MySQL的SQL语句解析

一、数据库介绍

1、什么是数据库

- 数据库就是一个存放计算机数据的仓库，这个仓库是按照一定的数据结构（数据结构是指数据的组织形式或数据之间的联系）来对数据进行组织和存储的，可以通过数据库提供的多种方法来管理其中的数据。

2、数据库的种类

- 最常用的数据库模式主要有两种，即关系型数据库和非关系型数据库。

3、生产环境常用数据库

- 生产环境主流的关系型数据库有 Oracle、Microsoft SQL Server、MySQL/MariaDB等。
- 生产环境主流的关系型数据库有 MongoDB Memcached Redis

4、关系型数据库

1、关系型数据库介绍

- 关系型数据库模型是把复杂的数据结构归结为简单的二元关系（即二维表格形式）。在关系型数据库中，对数据的操作几乎全部建立在一个或多个关系表格上，通过这些关联的表格分类、合并、连接或选取等运算来实现数据的管理。
- 关系型数据库诞生距今已有 40 多年了，从理论产生到发展到实现产品，例如：常见的 MySQL 和 Oracle 数据库，Oracle 在数据库领域里上升到了霸主地位，形成每年高达数百亿美元的庞大产业市场，而 MySQL 也是不容忽视的数据库，以至于被 Oracle 重金收购了。

2、关系型数据库小结

- 关系型数据库在存储数据时实际就是采用的一张二维表（和 Word 和 Excell 里表格几乎一样）。
- 市场占有量较大的是 MySQL 和 Oracle 数据库，而互联网场景最常用的是 MySQL 数据库。
- 通过 SQL 结构化查询语言来存取、管理关系型数据库的数据。
- 关系型数据库在保持数据安全和数据一致性方面很强，遵循 ACID 理论

5、非关系型数据库

1、非关系数据库诞生的背景

- 非关系型数据库也被称为 NoSQL 数据库，NoSQL 的本意是 “Not Only SQL”，指的是非关系型数据库，而不是“NO SQL”的意思，因此，NoSQL 的产生并不是要彻底否定关系型数据库，而是作为传统数据库的一个有效补充。NoSQL 数据库在特定的场景下可以发挥难以想象的高效率和高性能。
- 随着 Web2.0 网站的兴起，传统的关系型数据库在应付 Web2.0 网站，特别是对于规模日益扩大的海量数据，超大规模和高并发的微博、微信、SNS 类型的 Web2.0 纯动态网站已经显得力不从心，暴露了很多难以克服的问题，例如：传统的关系型数据库IO瓶颈、性能瓶颈都难以有效突

破，于是开始出现了大批针对特定场景，以高性能和使用便利为目的功能特异化的数据库产品。NoSQL（非关系型）类的数据库就是这样的情景中诞生并得到了非常迅速的发展。

- NoSQL 是非关系型数据库的广义定义。它打破了长久以来关系型数据库与ACID理论大一统的局面。NoSQL数据存储不需要固定的表结构，通常也不存在连续操作。在大数据存取上具备关系型数据库无法比拟的性能优势。该术语（NoSQL）在2009年初得到了广泛认同。
- 当今的应用体系结构需要数据存储在横向伸缩性上能够满足需求。而NoSQL存储就是为了实现这个需求而诞生的。Google的 BigTable 与Amazon的Dynamo是非常成功的商业 NoSQL 实现。一些开源的 NoSQL 体系，如 Facebook 的 Cassandra，Apache 的 HBase，也得到了广泛认同，Redis，MongoDB 也逐渐的越来越受到各类大中小型公司的欢迎和追捧。

2、非关系型数据库小结

- NoSQL 数据库不是否定关系型数据库，而是作为关系数据库的一个重要补充。
- NoSQL 数据库为了灵活及高性能、高并发而生，忽略影响高性能、高并发的功能。
- 在NoSQL 数据库领域，当今的最典型产品为 Redis（持久化缓存）、MongoDB、Memcached（纯内存）等。
- NoSQL 数据库没有标准的查询语言（SQL），通常使用REST式的数据接口或者查询API。

3、非关系型数据库种类

1、键值（Key-Value）存储数据库

- 键值数据库就类似传统语言中使用的哈希表。可以通过key来添加、查询或者删除数据，因为使用key主键访问，所以会获得很高的性能及扩展性。
- 键值（Key-Value）数据库主要是使用一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。Key-Value模型对于IT系统来说的优势在于简单、易部署、高并发。
- 典型产品：Memcached、Redis、MemcachedB、Berke ley DB

2、列存储（Column-Oriented）数据库

- 列存储数据库将数据存储存在列族（Column Family）中，一个列族存储经常被一起查询的相关数据。举个例子，如果有一个 Person 类，通常会一起查询他们的姓名和年龄而不是薪资。这种情况下，姓名和年龄就会被放入一个列族中，而薪资则在另一个列族中。这部分数据库通常用来应对分布式存储的海量数据。键仍然存在，但是他们的特点是指向了多个列。这些列是由列家族来安排的。
- 典型产品：Cassandra，HBase

3、面向文档（Document-Oriented）的数据库

- 文档型数据库的灵感是来自于Lotus Notes办公软件的，而且它同第一种键值存储相类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库可以看作是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。
- 面向文档数据库会将数据以文档的形式存储。每个文档都是自包含的数据单元，是一系列数据项的集合。每个数据项都有一个名称与对应的值，值既可以是简单的数据类型，如字符串、数字和日期等；也可以是复杂的类型，如有序列表和关联对象。数据存储的最小单位是文档，同一个表中存储的文档属性可以是不同的，数据可以使用XML、JSON 或者 JsonB 等多种形式存储。
- 典型产品：MongDB、CouchDB

4、图形（Graph）数据库

- 图形数据库允许我们将数据以图的方式存储。实体会被作为顶点，而实体之间的关系则会被作为边。比如我们有三个实体，Steve Jobs、Apple和Next，则会有两个“Founded by” 的边将 Apple 和 Next 连接到 Steve Jobs。
- 图形结构的数据库同其他行列以及刚性结构的 SQL 数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL 数据库没有标准的查询语言（SQL），因此进行数据查询需要定制数据模型。许多 NoSQL 数据库都有 REST 的数据接口或者查询 API。
- 典型产品：Neo4J、InfoGr id

6、常用关系型数据库管理系统

1、Oracle 数据库

- Oracle 前身叫 SDL，由 Larry Ellison 和另两个编程人员在1977创办，他们开发了自己的拳头产品，在市场上大量销售，1979年，Oracle 公司引入了第一个商用 SQL关系数据库管理系统。Oracle公司是最早开发关系数据库的厂商之一，其产品支持最广泛的操作系统平台。目前 Oracle 关系数据库产品的市场占有率数一数二。
- Oracle 公司是目前全球最大的数据库软件公司，也是近年业务增长极为迅速的软件提供与服务商。
- 2007年7月12日，甲骨文公司在美国纽约宣布推出数据库 Oracle llg，这是 Oracle 数据库的最新版本。Oracle 介绍说，Oraclellg 有400多项功能，经过了1500万个小时的测试，开发工作量达到了3.6万人/月。Oraclellg 在安全，XML DB，备份等方面得到了很大提升。
- 主要应用范围：传统大企业，大公司，政府，金融，证券等等。
- 版本升级：Oracle8i，Oracle9i，Oracle10g，Oracle11g，Oracle12c。

2、MySQL 数据库

- MySQL 数据库是一个中小型关系型数据库管理系统，软件开发者为瑞典 MySQL AB 公司。在2008年1月16号被 Sun 公司收购，后 Sun 公司又被 Oracle 公司收购。目前MySQL 被广泛地应用在 Internet 上的大中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多大中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库，甚至国内知名的淘宝网也选择弃用 Oracle 而更换为更开放的 MySQL。
- MySQL 数据库主要应用范围：互联网领域，大中小型网站，游戏公司，电商平台等等。

