第1章：mysql数据库的介绍以及课程大纲总览

简介：介绍什么是mysql以及整套课程的大纲知识点

1.1 mysql的简介介绍

简介：mysql的简介

- 1、什么是mysql？

> mysql是一个开源的关系型数据库管理系统，现在是oracle公司旗下的一款产品，由C和C++语言编写，可移植性高。支持在多种操作系统上安装，最常见有AIX，linux，window。mysql因为开源免费，所以受到了目前互联网行业的欢迎。
>
> 以mysql作为数据库，linux系统作为操作系统，apache或者nginx作为web服务器，perl/php/python作为服务端的脚本解释器，就可以搭建起一个免费的网站。被业界称为LNMP或者LAMP

- 2、适合人群：

> 适合零基础，运维人员，程序开发人员，DBA数据库管理员等等所有从事IT行业的人。

- 3、学后达到的水平：

> 入门偏中等水平

1.2 课程大纲总览

简介：介绍课程知识目录大纲

小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第2章：mysql数据库面试必备入门基础知识

简介：主讲面试时，面试官必问的知识点，主要关于版本号，服务进程，有哪些sql操作语句等

2.1 mysql各个版本的重要性介绍

简介：各个版本的区别 官网：https://dev.mysql.com/downloads/mysql/

- MySQL Community Server 社区版本，开源免费，但不提供官方技术支持。
- MySQL Enterprise Edition 企业版本，需付费，购买了之后可以电话支持
- MySQL Cluster 集群版，开源免费。可将几个MySQL Server封装成一个Server。

2.2 mysql的核心知识之服务管理

简介：mysql的service服务管理与登录管理

查看mysql服务进程：ps -ef | grep mysql

service服务管理：cp -a mysql.server /etc/rc.d/init.d/mysql

启动命令：service mysql start
关闭命令：service mysql stop
重新启动命令：service mysql restart
查看状态命令：service mysql status

登录管理： ln -s /usr/local/mysql/bin/* /bin
登录命令：mysql -uroot -p

默认端口号：3306
配置文件：/etc/my.cnf

2.3 mysql的可视化图形界面与命令行操作

简介：命令行的登录与图形化界面的登录

命令行模式：

    登录命令：mysql -u用户 -p密码

    退出命令：exit; quit;

图形化模式：
    ip地址或者主机名：120.76.62.13

2.4 mysql的库表深入解析

简介：mysql数据库的基本概念详解

- 什么是库？

顾名思义就是数据仓库的意思，存储着一定数据结构的数据，一个数据库中可能包含着若干个表，我们可以通过数据库提供的多种方法来管理数据库里边的数据。本质上mysql数据库是一个关系型数据服务管理系统

- 什么是表？

我们所说的表就是数据表，每一张表是由行和列组成，每记录一条数据，数据表就增加一行。列是由字段名与字段数据属性组成，我们称之列为字段，每一个字段有着多个属性。例如是否允许为空、长度、类型等等

```
数据库：database
数据表：table
字段（列）：column
行：row
```

2.5 mysql的sql各类语句精讲

简介：mysql的操作语句分类

- 操作语句分为四类：

1. DDL 数据定义语言 (Data Definition Language) 例如：建库，建表
2. DML 数据操纵语言(Data Manipulation Language) 例如：对表中的数据进行增删改操作
3. DQL 数据查询语言(Data Query Language) 例如：对数据进行查询
4. DCL 数据控制语言(Data Control Language) 例如：对用户的权限进行设置

第3章：mysql核心知识之DDL数据定义语言

简介：对DDl数据定义语言进行深入学习

3.1 mysql数据库必备知识之创建、查看以及使用/切换

简介：细讲数据库的创建使用

- 直接创建数据库 db1

```
create database db1;
```

- 查看当前在哪个库里边

```
select database();
```

- 进入库的操作

```
use 库名;
```

- 判断是否存在，如果不存在则创建数据库 db2

```
create database if not exists db2;
```

- 创建数据库并指定字符集为 gbk

```
create database db3 default character set gbk;
```

- 查看某个库是什么字符集；

```
show create database XD;
```

- 查看当前mysql使用的字符集

```
show variables like 'character%';
```

3.2 mysql创建表之常用数据类型

简介：细讲常用数据类型

- 数据类型是什么？

> 数据类型是指列、存储过程参数、表达式和局部变量的数据特征，它决定了数据的存储格式，代表了不同的信息类型。
> 有一些数据是要存储为数字的，数字当中有些是要存储为整数、小数、日期型等...

- mysql常见数据类型

```
<1>整数型
     类型       大小       范围（有符号）                  范围（无符号unsigned）     用途
     TINYINT    1 字节     (-128，127)                     (0，255)                   小整数值
     SMALLINT   2 字节     (-32768，32767)                 (0，65535)                 大整数值
     MEDIUMINT  3 字节     (-8388608，8388607)             (0，16777215)              大整数值
     INT        4 字节     (-2147483648，2147483647)       (0，4294967295)            大整数值
     BIGINT     8 字节       ( )                           (0，2的64次方减1)          极大整数值

<2>浮点型
 FLOAT(m,d)   4 字节     单精度浮点型   备注：m代表总个数，d代表小数位个数
 DOUBLE(m,d)  8 字节     双精度浮点型   备注：m代表总个数，d代表小数位个数

 <3>定点型
 DECIMAL(m,d)      依赖于M和D的值      备注：m代表总个数，d代表小数位个数

 <4>字符串类型
 类型             大小                  用途
 CHAR             0-255字节             定长字符串
 VARCHAR          0-65535字节           变长字符串
 TINYTEXT         0-255字节             短文本字符串
 TEXT             0-65535字节           长文本数据
 MEDIUMTEXT       0-16777215字节        中等长度文本数据
 LONGTEXT         0-4294967295字节      极大文本数据

 char的优缺点：存取速度比varchar更快，但是比varchar更占用空间
```

```
varchar的优缺点：比char省空间。但是存取速度没有char快

<5>时间型
数据类型     字节数            格式                      备注
date         3                 yyyy-MM-dd                存储日期值
time         3                 HH:mm:ss                  存储时分秒
year         1                 yyyy                      存储年
datetime     8                 yyyy-MM-dd HH:mm:ss       存储日期+时间
timestamp    4                 yyyy-MM-dd HH:mm:ss       存储日期+时间，可作时间戳
```

```
create table test_time (
            date_value date,
            time_value time,
            year_value year,
            datetime_value datetime,
            timestamp_value timestamp
        ) engine=innodb charset=utf8;
```

```
insert into test_time values(now(), now(), now(), now(), now());
```

3.3 mysql数据表必备知识之创建表

简介：讲解表是怎么来创建的，以及常见约束条件举例说明

- 语法：

```
CREATE TABLE 表名 (
                  字段名1 字段类型1 约束条件1 说明1,
                  字段名2 字段类型2 约束条件2 说明2,
                  字段名3 字段类型3 约束条件3 说明3
                  );
```

```
create table 新表名 as select * from 旧表名 where 1=2;(注意：建议这种创建表的方式用于日常测试，因　为可能索引什么的会复制不过来)
```

```
create table 新表名 like 旧表名;
```

- 约束条件：

```
comment          ----说明解释
not null         ----不为空
default          ----默认值
unsigned         ----无符号（即正数）
auto_increment   ----自增
zerofill         ----自动填充
unique key       ----唯一值
```

- 创建sql

```
CREATE TABLE student (
                    id tinyint(5) zerofill auto_increment  not null comment '学生学号',
                    name varchar(20) default null comment '学生姓名',
                    age  tinyint  default null comment '学生年龄',
                    class varchar(20) default null comment '学生班级',
                    sex char(5) not null comment '学生性别',
                    unique key (id)
                    )engine=innodb charset=utf8;;

CREATE TABLE student (
                    id tinyint(5)  auto_increment  default null comment '学生学号',
                    name varchar(20) default null comment '学生姓名',
                    age  tinyint  default null comment '学生年龄',
                    class varchar(20) default null comment '学生班级',
                    sex char(5) not null comment '学生性别',
                    unique key (id)
                    )engine=innodb charset=utf8;;
```

