----可调用试图**

**SELECT \* FROM findAllView;**

**索引的作用**

1. **大大的提升了数据库查询效率的性能**
2. **建立索引可以大大的提高检索的数据，以及减少表的检索行数**
3. **在分组和排序字句进行数据检索，可以减少查询时间中分组和排序时所消耗的时间**

**索引的创建:**

**格式:Create index [索引名] on [表名](字段1,字段2);(普通索引)**

**唯一索引: 索引列的值必须唯一sh的，但允许有空值(注意和主键不同)**

**格式:create unique index [索引名] on[表](字段);**

**UUID:唯一标识码**

**存储过程**

**格式:**

**Delimiter $$**

**Create procedure [存储过程名称](in/out/inout 变量名称 类型,(in/out/inout 变量名称 类型,)**

**Delimiter ;(结束存储)**

**调用:**

**Call [存储过程名称]();**

**案例:Delimiter $$**

**Create procedure getsum(in num1 int,in num2 int)**

**Select num1 + num2;**

**Delimiter ;**

**Call getsum(1,2);**

**----求和结果并返回**

**Delimiter $$**

**Create procedure getsum1(in num1 int,in num2 int，out sum1 int)**

**Select num1 + num2;**

**Delimiter ;**

**Call getsum1(2,3);**

**----插入一行数据**

**Delimiter $$**

**Create procedure insertdata(in num1 varchar(20),in num2 vaarchar(20),in score int)**

**Insert into [表名] values(num1,num2,score);**

**Delimiter ;**

**Call insertdata(“人名”,”课程”,分数)**

**Insert与存储过程插入数据的效率的区别:**

**Insert:每一次数据库都要进行编译**

**存储过程:写一次，数据库编译完成之后，会保存到数据库，后期调用的时候，直接将模板进行替换就行了**

**存储过程效率比较高**