3、MariaDB 数据库

- MariaDB 数据库管理系统是 MySQL 数据库的一个分支，主要由开源社区维护，采用 GPL 授权许可。开发这个 MariaDB 数据库分支的可能原因之一是：甲骨文公司收购了MySQL 后，有将 MySQL 闭源的潜在风险，因此 MySQL 开源社区采用分支的方式来避开这个风险。
- 开发 MariaDB 数据库的目的是完全兼容 MySQL 数据库，包括 API 和命令行，使之能轻松的成为 MySQL 的代替品。在存储引擎方面，使用 XtraDB （英语：XtraDB）来代替MySQL 的 InnoDB MariaDB 由 MySQL 的创始人 Michael Widenius（英语：Michael Widenius）主导开发，他早前曾以 10 亿美元的价格，将自己创建的公司MySQL AB卖给了 SUN，此后，随着 SUN 被甲骨文收购，MySQL 的所有权也落入Oracle 的手中，MariaDB 数据库的名称来自 MySQL 的创始人 Michael Widenius 的女儿 Maria 的名字。
- MariaDB 基于事务的 Maria 存储引擎，替换了MySQL 的 MyISAM 存储引擎，它使用了 Percona 的 XtraDB（InnoDB的变体）。这个版本还包括了 PrimeBaseXT（PBXT）和 FederatedX 存储引擎。

4、SQL Server 数据库

- Microsoft SQL Server是微软公司开发的大型关系型数据库系统。1987年，微软和IBM合作开发完成 OS/2，IBM 在其销售的 OS/2 ExtendedEdition 系统中绑定了 OS/2 DatabaseManager，而微软产品线中尚缺少数据库产品。为此，微软将目光投向 Sybase，同 Sybase 签订了合作协议，使用 Sybase 的技术开发基于 OS/2 平台的关系型数据库。1989年，微软发布了 SQLServer1.0 版。Microsoft 在与 Sybase分道扬镳后，随后在其6.05和7.0版本中重写了核心数据库系统。
- SQL Server 的功能比较全面，效率高，可以作为中型企业或单位的数据库平台。
- SQL Server 可以与 Windows 操作系统紧密集成，不论是应用程序开发速度还是系统事务处理运行速度，都能得到较大的 提升。对于在 Windows 平台上开发的各种企业级信息管理系统来说，不论是C/S（客户机/ 服务器）架构还是B/S（浏览器/服务器）架构，SQL Server 都是一个很好的选择。SQL Server 的缺点是只能在 Windows 系统下运行。
- 主要应用范围：部分企业电商（央视购物），使用windows服务器平台的企业。

6、其他关系型数据库

- DB2, PostgreSQL,Informix,Sybase等。这些关系型数据库逐步的淡出了普通运维的视线，特别是互联网公司几乎见不到。

7、常用非关系型数据库管理系统

1、Memcached（Key-Value）

- Memcached 是一个开源的、高性能的、具有分布式内存对象的缓存系统。通过它可以减轻数据库负载，加速动态的 Web 应用，最初版本由 LiveJoumal 的 Brad Fitzpatrick在2003年开发完成。目前全球有非常多的用户都在使用它来构建自己的大负载网站或提高自己的高访问网站的响应速度。注意：Memcache 是这个项目的名称，而Memcached 是服务器端的主程序文件名。
- 缓存一般用来保存一些经常被存取的对象或数据（例如，浏览器会把经常访问的网页缓存起来一样），通过缓存来存取对象或数据要比在磁盘上存取快很多，前者是内存，后 者是磁盘。Memcached 是一种纯内存缓存系统，把经常存取的对象或数据缓存在 Memcached 的内存中，这些被缓存的数据被程序通过API的方式被存取，Memcached里面的数据就像
- 一张巨大的 HASH 表，数据以 Key-Value 对的方式存在。Memcached 通过缓存经常被存取的对象或数据，从而减轻频繁读取数据库的压力，提高网站的响应速度，构建出速度更快的可扩展的Web应用。官方：http://Memcached.org/
- 由于Memcached 为纯内存缓存软件，一旦重启所有数据都会丢失，因此，新浪网基于Memcached 开发了一个开源项目 MemcacheDB。通过为 Memcached 增加 Berkeley DB 的持久化存储机制和异步主辅复制机制，使 Memcached 具备了事务恢复能力、持久化数据存储能力和分布式复制能力，MemcacheDB非常适合需要超高性能读写速度、持久化保存的应用场景，但是最近几年逐渐被其他的持久化产品替代例如Redis。

2、Redis（Key-Value）

- Redis 是一个Key-Value 型存储系统。但Redis支持的存储value 类型相对更多，包括 string（字符串）、list（链表）、set（集合）和 zset（有序集合）等。这些数据类型都支持 push/pop、add/remove 及取交集、并集和差集及更丰富的操作，而且这些操作都 是原子性的。在此基础上，Redis 支持各种不同方式的排序。与 Memcached 一样，为了保证效率，Redis 的数据都是缓存在内存中。区别是 Redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了 Master-Slave（主从）同步。
- Redis 是一个高性能的 Key-Value 数据库。Redis 的出现，很大程度补偿了 Memcached 这类 Key-Value 存储的不足，在部分场合可以对关系数据库綱很好的补充作用。它提供了 Python，Ruby，Erlang，PHP 客户端，使用很方便。官方：http://www.Redis.io/documentation
- Redis 特点：

1. 支持内存缓存，这个功能相当于 Memcached。

2. 支持持久化存储，这个功能相当于 MemcacheDB，Ttserver。
3. 数据类型更丰富。比其他 Key-Value 库功能更强。
4. 支持主从集群，分布式。
5. 支持队列等特殊功能。

- 应用：缓存从存取 Memcached 更改存取 Redis。

3、MongoDB （Document-Web）

- MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据结构非常松散，类似 Json 的 Bjson 格式，因此可以存储比较复杂的数据类型。MongoDB 最大的特点是他支持查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。它的特点是高性能、易部署、易使用，存储数据非常方便。

- 主要功能特性：

1. 面向集合（Collenction-Orented）存储，易存储对象类型的数据 ，数据被分组存储在数据集中，被称为一个集合（Collenction）。每个集合在数据库中都有一个唯一的标识名，并且可以包含无限数目的文挡。集合的概念类似关系型数据库（RDBMS）里的表（table），不同的是它不需要定义任何模式（schema）。
2. 模式自由（Schema-Free），意味着对于存储在 MongoDB 数据库中的文件，我们不需要知道它的任何结构定义。如果需要，你完全可以把不同结构的文件存储在同一个数据库里。
3. 支持动态查询
4. 支持完全索引，包含内部对象
5. 支持查询
6. 支持复制和故障恢复
7. 使用高效的二进制数据存储，包括大型对象（如视频等）
8. 自动处理碎片，以支持云计算层次的扩展性
9. 支持 Ruby，Python，Java，C++， PHP 等多种语言
10. 文件存储格式为 Bson （一种 Json 的扩展）Bson （Binary Serialized document Format）存储形式是指：存储在集合中的文档，被存储为键-值对的形式。键用于唯一标识一个文档，为字符串类型，而值则可以是各中复杂的文件类型。
11. 可通过网络访问

- MongoDB 服务端可运行在 Linux、Windows 或 OS X 平台，支持32位和64位应用，默认端口为27017。推荐运行在64位平台。

- McmgoDB 把数据存储在文件中（默认路径为：/data/db），为提高效率使用内存映射文件进行管理。

二、安装 MySQL 数据库

1、安装 MySQL Repository

1、默认 yum 存储库安装

```
[root@qfedu.com ~]# yum -y install wget     # 安装 wget下载工具
[root@qfedu.com ~]# wget https://repo.mysql.com/mysql57-community-release-el7-
11.noarch.rpm  # 下载 mysql 官方 yum 源安装包
[root@qfedu.com ~]# yum -y localinstall mysql57-community-release-el7-
11.noarch.rpm  # 安装 mysql 官方 yum 源
```