3.4 mysql数据表必备知识之查看

简介：如何查看表的基本结构信息

- 查看数据库中的所有表：show tables；
- 查看表结构：desc 表名;
- 查看创建表的sql语句：show create table 表名;
- \G :有结束sql语句的作用，还有把显示的数据纵向旋转90度
- \g :有结束sql语句的作用

3.5 mysql数据表必备知识之表结构维护与删除

简介：细讲核心知识表结构的修改

- 修改表名

```
 rename table 旧表名 to 新表名;
 rename table student to user;
```

- 添加列

```
给表添加一列：alter table 表名 add 列名 类型;
alter table user add addr varchar(50);

alter table add 列名 类型 comment '说明';
alter table user add famliy varchar(50) comment '学生父母';

给表最前面添加一列：alter table 表名 add 列名 类型 first;
alter table user add job varchar(10) first;

给表某个字段后添加一列：alter table 表名 add 列名 类型 after 字段名;
alter table user add servnumber int(11)  after id;

注意：没有给表某个字段前添加一列的说法。
```

- 修改列类型

```
alter table 表名 modify 列名 新类型;
alter table user modify servnumber varchar(20);
```

- 修改列名

```
alter table 表名 change 旧列名 新列名 类型;
alter table user change servnumber telephone varchar(20);
```

- 删除列

```
alter table 表名 drop 列名;
alter table user drop famliy;
```

- 修改字符集

```
alter table 表名 character set 字符集;
alter table user character  set GBK;
```

- mysql表的删除

```
drop table 表名;
drop table user;
```

```
看表是否存在，若存在则删除表：drop table if exists 表名;
drop table  if exists teacher;
```

小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第4章：mysql核心知识之DML数据操纵语言

简介：细讲核心知识对数据表的数据进行增删改

4.1 mysql深入剖析表数据新增

简介：讲解表数据新增的多种例子

- 普通的插入表数据

```
insert into 表名(字段名) values(字段对应值);

insert into employee (empno,ename,job,mgr,hiredate,sal,deptnu) values ('1000','小明','经
理','10001','2019-03-03','12345.23','10');

insert into 表名 values(所有字段对应值);
insert into employee  values ('1001','小明','经理','10001','2019-03-03','12345.23','10');
```

- 蠕虫复制(将一张表的数据复制到另一张表中)

```
insert into 表名1 select * from 表名2;

insert into 表名1(字段名1,字段名2) select 字段名1,字段名2 from 表名2;

insert into emp (empno,ename) select empno,ename from employee;
```

- 建表复制

```
create table 表名1 as select 字段名1,字段名2 from 表名2;

create table emp as select empno ,ename from employee;
```

- 一次性插入多个数据

```
 insert into 表名  (字段名) values (对应值1),(对应值2),(对应值3);
```

- 创建sql:

```
某个公司的员工表
CREATE TABLE employee(
    empno       INT  PRIMARY KEY comment '雇员编号',
    ename       VARCHAR(20) comment '雇员姓名',
    job         VARCHAR(20) comment '雇员职位',
    mgr         INT comment '雇员上级编号',
    hiredate    DATE comment '雇佣日期',
    sal         DECIMAL(7,2) comment '薪资',
    deptnu      INT comment '部门编号'
    );
```

4.2 mysql深入剖析表数据的修改以及删除

简介:讲解如何对表数据进行修改删除以及注意事项

- 修改(更新):

```
update 表名 set 字段名1=值1 where 字段名=值;

update 表名 set 字段名1=值1,字段名2=值2 where 字段名=值;
```

- 删除：

```
delete from 表名 where 字段名=值;

 truncate table 表名;
 delete from 表名;
 drop table 表名;
```

- 注意事项：

面试时：面试官问在删改数据之前，你会怎么做？
答案：会对数据进行备份操作，以防万一，可以进行数据回退

面试时：面试官会问，delete与truncate与drop 这三种删除数据的共同点都是删除数据，他们的不同点是什么?
delele 会把删除的操作记录给记录起来，以便数据回退，不会释放空间，而且不会删除定义。
truncate不会记录删除操作，会把表占用的空间恢复到最初，不会删除定义
drop会删除整张表，释放表占用的空间。

- 删除速度：

```
drop > truncate > delete
```

4.3 mysql核心知识之中文乱码问题

简介：详细讲解汉字显示乱码问题

- 查看当前mysql使用的字符集：show variables like 'character%';

```
mysql> show variables like 'character%';
+--------------------------+----------------------------------+
| Variable_name            | Value                            |
+--------------------------+----------------------------------+
| character_set_client     | utf8                             |
| character_set_connection | utf8                             |
| character_set_database   | utf8                             |
| character_set_filesystem | binary                           |
| character_set_results    | utf8                             |
| character_set_server     | utf8                             |
| character_set_system     | utf8                             |
| character_sets_dir       | /usr/local/mysql/share/charsets/ |
+--------------------------+----------------------------------+
```

- character_set_client：客户端请求数据的字符集
- character_set_connection：客户端与服务器连接的字符集
- character_set_database：数据库服务器中某个库使用的字符集设定，如果建库时没有指明，将默认使用配置上的字符集
- character_set_results：返回给客户端的字符集(从数据库读取到的数据是什么编码的)
- character_set_server：为服务器安装时指定的默认字符集设定。
- character_set_system：系统字符集(修改不了的，就是utf8)
- character_sets_dir：mysql字符集文件的保存路径

- 临时：set names gbk;
- 永久：修改配置文件my.cnf里边的

```
[client]
default-character-set=gbk
作用于外部的显示

[mysqld]
character_set_server=gbk
作用于内部，会作用于创建库表时默认字符集
```

- 修改库的字符集编码

```
alter database xiaoxiao default character set gbk;
```

- 修改表的字符集编码

```
alter table employee default character set utf8;
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第5章：mysql核心知识之DQL数据查询语言与项目高级查询实战

简介：细讲对数据表中的数据进行各种查询，以及项目实战查询

```
/*创建部门表*/
CREATE TABLE dept(
    deptnu      INT  PRIMARY KEY comment '部门编号',
    dname       VARCHAR(50) comment '部门名称',
    addr        VARCHAR(50) comment '部门地址'
);

某个公司的员工表
CREATE TABLE employee(
    empno       INT  PRIMARY KEY comment '雇员编号',
    ename       VARCHAR(50) comment '雇员姓名',
    job         VARCHAR(50) comment '雇员职位',
    mgr         INT comment '雇员上级编号',
    hiredate    DATE comment '雇佣日期',
    sal         DECIMAL(7,2) comment '薪资',
    deptnu      INT comment '部门编号'
)ENGINE=MyISAM DEFAULT CHARSET=utf8;

/*创建工资等级表*/
CREATE TABLE salgrade(
```

```
    grade       INT  PRIMARY KEY comment '等级',
    lowsal      INT comment '最低薪资',
    higsal      INT comment '最高薪资'
);

/*插入dept表数据*/
INSERT INTO dept VALUES (10, '研发部', '北京');
INSERT INTO dept VALUES (20, '工程部', '上海');
INSERT INTO dept VALUES (30, '销售部', '广州');
INSERT INTO dept VALUES (40, '财务部', '深圳');

/*插入emp表数据*/
INSERT INTO employee VALUES (1009, '唐僧', '董事长', NULL, '2010-11-17', 50000,  10);
INSERT INTO employee VALUES (1004, '猪八戒', '经理', 1009, '2001-04-02', 29750, 20);
INSERT INTO employee VALUES (1006, '猴子', '经理', 1009, '2011-05-01', 28500, 30);
INSERT INTO employee VALUES (1007, '张飞', '经理', 1009, '2011-09-01', 24500,10);
INSERT INTO employee VALUES (1008, '诸葛亮', '分析师', 1004, '2017-04-19', 30000, 20);
INSERT INTO employee VALUES (1013, '林俊杰', '分析师', 1004, '2011-12-03', 30000, 20);
INSERT INTO employee VALUES (1002, '牛魔王', '销售员', 1006, '2018-02-20', 16000, 30);
INSERT INTO employee VALUES (1003, '程咬金', '销售员', 1006, '2017-02-22', 12500, 30);
INSERT INTO employee VALUES (1005, '后裔', '销售员', 1006, '2011-09-28', 12500, 30);
INSERT INTO employee VALUES (1010, '韩信', '销售员', 1006, '2018-09-08', 15000,30);
INSERT INTO employee VALUES (1012, '安琪拉', '文员', 1006, '2011-12-03', 9500,  30);
INSERT INTO employee VALUES (1014, '甄姬', '文员', 1007, '2019-01-23', 7500, 10);
INSERT INTO employee VALUES (1011, '妲己', '文员', 1008, '2018-05-23', 11000, 20);
INSERT INTO employee VALUES (1001, '小乔', '文员', 1013, '2018-12-17', 8000, 20);

/*插入salgrade表数据*/
INSERT INTO salgrade VALUES (1, 7000, 12000);
INSERT INTO salgrade VALUES (2, 12010, 14000);
INSERT INTO salgrade VALUES (3, 14010, 20000);
INSERT INTO salgrade VALUES (4, 20010, 30000);
INSERT INTO salgrade VALUES (5, 30010, 99990);
```

5.1 mysql查询子句之一where条件查询

简介：详解where条件下的各种查询

- 简单查询

```
select * from employee;
select empno,ename,job as ename_job from employee;
```

- 精确条件查询

```
select * from employee where ename='后裔';
select * from employee where sal != 50000;
select * from employee where sal <> 50000;
select * from employee where sal > 10000;
```

- 模糊条件查询

```
show variables like '%aracter%';
select * from employee  where ename like '林%';
```

- 范围查询

```
select * from employee where sal between 10000 and 30000;
select * from employee where hiredate between '2011-01-01' and '2017-12-1';
```

- 离散查询

```
select * from employee where ename in ('猴子','林俊杰','小红','小胡');
```

- 清除重复值

```
select distinct(job) from employee;
```

- 统计查询（聚合函数）：

```
        count(code)或者count(*)
         select count(*) from employee;
         select count(ename) from employee;

        sum()  计算总和
         select sum(sal) from employee;

        max()    计算最大值
         select * from employee where sal= (select  max(sal) from employee);

        avg()   计算平均值
         select avg(sal) from employee;

        min()   计算最低值
         select * from employee where sal= (select  min(sal) from employee);

        concat函数： 起到连接作用
         select concat(ename,' 是 ',job) as aaaa from employee;
```

5.2 mysql查询子句之二group by分组查询（分组）

简介：详解group by的用法以及应用场景

- 作用：把行 按 字段 分组
- 语法：group by 列1，列2....列N
- 适用场合：常用于统计场合，一般和聚合函数连用

```
eg:
     select deptnu,count(*) from employee group by deptnu;
     select deptnu,job,count(*) from employee group by deptnu,job;
     select job,count(*) from employee group by job;
```

5.3 mysql查询子句之三having条件查询（筛选）

简介：详解having的用法以及应用场景

- 作用：对查询的结果进行筛选操作
- 语法：having 条件 或者 having 聚合函数 条件
- 适用场合：一般跟在group by之后

```
eg:
    select job,count(*) from employee group by job having job ='文员';
    select  deptnu,job,count(*) from employee group by deptnu,job having count(*)>=2;
    select  deptnu,job,count(*) as 总数 from employee group by deptnu,job having 总数>=2;
```

5.4 mysql查询子句之四order by排序查询（排序）

简介：详解order by的用法以及应用场景

- 作用：对查询的结果进行排序操作
- 语法：order by 字段1,字段2 .....
- 适用场合：一般用在查询结果的排序

```
eg:
     select * from employee order by sal;
     select * from employee order by hiredate;
     select  deptnu,job,count(*) as 总数 from employee group by deptnu,job having 总数>=2
order by deptnu desc;
     select  deptnu,job,count(*) as 总数 from employee group by deptnu,job having 总数>=2
order by deptnu asc;
     select  deptnu,job,count(*) as 总数 from employee group by deptnu,job having 总数>=2
order by deptnu;

     顺序：where ---- group by ----- having ------ order by
```

5.5 mysql查询子句之五limit限制查询（限制）

简介：详解limit的用法以及应用场景

- 作用：对查询结果起到限制条数的作用
- 语法：limit n，m n:代表起始条数值，不写默认为0；m代表：取出的条数
- 适用场合：数据量过多时，可以起到限制作用

```
eg:
    select * from XD.employee limit 4,5;
```

5.6 mysql查询之exists型子查询

简介：详解exists的用法

- exists型子查询后面是一个受限的select查询语句
- exists子查询，如果exists后的内层查询能查出数据，则返回 TRUE 表示存在；为空则返回 FLASE则不存在。

```
分为俩种：exists跟 not exists

select 1 from employee where 1=1;
select * from 表名 a where exists (select 1 from 表名2 where 条件);

eg:查询出公司有员工的部门的详细信息
select * from dept a where exists (select 1 from employee b where a.deptnu=b.deptnu);
select * from dept a where not exists (select 1 from employee b where a.deptnu=b.deptnu);
```

5.7 mysql查询之左连接查询与右连接查询

简介：详解左右连接的用法以及应用场景

- 左连接称之为左外连接 右连接称之为右外连接 这俩个连接都是属于外连接
- 左连接关键字：left join 表名 on 条件 / left outer 表名 join on 条件 右连接关键字：right join 表名 on 条件/ right outer 表名 join on 条件
- 左连接说明： left join 是left outer join的简写，左(外)连接，左表(a_table)的记录将会全部表示出来， 而右表(b_table)只会显示符合搜索条件的记录。右表记录不足的地方均为NULL。
- 右连接说明：right join是right outer join的简写，与左(外)连接相反，右(外)连接，左表(a_table)只会显示符合搜索条件的记录，而右表(b_table)的记录将会全部表示出来。左表记录不足的地方均为NULL。

```
eg:列出部门名称和这些部门的员工信息，同时列出那些没有的员工的部门
        dept，employee
        select a.dname,b.* from dept a  left join employee b on a.deptnu=b.deptnu;
        select b.dname,a.* from employee a  right join  dept b on b.deptnu=a.deptnu;
```

5.8 mysql查询之内连接查询与联合查询

简介：详解内连接与联合查询的用法以及应用场景

- 内连接：获取两个表中字段匹配关系的记录
- 主要语法：INNER JOIN 表名 ON 条件;

```
eg:想查出员工张飞的所在部门的地址
    select a.addr  from dept a inner join employee b on a.deptnu=b.deptnu and b.ename='张
飞';
    select a.addr from dept a,employee b where a.deptnu=b.deptnu and b.ename='张飞';
```

- 联合查询：就是把多个查询语句的查询结果结合在一起

主要语法1：... UNION ...（去除重复） 主要语法2：... UNION ALL ...（不去重复）

- union查询的注意事项：

```
(1)两个select语句的查询结果的“字段数”必须一致；

(2)通常，也应该让两个查询语句的字段类型具有一致性；

(3)也可以联合更多的查询结果；

(4)用到order by排序时，需要加上limit（加上最大条数就行），需要对子句用括号括起来

eg:对销售员的工资从低到高排序，而文员的工资从高到低排序
    (select * from employee a where a.job = '销售员'  order by a.sal limit 999999 )
union  (select * from employee b where b.job = '文员' order by b.sal desc limit
999999);
```

5.9 mysql查询之项目高级查询实战(一)

简介：高级查询实战

- 查出至少有一个员工的部门。显示部门编号、部门名称、部门位置、部门人数。