2、选择指定发行版本安装

- 使用 MySQL Yum 存储库时，默认情况下会选择要安装的最新GA版本MySQL。默认启用最新GA系列（当前为MySQL 8.0）的子存储库，而所有其他系列（例如，MySQL 5.7系列）的子存储库均被禁用。查看已启用或禁用了存储库。

1、列出所有版本

```
[root@qfedu.com ~]# yum repolist all | grep mysql
```

- 发现启用最新8.0版本是 enabled 的，5.7版本是 disabled 的，需要安装5.7版本时，所以把8.0的进行禁用，然后再启用5.7版本

2、安装 yum 配置工具

```
[root@qfedu.com ~]# yum -y install yum-utils
```

3、禁用 8.0 版本

```
[root@qfedu.com ~]# yum-config-manager --disable mysql80-community
```

4、启用 5.7 版本

```
[root@qfedu.com ~]# yum-config-manager --enable mysql57-community
```

5、检查启用版本

- 注意：进行安装时请确保只有一个版本启用，否则会显示版本冲突

```
[root@qfedu.com ~]# yum repolist enabled | grep mysql
```

2、安装 MySQL

- 需要安装MySQL Server, MySQL client 已经包括在 server 套件内

```
[root@qfedu.com ~]# yum -y install mysql-community-server mysql       # 安装服务端，客户端
[root@qfedu.com ~]# systemctl start mysqld                            # 启动 mysql 服务
[root@qfedu.com ~]# systemctl enable mysqld                           # 设置 mysql 服务开机启动
[root@qfedu.com ~]# ls /var/lib/mysql                                 # 查看 mysql 安装
[root@qfedu.com ~]# grep 'tqfeduorary password' /var/log/mysqld.log   # 获取首次登录密码
[root@qfedu.com ~]# mysql -uroot -p'awm3>!QFl6zR'                     # 登录mysql数据库
mysql > alter user 'root'@'localhost' identified by 'Qf.123com';      # 修改 mysql 数据库密码（密码必须符合复杂性要求，包含字母大小写，数字，特赦符号，长度不少于8位）
[root@qfedu.com ~]# mysql -uroot -p'Qf.123com'                        # 用新密码登录数据库
```

3、重启 MySQL

```
[root@qfedu.com ~]# systemctl restart mysqld
```

4、创建 qfedu 库并设置权限

```
[root@qfedu.com ~]# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.39 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>CREATE DATABASE qfedudb CHARACTER SET utf8 COLLATE utf8_bin;
Query OK, 1 row affected (0.00 sec)

mysql>GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER,INDEX on qfedudb.* TO
'qfedu'@'localhost' IDENTIFIED BY 'Yangge.123com';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW GRANTS FOR 'qfedu'@'localhost';
+-------------------------------------------------------------------------------
------------------------------+
| Grants for qfedu@localhost
                                |
+-------------------------------------------------------------------------------
------------------------------+
```

```
| GRANT USAGE ON *.* TO 'qfedu'@'localhost' IDENTIFIED BY PASSWORD
'*841E9705B9F4BD3195B7314CA58A7E3B3B349F71' |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON
`qfedudb`.* TO 'qfedu'@'localhost'        |
+-----------------------------------------------------------------------------
-----------------------------+
2 rows in set (0.00 sec)

mysql> SHOW GRANTS FOR 'qfedu'@'172.16.0.122';
+-----------------------------------------------------------------------------
-----------------------------+
| Grants for qfedu@172.16.0.122
                                  |
+-----------------------------------------------------------------------------
-----------------------------+
| GRANT USAGE ON *.* TO 'qfedu'@'172.16.0.122' IDENTIFIED BY PASSWORD
'*841E9705B9F4BD3195B7314CA58A7E3B3B349F71' |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON
`qfedudb`.* TO 'qfedu'@'172.16.0.122'        |
+-----------------------------------------------------------------------------
-----------------------------+
2 rows in set (0.01 sec)

mysql> \q
Bye
```

三、结构化查询语言 SQL

- 结构化查询语言(Structured Query Language)，简称SQL，是数据库的标准查询语言。可以通过DBMS对数据库进行定义数据，操纵数据，查询数据，数据控制等

1、数据定义语言 (DDL)

- Data Dafinitaon Language
- 如创建表 create
- 删除表 drop
- 修改表 alter
- 清空表 truncate，彻底清空，无法找回

```
show databases;                                  # 查看所有数据库：
show tables;                                     # 查看所有表：
drop database db1;                               # 删除数据库
create database db1 default character set utf8; # 创建数据库
use database;                                    # 选择数据库
create table t1(id int(3),name varchar(20));     # 创建表
desc t1                                          # 查看表
insert into t1 values(1,'user1');                # 插入数据
alter table t1 add(age int(3));                  # 添加表字段语句
alter table t1 drop id;                          # 删除表字段语句
alter table t1 modify age varchar(2);            # 修改表字段类型格式
alter table t1 change age p1age int(3);          # 修改表字段名称
alter table t1 rname per;                        # 修改表名
truncate table t1;                               # 清空表结构
drop table t1;                                   # 删除表结构
```

2、数据操纵语言(DML)

- insert：向表中插入数据
- delete：删除表中的数据，格式：delete from tablname [where 条件]
- update：修改表中的数据 格式：update tablname set colName1=value1[,colName2=value2] [where 条件]
- where : 对表中的数据增加条件进行限制，起到过滤的作用。

1. 格式: where colName 关系运算符 value [or|and 条件2]
2. 关系运算符: >，>=，<，<=, 等于：=，不等于：!= 或 <>

- null值操作：比较null时，不能使用=或者!= 或者<>，而是使用 is或者is not，在select子句中，使用关系运算符

3、实验实例

1、创建一个表

```
create table qfedu01(
id int,
name varchar(20),
birth date,
address varchar(50)
);
```

2、插入数据

```
插入数据  100，'jack' ,'shanghai' ，下面是两种插入方式，第二种可以批量插入。只需添加多个用逗号隔开即可
insert into qfedu01 values (101,'jack',null,'shanghai');
insert into qfedu01(id,name,address) values(102,'rose','beijing');
```

3、删除数据

```
# 删除表qfedu01中的所有数据。
delete from qfedu01;
# 删除表中qfedu01中的id为101的记录。
delete from qfedu01 where id=101;
# 删除表qfedu01中 id为102和地址为上海的的数据
select * from qfedu01;
delete from qfedu01 where id=102 and address='shanghai';
```

4、修改数据

```
# 修改表 qfedu01 中的字段 address 为中国。
update qfedu01 set address='china';
# 修改表 qfedu01 中的 id 为 102 的 address 为english，人名为 micheal。
update qfedu01 set address='english',name='micheal' where id=102;
# 将生日为 null 的记录的 name 改为 'general'
update qfedu01 set name='general' where tbirth is null;
# 将 id 为 101 的 birth 改为'2000-8-8';
update qfedu01 set birth = '2000-8-8' where id = 101;
# 将 id 为 102 的 address 改为 null.
update qfedu01 set address=null where id=102;
```

3、事物控制语言(TCL)

- 事物控制语言 （Transation Control Language） 有时可能需要使用 DML 进行批量数据的删除，修改，增加。比如，在一个员工系统中，想删除一个人的信息。除了删除这个人的基本信息外，还应该删除与此人有关的其他信息，如邮箱，地址等等。那么从开始执行到结束，就会构成一个事务。对于事务，要保证事务的完整性。要么成功，要么撤回。
- 事务要符合四个条件(ACID)：