```
    涉及表： employee dept
    语句：select deptnu,count(*) from employee group by deptnu
    语句：select a.deptnu,a.dname,a.addr, b.zongshu from dept a,(select deptnu,count(*) as
zongshu from employee group by deptnu) b where a.deptnu=b.deptnu;
```

- 列出薪金比安琪拉高的所有员工。

```
    涉及表：employee
    语句：select * from  employee where sal > (select sal from employee where ename='安琪
拉');
```

- 列出所有员工的姓名及其直接上级的姓名。

```
    涉及表：employee
    语句：select a.ename,ifnull(b.ename,'BOSS') as leader from employee a left join employee
b on a.mgr=b.empno;
```

- 列出受雇日期早于直接上级的所有员工的编号、姓名、部门名称。

```
    涉及表：employee dept
    条件：a.hiredate < b.hiredate
    语句：select a.empno,a.ename,c.dname from employee a left join employee b on
a.mgr=b.empno left join dept c on a.deptnu=c.deptnu where a.hiredate < b.hiredate;
```

- 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。

```
    涉及表：dept employee
    语句：select a.dname,b.* from dept a left join employee b on a.deptnu=b.deptnu;
```

- 列出所有文员的姓名及其部门名称，所在部门的总人数。

```
    涉及表：employee dept
    条件：job='文员'
    语句：select deptnu,count(*) as zongshu from employee group by deptnu;
    语句：select b.ename,a.dname,b.job,c.zongshu from dept a ,employee b ,(select
deptnu,count(*) as zongshu from employee group by deptnu) c where a.deptnu=b.deptnu and
b.job='文员' and b.deptnu=c.deptnu;
```

5.10 mysql查询之项目高级查询实战(二)

简介：高级查询实战续

- 列出最低薪金大于15000的各种工作及从事此工作的员工人数。

```
    涉及表：employee
    条件：min(sal) > 15000
    语句：select job,count(*) from employee group by job having   min(sal) > 15000;
```

- 列出在销售部工作的员工的姓名，假定不知道销售部的部门编号。

```
   涉及表：employee dept
   select  ename  from employee where deptnu=(select deptnu from dept where dname='销售部');
```

- 列出与诸葛亮从事相同工作的所有员工及部门名称。

```
    涉及表：employee dept
    语句：select a.ename,b.dname from employee a,dept b where a.deptnu=b.deptnu and a.job=
             (select job from employee where ename='诸葛亮');
    语句：select a.ename,b.dname from employee a left join dept b on a.deptnu=b.deptnu where
              a.job=(select job from employee where ename='诸葛亮');
```

- 列出薪金比 在部门30工作的员工的薪金 还高的员工姓名和薪金、部门名称。

```
    涉及表：employee dept
    语句：select a.ename,a.sal,b.dname from employee a ,dept b where a.deptnu=b.deptnu and
sal >          (select max(sal) from employee where deptnu=30);
```

- 列出每个部门的员工数量、平均工资。

```
    涉及表：employee
    语句：select deptnu , count(*) ,avg (sal) from employee  group by deptnu;
```

- 列出薪金高于公司平均薪金的所有员工信息，所在部门名称，上级领导，工资等级。

```
    涉及表：employee dept salgrade
    条件：select avg(sal) from employee
    语句：elect a.*,c.dname,b.ename,d.grade from employee a,employee b,dept c ,salgrade d
where           a.mgr=b.empno and a.deptnu =c.deptnu and a.sal > (select avg(sal) from
employee)  and           a.sal  between d.lowsal and d.higsal;
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第6章：mysql核心知识之DCL数据控制语言（对用户权限的设置）

简介：精讲数据控制语言如何限制用户的各种权限

- 什么是DCL数据控制语言？

  数据控制语言（DCL：Data Control Language）是用来设置或者更改数据库用户或角色权限的语句，这些语句包括GRANT、DENY、REVOKE等语句

6.1 mysql限制root用户指定ip登录

简介：详解如何从安全角度出发限制root用户指定ip登录

- 查看root用户可以在哪台机器登录

```
select user,host from mysql.user where user='root';
```

- 修改mysql库里边的user表

```
 update mysql.user set host='localhost' where user='root';
```

- 刷新权限

```
flush privileges;
```

6.2 mysql必备知识之用户密码

简介：详解忘记密码以及如何修改用户密码

- 修改用户密码分三种方法：

1. 第一种：set password for 用户@ip = password('密码');

   ```
   set password for root@localhost = password('root');
   ```

2. 第二种：mysqladmin -u用户 -p旧密码 password 新密码;

```
mysqladmin -urootmysqladmin -uroot -proot password;
```

3. 第三种：update mysql.user set authentication_string=password('密码') where user='用户' and host='ip';

```
update mysql.user set authentication_string=password('root') where user='root' and
host='localhost';
```

- 忘记密码:

1. 第一步：修改配置文件my.cnf (默认在/etc/my.cnf)，在[mysqld]下面加上 skip-grant-tables （跳过权限的意思）
2. 第二步：重启mysql服务
3. 第三步：mysql -uroot -p 无需密码登录进入
4. 第四步：修改密码

6.3 mysql实战系列之创建新用户并限制ip网段登录

简介：讲解如何创建新用户与删除用户并限制ip登录

- 创建用户的语法：create user 'username'@'host' identified by 'password';

username：你将创建的用户名

host：指定该用户在哪个主机上可以登陆，如果是本地用户可用localhost，如果想让该用户可以从任意远程主机 登陆，可以使用通配符%

password：该用户的登陆密码，密码可以为空，如果为空则该用户可以不需要密码登陆服务器

- 创建用户语法：

创建一个pig用户，并指定登录密码：123456，可以在任何一台远程主机都可以登录

```
create user 'pig'@'%' identified by '123456';
```

创建一个pig用户，并指定登录密码：为空，指定在120网段的机器登录

```
create user 'pig'@'120.%.%.%' identified by '';
```

- 查看权限：

```
select * from mysql.user where user='pig'\G
 mysql> show grants for 'pig'@'%';
 +---------------------------------+
 | Grants for pig@%                |
 +---------------------------------+
 | GRANT USAGE ON *.* TO 'pig'@'%' |
 +---------------------------------+
 USAGE：无权限的意思
 mysql> show grants for 'root'@'localhost';
 +---------------------------------------------------------------------+
```

```
| Grants for root@localhost                                                 |
+---------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------------+
WITH GRANT OPTION:表示这个用户拥有grant权限，即可以对其他用户授权
```

- 删除用户语法：drop user 'username'@'host';

```
drop user 'pig'@'%';
delete from mysql.user where user='pig';
```

6.4 mysql实战系列之库表权限授权与回收

简介：讲解如何限制用户对库表的增删改查权限

- 授权语法：grant 权限1,权限2..... on 数据库对象 to '用户'

```
grant 权限1,权限2..... on 数据库对象 to '用户'@'host' identified by 'password';
```

- all privileges:代表所有权限
- . :代表所有库所有表

```
对现有用户进行授权：对现有用户pig授予所有库所有表所有权限。
grant all privileges on *.*  to 'pig';
```

```
对没有的用户进行授权：创建一个新用户dog授予XD库的所有权限，登录密码123456，任何一台主机登录
grant all privileges on XD.* to 'dog'@'%' identified by '123456';
```

```
对没有的用户进行授权：创建一个新用户cat授予XD库的employee表 查与修改权限，登录密码123456，任何一台主机登录
grant select,update on XD.employee to 'cat'@'%' identified by '123456'
```

```
对没有的用户进行授权：对用户cat授予XD库的employee表insert 权限，登录密码123456，任何一台主机登录
grant insert on XD.employee to 'cat'@'%' identified by '123456';
```

- 回收语法：revoke 权限1,权限2..... on 数据库对象 from '用户'@'host';

```
回收pig用户的所有权限（注意：并没有回收它的登录权限）
revoke all privileges on *.*  from 'pig' @ '%';
flush privileges;
```

```
回收pig用户的所有权限（并回收它的登录权限）
delete from mysql.user where user='pig';
flush privileges;
```

```
回收cat用户对XD库的employee的查与修改权限
revoke select,update on XD.employee from 'cat'@'%';
flush privileges;
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第7章：mysql的核心知识之事务实战，视图，触发器，以及存储过程

简介：主要分5节课来详细讲解事务以及视图优缺点，触发器以及存储过程介绍

7.1 mysql 必备核心知识之事务的详细解析

简介：深入详解事务

- 什么是事务?

> 答：数据库事务通常指对数据库进行读或写的一个操作过程。有两个目的，第一个是为数据库操作提供了一个从失败中恢复到正常状态的方法，同时提供了数据库即使在异常状态下仍能保持一致性的方法；第二个是当多个应用程序在并发访问数据库时，可以在这些应用程序之间提供一个隔离方法，以防止彼此的操作互相干扰。

- 事务的特性（ACID）：

- [ ] 原子性(Atomicity)：事务必须是原子工作单元，一个事务中的所有语句，应该做到：要么全做，要么一个都不做；
- [ ] 一致性(Consistency):让数据保持逻辑上的“合理性”，比如：小明给小红打10000块钱，既要让小明的账户减少10000，又要让小红的账户上增加10000块钱；
- [ ] 隔离性(Isolation)：如果多个事务同时并发执行，但每个事务就像各自独立执行一样。
- [ ] 持久性(Durability)：一个事务执行成功，则对数据来说应该是一个明确的硬盘数据更改（而不仅仅是内存中的变化）。

- 你要使用事务的话，表的引擎要为innodb引擎

7.2 mysql 必备核心知识之事务实战

简介：主要讲解事务的开启以及事务实战，深入了解什么是事务

- 事务的开启与提交：

- [ ] 事务的开启：begin; start transaction;
- [ ] 事务的提交：commit;
- [ ] 事务的回滚：rollback;

创建一个账户表模拟转账

```
create table account (
                        id tinyint(5) zerofill auto_increment  not null comment 'id编号',
                        name varchar(20) default null comment '客户姓名',
                        money decimal(10,2) not null comment '账户金额',
                        primary key (id)
                        )engine=innodb charset=utf8;
```

- 开启autocommit（临时生效）：

  OFF（0）：表示关闭 ON （1）：表示开启

```
mysql> set autocommit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like 'autocommit';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | OFF   |
+---------------+-------+
mysql> set autocommit=1;
Query OK, 0 rows affected (0.00 sec)
mysql>
mysql> show variables like 'autocommit';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |
```

- 开启autocommit（永久生效）：

  修改配置文件：vi /etc/my.cnf 在[mysqld]下面加上：autocommit=1 记得重启服务才会生效

7.3 mysql核心知识之视图的应用

简介：详细讲解视图的优缺点以及应用

- 什么是视图？视图的作用是什么？

视图（view）是一种虚拟存在的表，是一个逻辑表，它本身是不包含数据的。作为一个select语句保存在数据字典中的。通过视图，可以展现基表（用来创建视图的表叫做基表base table）的部分数据，说白了视图的数据就是来自于基表

- 视图的优点是：

1）简单：使用视图的用户完全不需要关心后面对应的表的结构、关联条件和筛选条件，对用户来说已经是过滤好的复合条件的结果集。

2）安全：使用视图的用户只能访问他们被允许查询的结果集，对表的权限管理并不能限制到某个行某个列，但是通过视图就可以简单的实现。

3）数据独立：一旦视图的结构确定了，可以屏蔽表结构变化对用户的影响，源表增加列对视图没有影响;源表修改列名，则可以通过修改视图来解决，不会造成对访问者的影响。

4）不占用空间：视图是逻辑上的表，不占用内存空间

总而言之，使用视图的大部分情况是为了保障数据安全性，提高查询效率。

- 视图的创建以及修改

创建的基本语法是：

```
    create view <视图名称> as select 语句;
    create view <视图名称>（字段）as select 语句;
    create or replace view <视图名称>;
```

修改的语法是：

```
alter view <视图名称> as select 语句;
```

视图删除语法：

```
drop view <视图名称> ;
```

- 视图的缺点

1)性能差：sql server必须把视图查询转化成对基本表的查询，如果这个视图是由一个复杂的多表查询所定义，那么，即使是视图的一个简单查询，sql server也要把它变成一个复杂的结合体，需要花费一定的时间。

2)修改限制：当用户试图修改试图的某些信息时，数据库必须把它转化为对基本表的某些信息的修改，对于简单的试图来说，这是很方便的，但是，对于比较复杂的试图，可能是不可修改的。

7.4 mysql 的触发器介绍

简介:介绍什么是触发器以及如何来创建

- 什么是触发器？

触发器就是监视某种情况，并触发某种操作

- 创建触发器的语法:

```
create trigger 触发器名称  after/before   insert/update/delete on 表名
        for each row
        begin
        sql语句;
        end
```

```
after/before:可以设置为事件发生前或后
insert/update/delete:它们可以在执行insert、update或delete的过程中触发
for each row:每隔一行执行一次动作
```

- 删除触发器的语法：

  ```
  drop trigger 触发器名称;
  ```

- 演示：

```
创建一个员工迟到表：
 create table work_time_delay(
            empno int not null comment '雇员编号',
            ename varchar(50) comment '雇员姓名',
            status int comment '状态'
            );
```

```
delimiter // 自定义语句的结束符号

    mysql> delimiter //
    mysql>
    mysql> create trigger trig_work after insert on work_time_delay
        -> for each row
        -> begin
        -> update employee set sal=sal-100 where empno=new.empno;
        -> end
        -> //
    Query OK, 0 rows affected (0.01 sec)

new：指的是事件发生before或者after保存的新数据
```

7.5 mysql的存储过程介绍

简介：介绍什么是存储过程

- 什么是存储过程？

```
存储过程就是把复杂的一系列操作，封装成一个过程。类似于shell，python脚本等。
```

- 存储过程的优缺点

```
    优点是：
        1)复杂操作，调用简单
        2)速度快

    缺点是：
        1）封装复杂
        2） 没有灵活性
```

- 创建存储过程语法：

```
create procedure 名称（参数....）
        begin
         过程体;
         过程体;
         end
```

```
参数：in|out|inout 参数名称 类型（长度）
        in：表示调用者向过程传入值（传入值可以是字面量或变量）
        out：表示过程向调用者传出值(可以返回多个值)（传出值只能是变量）
        inout：既表示调用者向过程传入值，又表示过程向调用者传出值（值只能是变量）
```

- 声明变量：declare 变量名 类型(长度) default 默认值;
- 给变量赋值：set @变量名=值;
- 调用存储命令：call 名称(@变量名);
- 删除存储过程命令：drop procedure 名称;
- 查看创建的存储过程命令：