1. 原子性(Atomicity)：事务要么成功，要么撤回。不可切割性。
2. 一致性(Consistency)：事务开始前和结束后，要保证数据的一致性。转账前账号A和账号B的钱的总数为10000，转账后账号A和账号B的前的总数应该还是10000;
3. 隔离性(Isolation)：当涉及到多用户操作同一张表时，数据库会为每一个用户开启一个事务。那么当其中一个事务正在进行时，其他事务应该处于等待状态。保证事务之间不会受影响。
4. 持久性(Durability)：当一个事务被提交后，我们要保证数据库里的数据是永久改变的。即使数据库崩溃了，我们也要保证事务的完整性。

- commit 提交
- rollback 撤回，回滚。
- savepoint 保存点
- 只有DML操作会触发一个事务。存储引擎(ENGINE):就是指表类型.当存储引擎为innodb时，才支持事务。默认的存储引擎为 Myisam。不支持事务。
- 事务的验证：

  第一步：start transaction (交易) 第二步：savepoint 保存点名称。 第三步：DML 第四步：commit/rollback;

4、数据查询语言(DQL)

数据查询语言（Data Query Language）

1、Select/for 基本查询语句

```
格式： select子句 from子句     select colName[,colName.......] from tablname;
```

2、给列起别名

```
格式： select 列名1 as "要起的名" [, 列名2 as "要起的名" ,... ] from tablname;
```

3、Where子句

- 作用：在增删改查时，起到条件限制的作用
- 多条件写法：in | not in (集合元素，使用逗号分开); 注意：同一个字段有多个值的情况下使用。in 相当于 or， not in 相当于 and
- all | any 与集合连用，此时集合中的元素不能是固定的必须是从表中查询到的数据。
- 范围查询：colName between val1 and val2; 查询指定列名下的val1到val2范围中的数据
- 模糊查询：like

1. 通配符：% 表示0或0个以上字符。_ 表示匹配一个字符
2. 格式: colName like value;

4、 实验实例

1、创建表

```
create table qfedu (
qfeduid int(5),
name VARCHAR(10),
job VARCHAR(9),
mgr int(4),
hiredate DATE,
sal int(7),
comm int(7),
deptid int(2)
);
```

2、添加 数据

```
insert into qfedu (qfeduid,name,job,mgr,hiredate,sal,comm,deptid) values
(7369  ,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20),
(7499  ,'qfedu','SALESMAN' , 7698 , '1981-2-10' ,  1600 , 300  , 30),
(7521  ,'WARD', 'SALESMAN' , 7698,  '1981-2-22' ,  1250 , 500  , 30),
(7566  ,'JONES','MANAGER' ,  7839,  '1981-4-2'  ,  2975 , NULL , 20),
(7654  ,'MARTIN','SALESMAN' , 7698,  '1981-9-28',   1250 , 1400 , 30),
(7698  ,'BLAKE','MANAGER' ,  7839 , '1981-5-1' ,   2850 , NULL , 30),
(7782  ,'CLARK','MANAGER' ,  7839 , '1981-6-9' ,   2450 , NULL , 10),
(7788  ,'SCOTT','ANALYST' ,  7566,  '1987-4-19',   3000 , NULL , 20),
(7839  ,'KING','PRESIDENT' ,NULL,  '1981-11-17',  5000 , NULL , 10),
(7844  ,'TURNER','SALESMAN' , 7698,  '1981-9-8',    1500 , 0    ,  30),
(7876  ,'ADAMS','CLERK'  ,   7788 , '1987-5-23',   1100,  NULL , 20),
(7900  ,'JAMES','CLERK' ,    7698 , '1981-12-3',   950 ,  NULL , 30),
(7902  ,'FORD','ANALYST' ,  7566 , '1981-12-3' ,  3000 , NULL , 20),
(7934  ,'MILLER','CLERK',     7782 , '1982-1-23',   1300 , NULL , 10),
(8002  ,'IRONMAN','MANAGER',   7839 , '1981-6-9',    1600, NULL , 10),
(8003  ,'SUPERMAN','MANAGER',   7839 , '1981-6-9',    1600 , NULL , NULL);
```

3、查询数据

```
# 查询员工表 qfedu 中的 员工姓名，员工职位，员工入职日期和员工所在部门。
select name,job,hiredate,deptid from qfedu;
```

4、给列起别名

```
# 查询员工姓名和员工职位。分别起别名，姓名和职位。
select name as "姓名" ,job as "职位" from qfedu;
```

5、Where 子句

```
# 查询员工表中部门号为 10 和 20 的员工的编号，姓名，职位，工资
select qfeduid,name,job,sal from qfedu where deptid=10 or deptid =20;
# 查询员工表中部门号不是 10 和 20 的员工的所有信息。
select * from qfedu where deptid<>10 and deptid<>20;

# 使用in ，not in修改上面的练习
select qfeduid,name,job,sal from qfedu where deptid in(10,20);
select * from qfedu where deptid not in (10,20);
```

6、All|Any与集合连用

```
# 查询员工qfedu,blake,clark三个人的工资
select * from qfedu where sal>all(select sal from qfedu where name in
('qfedu','blake','clark'));
```

7、范围查询

```
# 查询工资大于等于1500并且小于等于2500的员工的所有信息。
select * from qfedu where sal between 1500 and 2500;
# 查询工资小于2000和工资大于2500的所有员工信息。
select * from qfedu where sal not between 2000 and 2500;
```

8、模糊查询：Like

```
# 查询员工姓名中有a和s的员工信息。
select name,job,sal,comm,deptid from qfedu where name like '%a%' and name like
'%s%';
select name,job,sal,comm,deptid from qfedu where name like '%a%s%' or name like
'%s%a%';
```

4、排序查询 Order By子句

- 当在查询表中数据时，记录比较多，有可能需要进行排序，此时可以使用 order by子句。
- 语法：

```
select...from tablname [where 子句][order by 子句]
```

- 注意: 可以通过一个或多个字段排序。
- 格式:

```
order by colName [ ASC | DESC ][ , colName1....[ ASC | DESC ] ];
```

- 排序规则: ASC：升序，DESC: 降序，默认是升序排序。

```
select * from qfedu order by sal DESC;
```

5、Distinct 去重

- 有的时候我们需要对重复的记录进行去重操作，比如，查询表中有哪些职位，此时，一种职位只需要显示一条记录就够。
- 位置：必须写在select关键字后。

```
select distinct name from qfedu;
```

6、分组查询 Group By 子句

- 分组查询与分组函数(聚合函数)：有的时候，我们可能需要查询表中的记录总数，或者查询表中每个部门的总工资,平均工资，总人数。这种情况需要对表中的数据进行分组统计。需要group by子句。
- 位置：

```
select...from name [where 条件] [group by子句] [order by子句]
```

- 用法:

```
group by Field1[,Field2]
```

- 注意：在分组查询时，select子句中的字段，除了聚合函数外，只能写分组字段。

```
# 查询以部门id为分组员工的平均工资
select deptid,avg(sal) as av from qfedu group by deptid;
```

- 聚合函数:

1. count(Filed) 统计指定字段的记录数。
2. sum(Filed) 统计指定字段的和。
3. avg(Filed) 统计指定字段的平均值
4. max(Filed) 返回指定字段中的最大值。
5. min(Filed) 返回指定字段中的最小值。

- 聚合函数会忽略null值。因此有时候需要使用函数：ifnull(field,value)（如果field字段对应的值不是null,就使用field的值，如果是null,就使用value.）
- 注意：多字段分组时（多表联合查询时不加任何条件），最多分组的数目为Filed1Field2[Filed3....]