```
show create procedure 名称\G;
```

```
创建一个简单的存储过程：
    mysql> delimiter //
    mysql> create procedure  name(in n int)
        ->             begin
        ->             select * from employee limit n;
        ->             end
        -> //
    Query OK, 0 rows affected (0.00 sec)

    mysql> set @n=5;
        -> //
    Query OK, 0 rows affected (0.00 sec)

    mysql>
    mysql> call name(@n);
```

```
mysql>          create procedure  name()
    ->              begin
    ->              declare  n int default 6;
    ->              select * from employee limit n;
    ->              end
    -> //
  Query OK, 0 rows affected (0.00 sec)

  mysql> call name();
```

愿景："让编程不在难学，让技术与生活更加有趣"更多教程请访问 xdclass.net

第8章：mysql必备核心知识之索引与存储引擎的介绍

简介：各种索引的概述，以及如何创建索引与引擎的介绍

8.1 mysql的存储引擎介绍

简介：主要来详细介绍MyISAM与InnoDB引擎

（1）什么是数据库存储引擎？

数据库引擎是数据库底层软件组件，不同的存储引擎提供不同的存储机制，索引技巧，锁定水平等功能，使用不同的数据库引擎，可以获得特定的功能

（2）如何查看引擎？

如何查看数据库支持的引擎
```
show engines;
```

查看当前数据的引擎：
```
show create table 表名\G
```

查看当前库所有表的引擎：
```
show table status\G
```

（3）建表时指定引擎

```
create table yingqin (id int,name varchar(20)) engine='InnoDB';
```

（4）修改表的引擎

```
alter table 表名 engine='MyiSAm';

修改默认引擎
•    vi /etc/my.cnf
•    [mysqld]下面
•    default-storage-engine=MyIsAM
•    记得保存后重启服务
```

（5）MyISAM与InnoDB的区别

```
MyISAM：支持全文索引（full text）;不支持事务;表级锁;保存表的具体行数;奔溃恢复不好

Innodb：支持事务;以前的版本是不支持全文索引，但在5.6之后的版本就开始支持这个功能了;行级锁（并非绝对，当执行sql语句时不能确定范围时，也会进行锁全表例如： update table set id=3 where name like 'a%';）;不保存表的具体行数;奔溃恢复好
```

（6）总结：什么时候选择什么引擎比较好

```
MyISAM：
•    一般来说MyISAM不需要用到事务的时候
•    做很多count计算

InnoDB：
•    可靠性要求高的，或者要求支持事务
•    想要用到外键约束的时候（讲外键的时候会讲）

推荐：
•    推荐用InnoDB
```

8.2 mysql必备知识之常用索引精讲

简介：讲解什么是索引，索引的优缺点，以及常见索引有哪些

- 什么是索引？

```
索引是一个单独的，存储在磁盘中上的数据库结构，它们包含着对数据表里的所有记录的引用指针。使用索引可以快速的找出在某列或多列中有特定值的行。
```

- 索引的优点：

```
•    通过创建唯一索引，来保证数据库表中的每一行数据的唯一性。
•    可以加快数据的检索速度。
•    可以保证表数据的完整性与准确性
```

- 索引的缺点：

```
•    索引需要占用物理空间。
•    对表中的数据进行改动时，索引也需要跟着动态维护，降低了数据的维护速度。
```

- 索引的常见类型：

- index：普通索引
- unique：唯一索引
- primary key：主键索引
- foreign key：外键索引
- fulltext: 全文索引
- 组合索引

- 创建表的sql语句：

```
create table test (
    id int(7) zerofill auto_increment not null,
    username varchar(20),
    servnumber varchar(30),
    password varchar(20),
    createtime datetime,
    primary key (id)
)DEFAULT CHARSET=utf8;
```

- 生成百万甚至千万级别表的sql 语句 shell脚本：

```
#!/bin/bash

echo "请输入字段servnumber的值："
read serber
echo "请输入创建sql语句的数量："
read number

# char=`head /dev/urandom | tr -dc 0-9 | head -c 11`

for (( i=0;i<$number;i++ ))
        do
        pass=`head /dev/urandom | tr -dc a-z | head -c 8`
        let serber=serber+1
        echo "insert into test(id,username,servnumber,password,createtime)
values('$i','user${i}','${serber}','$pass',now());" >>sql.txt

    done
```

vi test.sh

执行shell脚本：sh test.sh

进行插数操作：source /home/dazhu/sql.txt

8.3 mysql必备核心知识之普通索引与唯一索引

简介：介绍普通索引与唯一索引

- 什么是普通索引？

普通索引（index）顾名思义就是各类索引中最为普通的索引，主要任务就是提高查询速度。其特点是允许出现相同的索引内容，允许空（null）值

- 什么是唯一索引？

唯一索引：（unique）顾名思义就是不可以出现相同的索引内容，但是可以为空（null）值

- 如何创建普通索引或者唯一索引？
  - 创建表的时候创建

```
create table test (
                        id int(7) zerofill auto_increment not null,
                        username varchar(20),
                        servnumber varchar(30),
                        password varchar(20),
                        createtime datetime,
                        unique (id)
                   )DEFAULT CHARSET=utf8;
```

  - 直接为表添加索引

```
语法：
     alter table 表名 add index 索引名称 (字段名称);
 eg:
     alter table test add unique unique_username (username);

注意：假如没有指定索引名称时，会以默认的字段名为索引名称
```

  - 直接创建索引

```
语法：create index 索引 on 表名 (字段名);
eg：create index index_createtime on test (createtime);
```

- 查看索引

```
语法：show index from 表名\G
eg: show index from test\G
```

- 如何删除索引

```
语法：drop index 索引名称 on 表名;
eg：drop index unique_username on test;
```

语法：alter table 表名 drop index 索引名;
eg：alter table test drop index createtime;

8.4 mysql必备核心知识之主键索引

简介：详细讲解主键索引

- 什么是主键索引？

把主键添加索引就是主键索引，它是一种特殊的唯一索引，不允许有空值，而唯一索引（unique是允许为空值的）。指定为“PRIMARY KEY”

主键：主键是表的某一列，这一列的值是用来标志表中的每一行数据的。
注意：每一张表只能拥有一个主键

- 创建主键：

  1）创建表的时候创建

  2）直接为表添加主键索引
  语法：alter table 表名 add primary key (字段名);
  eg：alter table test add primary key (id);

- 删除主键：

语法：
alter table 表名 drop primary key;
eg：
alter table test drop primary key;

注意：在有自增的情况下，必须先删除自增，才可以删除主键

删除自增：alter table test change id id int(7) unsigned zerofill not null;

8.5 mysql核心知识之全文索引的使用

简介：介绍什么是全文索引以及使用

- 什么是全文索引？

  全文索引是将存储在数据库中的文章或者句子等任意内容信息查找出来的索引，单位是词。全文索引也是目前搜索引擎使用的一种关键技术。指定为 fulltex

- 创建练习表的sql：

```
create table command (
id int(5) unsigned primary key  auto_increment,
name varchar(10),
instruction varchar(60)
)engine=MyISAM;
```

- 插入数据sql：

```
insert into command values('1','ls','list directory contents');
insert into command values('2','wc','print newline, word, and byte counts for each
file');
insert into command values('3','cut','remove sections from each line of files');
insert into command values('4','sort','sort lines of text files');
insert into command values('5','find','search for files in a directory hierarchy');
insert into command values('6','cp','复制文件或者文件夹');
insert into command values('7','top','display Linux processes');
insert into command values('8','mv','修改文件名，移动');
insert into command values('9','停止词','is,not,me,yes,no ...');
```

- 添加全文索引：
  - 创建表的时候创建全文索引
  - 通过alter添加

    ```
    alter table command  add fulltext(instruction);
    ```

- 使用全文索引：

```
select * from 表名 where match  (字段名) against ('检索内容');
select * from command where match(instruction) against ('sections');
```

- 查看匹配度：

```
select * from command where match(instruction) against ('directory');
```

- 停止词：

```
出现频率很高的词，将会使全文索引失效
```

- in boolean mode 模式：

```
in boolean mode：意思是指定全文检索模式为布尔全文检索（简单可以理解为是检索方式）
select * from 表名 where match (字段名) against ('检索内容' in boolean mode);
```

- 注意点：