7、分组查询添加条件 Having 子句

- 在分组查询时，有的时候可能需要再次使用条件进行过滤，这个时候不能 where子句，应该使用having子句。having子句后可以使用聚合函数。
- 位置：位于group by子句后

```
# 查询以部门id为分组员工的平均工资并且大于2000的部门
select deptid,avg(sal) as av from qfedu group by deptid having avg(sal)>2000;
```

8、基本查询总结

- 基本的查询语句包含的子句有：select子句，from子句，where子句，group by子句，having子句，order by子句
- 完整的查询语句：

```
select..from..[where..][group by..][having..][order by..]
```

- 执行顺序：先执行from子句，再执行where子句，然后group by子句，再次having子句，之后select子句，最后order by子句

9、高级关联查询

- 有的时候，查询的数据一个简单的查询语句满足不了，并且使用的数据在表中不能直观体现出来。而是需要预先经过一次查询才会有所体现。先执行的子查询。子查询嵌入到的查询语句称之为父查询。
- 子查询返回的数据特点：

1. 可能是单行单列的数据。
2. 可能是多行单列的数据
3. 可能是单行多列的数据
4. 可能是多行多列的数据

- 子查询可以在where、from、having、select字句中，在select子句中时相当于外连接的另外一种写法。

```
# 查询表中各部门人员中大于部门平均工资的人
select name,sal,a.deptid ,b.av
from qfedu a,
(select deptid,avg(ifnull(sal,0)) as av from qfedu group by deptid) b
where a.deptid=b.deptid and a.sal>b.av
order by deptid ASC;
```

5、数据控制语言(DCL)

数据控制语言 Data Control Language,作用是用来创建用户，给用户授权，撤销权限，删除用户。

格式:

1、创建用户

```
create user username@ip identified by newPwd;
create user 'qfedu'@'192.168.152.166' identified by 'Qfedu.123com';
```

2、显示用户的权限

```
show grants for username@ip;
```

3、授权

```
grant 权限1，权限2... on 数据库名.* to username@ip;
grant select,drop,insert on qfedu.* to 'qfedu'@'192.168.152.166';
```

- DML 权限：insert,delete,update
- DQL 权限：select
- DDL 权限：create,alter,drop...

4、撤销权限

```
revoke 权限1，权限2..on 数据库名.* from username@ip;
revoke drop on qfedu.* to 'qfedu'@'192.168.152.166';
```

5、删除用户

```
drop user username;
drop user qfedu;
```

6、使权限立即生效

```
flush privileges;
```

四、MySQL 操作实例

1、创建表

1、语法

```
create table 表名(
    字段名 列类型 [可选的参数], # 记住加逗号
    字段名 列类型 [可选的参数], # 记住加逗号
    字段名 列类型 [可选的参数]  # 最后一行不加逗号
)charset=utf8; # 后面加分号
```

2、列约束

- auto_increment : 自增 1
- primary key : 主键索引，加快查询速度, 列的值不能重复
- NOT NULL 标识该字段不能为空
- DEFAULT 为该字段设置默认值

3、实例

```
mysql> create table slt(
num int auto_increment primary key,
name char(10),
job char(10),
age int,
salary int,
descrip char(128)not null default ''
)charset=utf8;
```

2、查看表结构

```
mysql> desc slt;
+---------+-----------+------+-----+---------+----------------+
| Field   | Type      | Null | Key | Default | Extra          |
+---------+-----------+------+-----+---------+----------------+
| num     | int(11)   | NO   | PRI | NULL    | auto_increment |
| name    | char(10)  | YES  |     | NULL    |                |
| job     | char(10)  | YES  |     | NULL    |                |
| age     | int(11)   | YES  |     | NULL    |                |
| salary  | int(11)   | YES  |     | NULL    |                |
| descrip | char(128) | NO   |     |    |                |
+---------+-----------+------+-----+---------+----------------+
6 rows in set (0.02 sec)
```

此时表数据为空

```
mysql> select * from slt;
qfeduty set (0.00 sec)
```

3、插入数据

1、语法

```
insert into 表名 (列1, 列2) values (值1,'值2');
```

2、 实例

```
mysql> insert into slt(name, job, age, salary,descrip) values
('Tom','teacher',30,20000,'level2'),
('frank','teacher',31,21000,'level2'),
('jack','teacher',32,22000,'level2'),
('jhon','asistant',23,8000,'level3'),
('hugo','manager',45,30000,'level4'),
('jinhisan','teacher',26,9000,'level1')
;
Query OK, 6 row affected (0.01 sec)
```

4、数据查询

1、语法:

```
select 列1, 列2 from 表名;   （*代表查询所有的列）
```

1、查看岗位是 teacher 的员工姓名、年龄

```
mysql> select distinct name, age from slt where job='teacher';
```

2、查看岗位是teacher且年龄大于30岁的员工姓名、年龄

```
mysql> select distinct name, age from slt where job='teacher' and age>30;
```

3、查看岗位是 teacher且薪资在1000-9000范围内的员工姓名、年龄、薪资

```
mysql> select distinct name, age, salary from slt where job='teacher' and
(salary>1000 and salary<9000);
```

4、查看岗位描述不为NULL的员工信息

```
mysql> select * from slt where descrip!=NULL ;
```

5、查看岗位是teacher且薪资是10000或9000或30000的员工姓名、年龄、薪资

```
mysql> select distinct name, age, salary from slt where job='teacher' and
(salary=10000 or salary=9000 or salary=30000);
```

6、查看岗位是teacher且薪资不是10000或9000或30000的员工姓名、年龄、薪资

```
mysql> select distinct name, age, salary from slt where job='teacher' and
(salary not in (10000,9000,30000));
```

7、查看岗位是teacher且名字是'jin'开始的员工姓名、年薪

```
mysql> select * from slt where job='teacher' and name like 'jin%' ;
```

2、实例

```
例子1:
创建表:
```

```
mysql> create table t1(
    id int,
    name char(5)
)charset=utf8;
Query OK, 0 rows affected (0.72 sec)    # 如果回显是queryok，代表创建成功

插入数据：
mysql> insert into t1 (id, name) values (1, 'qfedu');
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+------+-------+
| id   | name  |
+------+-------+
|    1 | qfedu |
+------+-------+
1 row in set (0.00 sec)


例子2:
mysql> create table t2(
    id int auto_increment primary key,
    name char(10)
)charset=utf8;

mysql> insert into t2 (name) values ('qfedu1');

例子3: (推荐)
mysql> create table t3(
    id  int unsigned auto_increment primary key,
    name char(10) not null default 'xxx',
    age int not null default 0
)charset=utf8;

mysql> insert into t3 (age) values (10);
Query OK, 1 row affected (0.00 sec)

mysql> select * from t3;
+----+------+-----+
| id | name | age |
+----+------+-----+
|  1 | xxx  |  10 |
+----+------+-----+
1 row in set (0.00 sec)
```

5，修改表

1、修改表名

1、语法

```
ALTER TABLE 旧表名 RENAME 新表名;
```

2、实例

```
mysql> alter table t3 rename t4;
Query OK, 0 rows affected (0.19 sec)
```

2、增加字段

1、添加到最后语法

```
ALTER TABLE 表名
ADD 字段名 列类型 [可选的参数],
ADD 字段名 列类型 [可选的参数];
```

实例

```
mysql> create table t88(
    id int,
    age int
)charset=utf8;
mysql> alter table t88 add name varchar(32) not null default '';
Query OK, 0 rows affected (0.82 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

上面添加的列永远是添加在最后一列之后,还可以添加到最前面

2、添加到开头语法

```
ALTER TABLE 表名
ADD 字段名 列类型 [可选的参数] FIRST;
```

实例

```
mysql> alter table t88 add name3 varchar(32) not null default '' first;
Query OK, 0 rows affected (0.83 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

3、添加到指定字段后面

```
ALTER TABLE 表名
ADD 字段名 列类型 [可选的参数] AFTER 字段名;
```

实例

```
mysql> alter table t88 add name4 varchar(32) not null default '' after name;
Query OK, 0 rows affected (0.68 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

3、删除字段

1、语法

```
ALTER TABLE 表名  DROP 字段名;
```

2、实例

```
mysql> alter table t88 drop name4;
Query OK, 0 rows affected (0.66 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