```
使用通配符*时，只能放在词的后边，不能放前边。
```

- 删除全文索引：

```
alter table command drop index instruction;
```

- 注意点总结：

  - 1、一般情况下创建全文索引的字段数据类型为 char、varchar、text 。其它字段类型不可以
  - 2、全文索引不针对非常频繁的词做索引。比如is，no，not，you，me，yes这些，我们称之为停止词
  - 3、对英文检索时忽略大小写

8.6 mysql核心知识之外键约束剖析

简介：解析什么是外键约束，以及有什么作用

- 什么是外键？

外键就是作用于两个表数据之间的链接的一列或多列，用来保证表与表之间的数据的完整性和准确性。

- 添加外键约束：

语法：foreign key（字段名） references 关联的表名(关联表的字段名)

注意：主键跟外键的字段类型一定要相

create table的方法：

```
CREATE TABLE `employee` (
  `empno` int(11) NOT NULL COMMENT '雇员编号',
  `ename` varchar(50) DEFAULT NULL COMMENT '雇员姓名',
  `job` varchar(30) DEFAULT NULL,
  `mgr` int(11) DEFAULT NULL COMMENT '雇员上级编号',
  `hiredate` date DEFAULT NULL COMMENT '雇佣日期',
  `sal` decimal(7,2) DEFAULT NULL COMMENT '薪资',
  `deptnu` int(11) DEFAULT NULL COMMENT '部门编号',
  PRIMARY KEY (`empno`),
 foreign key (deptnu) references dept(deptnu)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

 alter table的方法：
 alter table employee add foreign key (deptnu) references dept(deptnu);
```

- 删除外键约束：

注意：在干掉外键索引之前必须先把外键约束删除，才能删除索引

```
mysql> alter table employee drop index deptnu;
ERROR 1553 (HY000): Cannot drop index 'deptnu': needed in a foreign key constraint
mysql>
mysql> alter table employee drop foreign key employee_ibfk_1;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> alter table employee drop index deptnu;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- 注意点总结：

```
（1）俩个表，主键跟外键的字段类型一定要相同

（2）要使用外键约束表的引擎一定得是InnoDB引擎，MyISAM是不起作用的

（3）在干掉外键索引之前必须先把外键约束删除，才能删除索引
```

8.7 mysql核心知识之联合索引

简介：详细介绍联合索引

- 什么是联合索引？

```
联合索引又称组合索引或者复合索引，是建立在俩列或者多列以上的索引。
```

- 怎么来创建联合索引？

```
alter table 表名 add index(字段1,字段2,字段3);

alter table test add index(username,servnumber,password);
```

- 怎么删除联合索引？

```
alter table test drop index username;
```

- 为什么要使用联合索引，而不使用多个单列索引？

```
联合索引的效率远远高于单列索引
```

- 联合索引的最左原则
- 注意点总结：

- 索引并非越多越好，过多的索引会增加数据的维护速度还有磁盘空间的浪费。
- 当表的数据量很大的时候，可以考虑建立索引。
- 表中经常查数据的字段，可以考虑建立索引。
- 想要保证表中数据的唯一性，可以考虑建立唯一索引。
- 想要保证俩张表中的数据的完整性跟准确性，可以考虑建立外键约束。
- 经常对多列数据进行查询时，可以考虑建立联合索引。

小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第9章：mysql不得不学的sql语句优化思路

简介：介绍如何对sql查询语句进行优化

9.1 mysql的慢查询日志开启与问题定位

简介：介绍如何开启慢查询日志与问题定位

- 第一步：查看是否已经开启了慢查询日志

```
mysql> show variables like 'slow%';
+---------------------+--------------------------------------+
| Variable_name       | Value                                |
+---------------------+--------------------------------------+
| slow_launch_time    | 2                                    |
| slow_query_log      | OFF                                  |
| slow_query_log_file | /data/mydata/xdclass-public-slow.log |
+---------------------+--------------------------------------+
```

- 第二步：开启慢查询日志

```
set global slow_query_log = on ;

日志路径也可以自定义：

set global slow_query_log_file = '路径';
```

- 第三步：查看慢查询的时间临界值

```
show variables like '%long%';
```

- 第四步：设置慢查询的时间标准

```
set long_query_time=0.4;
```

- 注意：重启mysql服务会让在交互界面设置的慢查询恢复到默认

```
永久生效的设置方法：修改配置文件 vi /etc/my.cnf
[mysqld]
slow_query_log = 1
long_query_time = 0.1
slow_query_log_file =/usr/local/mysql/mysql_slow.log

最后必须重启服务才能生效！
```

9.2 mysql的sql语句执行过程解析

简介：介绍如何开启性能详情

- 第一步：查看性能详情是否开启

```
mysql> show variables like '%profiling%';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| have_profiling         | YES   |
| profiling              | OFF   |
| profiling_history_size | 15    |
+------------------------+-------+
```

- 第二步：开启性能记录功能

```
set profiling = on ;
```

- 第三步：查看性能的记录

```
mysql> show profiles;
+----------+------------+-----------------------------------------------------+
| Query_ID | Duration   | Query                                               |
+----------+------------+-----------------------------------------------------+
|        1 | 0.00177775 | show variables like '%profiling%'                   |
|        2 | 0.00037900 | select * from test where id='087878'                |
|        3 | 0.34618025 | select * from test where servnumber='1367008787'    |
|        4 | 0.31986825 | select * from test where servnumber='13670087879'   |
+----------+------------+-----------------------------------------------------+
```

- 第四步：查看语句的执行性能详情

```
mysql> show profile for query 4;
+----------------------+----------+
| Status               | Duration |
```

```
+----------------------+----------+
| starting             | 0.000100 |
| checking permissions | 0.000010 |
| Opening tables       | 0.000023 |
| init                 | 0.000045 |
| System lock          | 0.000015 |
| optimizing           | 0.000016 |
| statistics           | 0.000028 |
| preparing            | 0.000020 |
| executing            | 0.000006 |
| Sending data         | 0.319489 |
| end                  | 0.000037 |
| query end            | 0.000012 |
| closing tables       | 0.000012 |
| freeing items        | 0.000040 |
| cleaning up          | 0.000017 |
+----------------------+----------+
```

- 性能线程的详细解释官方文档链接：

```
https://dev.mysql.com/doc/refman/5.7/en/general-thread-states.html
```

9.3 mysql语句优化的几个小建议

简介：介绍日常工作应该尽量避免的sql语句

- 第一个注意点：

```
尽量避免使用select *from ，尽量精确到想要的结果字段
```

- 第二个注意点：

```
尽量避免条件使用or
```

- 第三个注意点：

```
记得加上limit 限制行数，避免数据量过大消耗性能
```

- 第四个注意点：

```
使用模糊查询时，%放在前面是会使索引失效

mysql> explain select * from test where servnumber like '%1367000%'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
   partitions: NULL
         type: ALL
possible_keys: NULL
```

```
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 996303
     filtered: 11.11
        Extra: Using where
```