4、修改字段

1、语法

```
ALTER TABLE 表名 MODIFY 字段名 数据类型 [完整性约束条件…];
```

2、实例

```
mysql> alter table t88 modify name char(20);
Query OK, 1 row affected (0.88 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

3、语法2

```
ALTER TABLE 表名 CHANGE 旧字段名 新字段名 新数据类型 [完整性约束条件…];
```

4、实例2

```
mysql> alter table t88 change name name2 varchar(32) not null default '';
Query OK, 1 row affected (0.82 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

6、删出表

1、语法

```
 drop table 表名;  # 线上禁用
```

2、实例

```
mysql> drop table t9;
Query OK, 0 rows affected (0.18 sec)
```

7、查询表

```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| t1             |
+----------------+
1 row in set (0.00 sec)
```

8、复制表结构

```
mysql> create table t8 like t1;
Query OK, 0 rows affected (0.33 sec)
```

9、操作表数据行

1、增加数据

1、语法

```
insert into 表名 (列1, 列2) values (值1,'值2');
```

2、实例

```
# 暴力复制
mysql> insert into t1 (name) select name from t8;
Query OK, 4 rows affected (0.09 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

2、删除数据

1、delete from 表名 where 条件;

```
mysql> insert into t1 (id, name) values (1, 'qf1');
mysql> insert into t1 (id, name) values (2, 'qf2');
mysql> insert into t1 (id, name) values (3, 'qf3');
mysql> insert into t1 (id, name) values (4, 'qf4');

mysql> delete from t1 where id=1;
mysql> delete from t1 where id>1;
mysql> delete from t1 where id>=1;
mysql> delete from t1 where id<1;
mysql> delete from t1 where id<=1;
mysql> delete from t1 where id>=1 and id<10;
Query OK, 1 row affected (0.06 sec)
```

2、truncate 表名; # 没有where条件的

```
mysql> truncate t1;
Query OK, 0 rows affected (0.25 sec)

mysql> select * from t1;
Empty set (0.00 sec)

mysql> insert into t1 (id, name) values (1, 'qf1');
Query OK, 1 row affected (0.06 sec)

mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
|    1 | qf1  |
+------+------+
1 row in set (0.00 sec)
```

3、区别

- delete 之后，插入数据从上一次主键自增加1开始， truncate则是从1开始
- delete 删除， 是一行一行的删除， truncate：全选删除 truncate删除的速度是高于delete的

10、修改数据

1、语法

```
update 表名 set 列名1=新值1,列名2=新值2 where 条件;
```

2、实例

```
mysql> insert into t1 (id, name) values (1, 'qf1');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 (id, name) values (2, 'qf2');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 (id, name) values (3, 'qf3');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 (id, name) values (4, 'qf4');
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
|    1 | qf1  |
|    2 | qf2  |
|    3 | qf3  |
|    4 | qf4  |
+------+------+
4 rows in set (0.00 sec)

mysql> update t1 set name='qfx' where id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from t1 ;
+------+------+
| id   | name |
+------+------+
|    1 | qfx  |
|    2 | qf2  |
|    3 | qf3  |
|    4 | qf4  |
+------+------+
4 rows in set (0.00 sec)
```

11、查询数据

1、语法

```
select 列1, 列2 from 表名;   （*代表查询所有的列）
select * from 表名;   （*代表查询所有的列）
```

2、实例

```
mysql> select * from t1;
+------+------+------+
| id   | name | age  |
+------+------+------+
|    1 | qfx  | NULL |
|    2 | qf2  | NULL |
|    3 | qf3  | NULL |
|    4 | qf4  | NULL |
|    5 | qf3  | NULL |
+------+------+------+
6 rows in set (0.00 sec)


# between..and...: 取值范围是闭区间

mysql> select * from t1 where id between 2 and 3;
+------+------+------+
| id   | name | age  |
+------+------+------+
|    2 | qf2  | NULL |
|    3 | qf3  | NULL |
+------+------+------+
2 rows in set (0.00 sec)

# 避免重复DISTINCT
mysql> insert into t1 (id, name) values (5, 'qf3');
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+------+------+
| id   | name |
+------+------+
|    1 | qfx  |
|    2 | qf2  |
|    3 | qf3  |
|    4 | qf4  |
|    5 | qf3  |
+------+------+
5 rows in set (0.00 sec)

mysql> select distinct name from t1;
+------+
| name |
+------+
| qfx  |
| qf2  |
| qf3  |
| qf4  |
+------+
4 rows in set (0.00 sec)

# 通过四则运算查询 （不要用）
mysql> alter table t1 add age int;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> insert into t1 (id, name,age) values (5, 'qf3',10);
```

```
Query OK, 1 row affected (0.01 sec)

mysql> select * from t1;
+------+------+------+
| id   | name | age  |
+------+------+------+
|    1 | qfx  | NULL |
|    2 | qf2  | NULL |
|    3 | qf3  | NULL |
|    4 | qf4  | NULL |
|    5 | qf3  | NULL |
|    5 | qf3  |   10 |
+------+------+------+
6 rows in set (0.00 sec)

mysql> select name, age*10 from t1;
+------+--------+
| name | age*10 |
+------+--------+
| qfx  |   NULL |
| qf2  |   NULL |
| qf3  |   NULL |
| qf4  |   NULL |
| qf3  |   NULL |
| qf3  |    100 |
+------+--------+
6 rows in set (0.00 sec)

mysql>  select name, age*10 as age from t1;
+------+------+
| name | age  |
+------+------+
| qfx  | NULL |
| qf2  | NULL |
| qf3  | NULL |
| qf4  | NULL |
| qf3  | NULL |
| qf3  |  100 |
+------+------+
6 rows in set (0.00 sec)

mysql> select * from t1 where id in (3,5,6);
+------+------+------+
| id   | name | age  |
+------+------+------+
|    3 | qf3  | NULL |
|    5 | qf3  | NULL |
|    5 | qf3  |   10 |
+------+------+------+
3 rows in set (0.00 sec)

# like : 模糊查询
# 以x开头:
mysql> select * from t1 where name like 'q%';
+------+------+------+
| id   | name | age  |
+------+------+------+
|    1 | qfx  | NULL |
```

```
|    2 | qf2  | NULL |
|    3 | qf3  | NULL |
|    4 | qf4  | NULL |
|    5 | qf3  | NULL |
|    5 | qf3  |   10 |
+------+------+------+
6 rows in set (0.00 sec)

# 以x结尾
mysql> select * from t1 where name like '%3';
+------+------+------+
| id   | name | age  |
+------+------+------+
|    3 | qf3  | NULL |
|    5 | qf3  | NULL |
|    5 | qf3  |   10 |
+------+------+------+
3 rows in set (0.00 sec)

# 包含x的:
mysql> select * from t1 where name like '%2%';
+------+------+------+
| id   | name | age  |
+------+------+------+
|    2 | qf2  | NULL |
+------+------+------+
1 row in set (0.00 sec)
```

3、查询时用'Name Is Null' 作为条件

```
mysql>create table t8(
id int auto_increment primary key,
name varchar(32),
email varchar(32)
)charset=utf8;

mysql>insert into t8(email) values ('qfedu');

mysql> select * from t8;
+----+------+-------+
| id | name | email |
+----+------+-------+
|  1 | NULL | qfedu |
+----+------+-------+
1 row in set (0.01 sec)

mysql> select * from t8 where name is null;
+----+------+-------+
| id | name | email |
+----+------+-------+
|  1 | NULL | qfedu |
+----+------+-------+
1 row in set (0.00 sec)
```

4、查询时用'Name='' '作为查询条件

```
mysql> create table t9(
id int auto_increment primary key,
name varchar(32) not null default '',
email varchar(32) not null default ''
)charset=utf8;

mysql> insert into t9 (email) values ('qfedu');

mysql> select * from t9;
+----+------+-------+
| id | name | email |
+----+------+-------+
|  1 |      | qfedu |
+----+------+-------+
1 row in set (0.00 sec)

mysql> select * from t9 where name='';
+----+------+-------+
| id | name | email |
+----+------+-------+
|  1 |      | qfedu |
+----+------+-------+
1 row in set (0.01 sec)
```

12、单表操作

1、单表查询的语法：

```
select 字段1，字段2 from 表名
                        where 条件
                        group by field
                        having 筛选
                        order by field
                        limit 限制条数
```

2、分组：Group By

- 分组指的是：将所有记录按照某个相同字段进行归类，比如针对员工信息表的职位分组，或者按照性别进行分组等
- 用法：select 聚合函数，字段名 from 表名 group by 分组的字段；
- group by 是分组的关键词， 必须和聚合函数 一起出现
- where 条件语句和 group by 分组语句的先后顺序：
- where > group by > having

1、创建表

```
create table qfedu(
id int not null unique auto_increment,
name varchar(20) not null,
gender enum('male','female') not null default 'male',
age int(3) unsigned not null default 28,
hire_date date not null,
post varchar(50),
post_comment varchar(100),
salary double(15,2),
office int,
depart_id int
);
```

2、插入数据

- 教学部 1，销售部门 2 ， 运营部门 3.

```
insert into qfedu(name,gender,age,hire_date,post,salary,office,depart_id) values
('egon','male',18,'20170301','www.qfedu.com',7300.33,401,1),
('yangsheng','male',78,'20150302','teacher',1000000.31,401,1),
('yangqiang','male',81,'20130305','teacher',8300,401,1),
('liuchao','male',73,'20140701','teacher',3500,401,1),
('luoyinsheng','male',28,'20121101','teacher',2100,401,1),
('nana','female',18,'20110211','teacher',9000,401,1),
('jinxin','male',18,'19000301','teacher',30000,401,1),
('xingdian','male',48,'20101111','teacher',10000,401,1),

('meiling','female',48,'20150311','sale',3000.13,402,2),
('lili','female',38,'20101101','sale',2000.35,402,2),
('xiaomei','female',18,'20110312','sale',1000.37,402,2),
('zhenzhen','female',18,'20160513','sale',3000.29,402,2),
('qianqian','female',28,'20170127','sale',4000.33,402,2),

('zhangye','male',28,'20160311','operation',10000.13,403,3),
('xiaolong','male',18,'19970312','operation',20000,403,3),
('zhaowei','female',18,'20130311','operation',19000,403,3),
('wujing','male',18,'20150411','operation',18000,403,3),
('liying','female',18,'20140512','operation',17000,403,3)
;
```

1、以性别为例， 进行分组， 统计一下男生和女生的人数是多少个：

```
mysql> select count(id), gender from qfedu group by gender;
+-----------+--------+
| count(id) | gender |
+-----------+--------+
|        10 | male   |
|         8 | female |
+-----------+--------+
2 rows in set (0.00 sec)
```

2、对部门进行分组， 求出每个部门年龄最大的那个人

- 注意： 出现 sql_mode=only_full_group_by 错误 需修改 配置文件/ etc/my.cnf [mysqld] 段添加 sql模式，重启 mysqld

```
sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_
ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
```

```
mysql> select depart_id,name, max(age) from qfedu group by depart_id;
+-----------+---------+----------+
| depart_id | name    | max(age) |
+-----------+---------+----------+
|         1 | egon    |       81 |
|         2 | meiling |       48 |
|         3 | zhangye |       28 |
+-----------+---------+----------+
3 rows in set (0.00 sec)
```

3、Min : 求最小的

4、Sum： 求和

```
mysql> select depart_id, sum(age) from qfedu group by depart_id;
+-----------+----------+
| depart_id | sum(age) |
+-----------+----------+
|         1 |      362 |
|         2 |      150 |
|         3 |      100 |
+-----------+----------+
3 rows in set (0.00 sec)
```

5、Count： 计数

```
mysql> select depart_id,count(depart_id) from qfedu group by depart_id;
+-----------+-----------------+
| depart_id | count(depart_id) |
+-----------+-----------------+
|         1 |               8 |
|         2 |               5 |
|         3 |               5 |
+-----------+-----------------+
3 rows in set (0.00 sec)
```

6、Avg： 平均数

```
mysql> select depart_id, avg(age) from qfedu group by depart_id;
+-----------+----------+
| depart_id | avg(age) |
+-----------+----------+
|         1 |  45.2500 |
|         2 |  30.0000 |
|         3 |  20.0000 |
+-----------+----------+
3 rows in set (0.00 sec)
```

3、Having

- Having 用于对 Group By 之后的数据进行进一步的筛选

```
mysql> select depart_id, avg(age) from qfedu group by depart_id having
avg(age)>35;
+-----------+----------+
| depart_id | avg(age) |
+-----------+----------+
|         1 |  45.2500 |
+-----------+----------+
1 row in set (0.01 sec)
```

4、Order By: Order By 字段名 Asc（升序）/Desc(降序)

- 对多个字段进行排序：
- age asc, depart_id desc; 表示先对age进行降序，再把age相等的行按部门号进行升序排列

```
mysql> select * from qfedu order by age asc, depart_id desc;
+----+-------------+--------+-----+------------+--------------+--------------+-
----------+--------+-----------+
| id | name        | gender | age | hire_date  | post         | post_comment |
salary     | office | depart_id |
+----+-------------+--------+-----+------------+--------------+--------------+-
----------+--------+-----------+
| 18 | liying      | female |  18 | 2014-05-12 | operation    | NULL         |
  17000.00 |    403 |         3 |
| 17 | wujing      | male   |  18 | 2015-04-11 | operation    | NULL         |
  18000.00 |    403 |         3 |
| 16 | zhaowei     | female |  18 | 2013-03-11 | operation    | NULL         |
  19000.00 |    403 |         3 |
| 15 | xiaolong    | male   |  18 | 1997-03-12 | operation    | NULL         |
  20000.00 |    403 |         3 |
| 11 | xiaomei     | female |  18 | 2011-03-12 | sale         | NULL         |
   1000.37 |    402 |         2 |
| 12 | zhenzhen    | female |  18 | 2016-05-13 | sale         | NULL         |
   3000.29 |    402 |         2 |
|  6 | nana        | female |  18 | 2011-02-11 | teacher      | NULL         |
   9000.00 |    401 |         1 |
|  7 | jinxin      | male   |  18 | 1900-03-01 | teacher      | NULL         |
  30000.00 |    401 |         1 |
|  1 | egon        | male   |  18 | 2017-03-01 | www.qfedu.com | NULL         |
   7300.33 |    401 |         1 |
| 14 | zhangye     | male   |  28 | 2016-03-11 | operation    | NULL         |
  10000.13 |    403 |         3 |
| 13 | qianqian    | female |  28 | 2017-01-27 | sale         | NULL         |
   4000.33 |    402 |         2 |
|  5 | luoyinsheng | male   |  28 | 2012-11-01 | teacher      | NULL         |
   2100.00 |    401 |         1 |
| 10 | lili        | female |  38 | 2010-11-01 | sale         | NULL         |
   2000.35 |    402 |         2 |
|  9 | meiling     | female |  48 | 2015-03-11 | sale         | NULL         |
   3000.13 |    402 |         2 |
|  8 | xingdian    | male   |  48 | 2010-11-11 | teacher      | NULL         |
  10000.00 |    401 |         1 |
|  4 | liuchao     | male   |  73 | 2014-07-01 | teacher      | NULL         |
   3500.00 |    401 |         1 |
```

```
|  2 | yangsheng   | male   |  78 | 2015-03-02 | teacher       | NULL         |
1000000.31 |    401 |         1 |
|  3 | yangqiang   | male   |  81 | 2013-03-05 | teacher       | NULL         |
   8300.00 |    401 |         1 |
+----+-------------+--------+-----+------------+---------------+--------------+-
----------+--------+-----------+
18 rows in set (0.00 sec)
```

- select * from qfedu order by depart_id asc, age desc;