- 第五个注意点：

```
要小心条件字段类型的转换
```

小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 xdclass.net

第10章：mysql数据安全核心知识之备份技能

简介：介绍备份数据的重要性以及一些常见的数据备份技能

10.1 mysql数据安全之备份的背景意义

简介：数据库备份的意义

- 数据备份的意义：

（1）保护数据的安全；

（2）在出现意外的时候（硬盘的损坏，断电，黑客的攻击），以便数据的恢复；

（3）导出生产的数据以便研发人员或者测试人员测试学习；

（4）高权限的人员操作失误导致数据丢失，以便恢复；

10.2 mysql数据安全之备份的介绍

简介：介绍数据备份

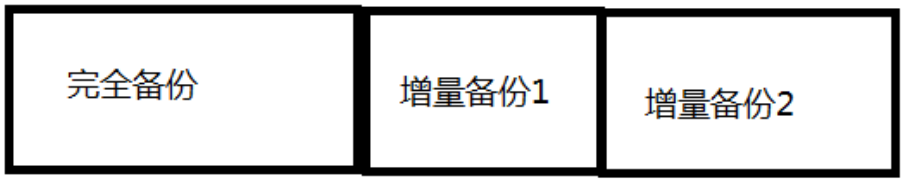
- 数据库的备份类型：

（1）完全备份：对整个数据库的数据进行备份

（2）部分备份：对部分数据进行备份（可以是一张表也可以是多张表）

增量备份：是以上一次备份为基础来备份变更数据的，节约空间

差异备份：是以第一次完全备份的基础来备份变更备份的，浪费空间



- 数据库备份的方式：

（1）逻辑备份：直接生成sql语句保存起来，在恢复数据的时候执行备份的sql语句来实现数据的恢复

（2）物理备份：直接拷贝相关的物理数据

区别：逻辑备份效率低，恢复数据效率低，但是逻辑备份节约空间；物理备份浪费空间，但是相对逻辑备份而言效率比较高

- 数据库备份的场景：

（1）热备份：备份时，数据库的读写操作不会受到影响

（2）温备份：备份时，数据库的读操作可以进行，但是写操作不能执行

（3）冷备份：备份时，不能进行任何操作

10.3 mysql数据安全之mysqldump备份实例（跨机器）

简介：如何利用mysql自带命令mysqldump来备份单库或者多库

- mysqldump使用语法：

  mysqldump -u 用户 -h host -p 密码 dbname table > 路径

- 远程备份单库例子：

```
mysqldump -uroot -pabc123456 -h120.25.93.69 zabbix | gzip  >
/mysql_data_back/zabbix_users.sql.gz
```

- 远程备份单库例子并保留创建库语句：

```
mysqldump -uroot -pabc123456 -h120.25.93.69 --databases zabbix | gzip  >
/mysql_data_back/zabbix_bak.sql.gz
```

- 远程备份单库单表的例子：

```
mysqldump -uroot -pabc123456 -h120.25.93.69 zabbix  users | gzip  >
/mysql_data_back/zabbix_users.sql.gz
```

- 远程备份多库的例子：

```
mysqldump -uroot -pabc123456 -h120.25.93.69 --databases zabbix XD | gzip  >
/mysql_data_back/zabbix_XD.sql.gz
```

- 远程备份全库的例子：

```
mysqldump -uroot -pabc123456 -h120.25.93.69 --all-databases  | gzip  >
/mysql_data_back/all.sql.gz
```

10.4 mysql数据安全之mysql数据的恢复

简介：如何恢复数据

- 远程恢复数据（备份的数据文件里有创建库的语句）：

```
mysql -uroot -pabc123456 -h120.25.93.69  < zabbix_bak.sql
```

- 远程恢复数据（备份的数据文件里没有创建库的语句）：

```
mysql -uroot -pabc123456 -h120.25.93.69  zabbix < zabbix_bak.sql
```

10.5 mysql数据安全之物理备份

简介：详解数据库源文件以及如何物理备份

- 看找数据库源文件路径（一）：

```
mysql> show variables like 'datadir%';
+---------------+---------------+
| Variable_name | Value         |
+---------------+---------------+
| datadir       | /data/mydata/ |
+---------------+---------------+
```

- 看找数据库源文件路径（二）：

```
 cat /etc/my.cnf
```

- MyISAM表源文件：

```
db.opt：创建库的时候生成，主要存储着当前库的默认字符集和字符校验规则

.frm ：记录着表结构信息的文件

.MYI：记录着索引的文件

.MYD :记录着表的数据
```

- InnoDB表源文件：InnoDB有着共享表空间跟独立表空间的概念。

```
db.opt：创建库的时候生成，主要存储着当前库的默认字符集和字符校验规则

.frm ：记录着表结构信息的文件

.ibd ：独立表空间，里边记录这个表的数据和索引

ibdata1：共享表空间，里边记录表的数据和索引
```

- 请求全局读锁：

```
flush tables with read lock;
```

- 解锁：

```
unlock tables;
```

10.6 mysql数据安全之利用二进制日志mysqlbinlog备份数据

简介：讲解如何利用二进制日志来备份数据

- 什么是二进制日志：

```
二进制日志就是记录着mysql数据库中的一些写入性操作，比如一些增删改，但是，不包括查询！
```

- 二进制日志有哪些功能：

```
一般情况下，二进制日志有着数据复制和数据恢复的功能
```

- 注意：

```
开启二进制日志会有1%的性能消耗！
```

- 查看二进制日志是否开启：

```
mysql> show variables like 'log_bin%';
+---------------------------------+-------+
| Variable_name                   | Value |
+---------------------------------+-------+
| log_bin                         | OFF   |
```

- 开启二进制日志 : vi /etc/my.cnf

```
[mysqld]
log-bin=/data/mydata/log_bin/mysql-bin
server-id=1
```

- 查看所有的binlog日志列表：

```
mysql> show master logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |     23638 |
+------------------+-----------+
```

- 刷新二进制日志：

```
flush logs;
```

- 重置(清空) 二进制日志文件：

```
mysql> show master logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |      1091 |
+------------------+-----------+
```

- 使用mysqldump备份数据时，加上-F选项可以重新生成一个新的二进制日志文件

```
mysqldump -uroot -p XD user -F  > user_bak.sql
```

10.7 mysql数据安全之利用二进制日志mysqlbinlog恢复数据

简介：讲解如何利用二进制日志来恢复数据

- 查看二进制日志文件的内容报错：

```
[root@xdclass-public log_bin]# mysqlbinlog mysql-bin.000002
mysqlbinlog: [ERROR] unknown variable 'default-character-set=utf8'
```

    - 解决：

      第一种：在mysqlbinlog 后边加上 --no-defaults

      第二种：注释掉配置文件里边的default-character-set=utf8

- 把二进制日志文件导出成普通文件：

```
 mysqlbinlog --base64-output=DECODE-ROWS -v mysql-bin.000002 > mysqlbin.sql
```

- 找出要恢复的位置：

```
（1）找出关键字的行数：mysqlbinlog --no-defaults mysql-bin.000002 | cat -n  | grep -iw
'drop'

[root@xdclass-public log_bin]# mysqlbinlog --no-defaults mysql-bin.000002 | cat -n  |
grep -iw 'drop'
  4180  DROP TABLE `user` /* generated by server */


（2）打印出相关内容：mysqlbinlog --no-defaults mysql-bin.000002 | cat -n | sed -n
'4170,4180p'

[root@xdclass-public log_bin]# mysqlbinlog --no-defaults mysql-bin.000002 | cat -n |
sed -n '4170,4180p'
  4170  # at 59578
  4171  #190419  0:41:48 server id 1  end_log_pos 59609 CRC32 0x36cda2b7        Xid =
6380
  4172  COMMIT/*!*/;
  4173  # at 59609
  4174  #190419  0:41:48 server id 1  end_log_pos 59674 CRC32 0x8de2f06a
Anonymous_GTID  last_committed=145      sequence_number=146
  4175  SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
  4176  # at 59674
  4177  #190419  0:41:48 server id 1  end_log_pos 59787 CRC32 0x6b2edd2b        Query
  thread_id=14    exec_time=0     error_code=0
  4178  use `XD`/*!*/;
  4179  SET TIMESTAMP=1555605708/*!*/;
  4180  DROP TABLE `user` /* generated by server */
[root@xdclass-public log_bin]#
```

- 恢复数据：
  - 第一步：把备份的数据表user恢复到数据库中：mysql -uroot -p XD < /mysql_data_back/user_bak.sql
  - 第二步：利用上面找到的删除的位置进行恢复数据

```
mysqlbinlog --no-defaults --set-charset=utf8  --stop-position="59674"
/data/mydata/log_bin/mysql-bin.000002 | mysql -uroot -p
```

- 登录数据库查看数据是否恢复回来

10.8 总结第一季整套课程

简介：总览课程大纲

第二季待续