```
mysql> select * from qfedu order by depart_id asc, age desc;
+----+-------------+--------+-----+------------+---------------+--------------+-
----------+--------+-----------+
| id | name        | gender | age | hire_date  | post          | post_comment |
salary     | office | depart_id |
+----+-------------+--------+-----+------------+---------------+--------------+-
----------+--------+-----------+
|  3 | yangqiang   | male   |  81 | 2013-03-05 | teacher       | NULL         |
   8300.00 |    401 |         1 |
|  2 | yangsheng   | male   |  78 | 2015-03-02 | teacher       | NULL         |
1000000.31 |    401 |         1 |
|  4 | liuchao     | male   |  73 | 2014-07-01 | teacher       | NULL         |
   3500.00 |    401 |         1 |
|  8 | xingdian    | male   |  48 | 2010-11-11 | teacher       | NULL         |
  10000.00 |    401 |         1 |
|  5 | luoyinsheng | male   |  28 | 2012-11-01 | teacher       | NULL         |
   2100.00 |    401 |         1 |
|  6 | nana        | female |  18 | 2011-02-11 | teacher       | NULL         |
   9000.00 |    401 |         1 |
|  7 | jinxin      | male   |  18 | 1900-03-01 | teacher       | NULL         |
  30000.00 |    401 |         1 |
|  1 | egon        | male   |  18 | 2017-03-01 | www.qfedu.com | NULL         |
   7300.33 |    401 |         1 |
|  9 | meiling     | female |  48 | 2015-03-11 | sale          | NULL         |
   3000.13 |    402 |         2 |
| 10 | lili        | female |  38 | 2010-11-01 | sale          | NULL         |
   2000.35 |    402 |         2 |
| 13 | qianqian    | female |  28 | 2017-01-27 | sale          | NULL         |
   4000.33 |    402 |         2 |
| 11 | xiaomei     | female |  18 | 2011-03-12 | sale          | NULL         |
   1000.37 |    402 |         2 |
| 12 | zhenzhen    | female |  18 | 2016-05-13 | sale          | NULL         |
   3000.29 |    402 |         2 |
| 14 | zhangye     | male   |  28 | 2016-03-11 | operation     | NULL         |
  10000.13 |    403 |         3 |
| 15 | xiaolong    | male   |  18 | 1997-03-12 | operation     | NULL         |
  20000.00 |    403 |         3 |
| 16 | zhaowei     | female |  18 | 2013-03-11 | operation     | NULL         |
  19000.00 |    403 |         3 |
| 17 | wujing      | male   |  18 | 2015-04-11 | operation     | NULL         |
  18000.00 |    403 |         3 |
| 18 | liying      | female |  18 | 2014-05-12 | operation     | NULL         |
  17000.00 |    403 |         3 |
+----+-------------+--------+-----+------------+---------------+--------------+-
----------+--------+-----------+
18 rows in set (0.00 sec)
```

5、Limit 分页： Limit Offset, Size

- offset 表示 行数据索引； size 表示取多少条数据
- 从第 offset 行开始，取 size 行数据。

```
mysql> select * from qfedu limit 0,10;
+----+-------------+--------+-----+------------+---------------+--------------+-
-----------+--------+-----------+
| id | name        | gender | age | hire_date  | post          | post_comment |
salary     | office | depart_id |
+----+-------------+--------+-----+------------+---------------+--------------+-
-----------+--------+-----------+
|  1 | egon        | male   |  18 | 2017-03-01 | www.qfedu.com | NULL         |
   7300.33 |    401 |         1 |
|  2 | yangsheng   | male   |  78 | 2015-03-02 | teacher       | NULL         |
1000000.31 |    401 |         1 |
|  3 | yangqiang   | male   |  81 | 2013-03-05 | teacher       | NULL         |
   8300.00 |    401 |         1 |
|  4 | liuchao     | male   |  73 | 2014-07-01 | teacher       | NULL         |
   3500.00 |    401 |         1 |
|  5 | luoyinsheng | male   |  28 | 2012-11-01 | teacher       | NULL         |
   2100.00 |    401 |         1 |
|  6 | nana        | female |  18 | 2011-02-11 | teacher       | NULL         |
   9000.00 |    401 |         1 |
|  7 | jinxin      | male   |  18 | 1900-03-01 | teacher       | NULL         |
  30000.00 |    401 |         1 |
|  8 | xingdian    | male   |  48 | 2010-11-11 | teacher       | NULL         |
  10000.00 |    401 |         1 |
|  9 | meiling     | female |  48 | 2015-03-11 | sale          | NULL         |
   3000.13 |    402 |         2 |
| 10 | lili        | female |  38 | 2010-11-01 | sale          | NULL         |
   2000.35 |    402 |         2 |
+----+-------------+--------+-----+------------+---------------+--------------+-
-----------+--------+-----------+
10 rows in set (0.00 sec)
```

- 从第6行开始取10行：

```
mysql> select * from qfedu limit 6,10;
+----+----------+--------+-----+------------+----------+--------------+-------
---+--------+-----------+
| id | name     | gender | age | hire_date  | post     | post_comment | salary
   | office | depart_id |
+----+----------+--------+-----+------------+----------+--------------+-------
---+--------+-----------+
|  7 | jinxin   | male   |  18 | 1900-03-01 | teacher  | NULL         |
30000.00 |    401 |         1 |
|  8 | 成龙     | male   |  48 | 2010-11-11 | teacher  | NULL         |
10000.00 |    401 |         1 |
|  9 | 歪歪     | female |  48 | 2015-03-11 | sale     | NULL         |
 3000.13 |    402 |         2 |
| 10 | 丫丫     | female |  38 | 2010-11-01 | sale     | NULL         |
 2000.35 |    402 |         2 |
| 11 | 丁丁     | female |  18 | 2011-03-12 | sale     | NULL         |
 1000.37 |    402 |         2 |
```

```
| 12 | 星星      | female |  18 | 2016-05-13 | sale      | NULL         |
 3000.29 |    402 |         2 |
| 13 | 格格      | female |  28 | 2017-01-27 | sale      | NULL         |
 4000.33 |    402 |         2 |
| 14 | 张野      | male   |  28 | 2016-03-11 | operation | NULL         |
10000.13 |    403 |         3 |
| 15 | 程咬金    | male   |  18 | 1997-03-12 | operation | NULL         |
20000.00 |    403 |         3 |
| 16 | 程咬银    | female |  18 | 2013-03-11 | operation | NULL         |
19000.00 |    403 |         3 |
+----+----------+--------+-----+------------+-----------+--------------+-------
---+--------+-----------+
10 rows in set (0.00 sec)
```

13、多表操作

- 概念：当在查询时，所需要的数据不在一张表中，可能在两张表或多张表中。此时需要同时操作这些表。即关联查询。
- 等值连接：在做多张表查询时，这些表中应该存在着有关联的两个字段。使用某一张表中的一条记录与另外一张表通过相关联的两个字段进行匹配，组合成一条记录。
- 笛卡尔积：在做多张表查询时，使用某一张表中的每一条记录都与另外一张表的所有记录进行组合。比如表A有x条，表B有y条，最终组合数为x*y，这个值就是笛卡尔积，通常没有意义。
- 内连接：只要使用了join on。就是内连接。查询效果与等值连接一样。
- 用法：

```
表A [inner] join  表B  on 关联条件
```

- 外连接：在做多张表查询时，所需要的数据，除了满足关联条件的数据外，还有不满足关联条件的数据。此时需要使用外连接。

1. 驱动表(主表)：除了显示满足条件的数据，还需要显示不满足条件的数据的表
2. 从表(副表)：只显示满足关联条件的数据的表
3. 外连接分为三种

> 左外连接：表A left [outer] join 表B on 关联条件，表A是驱动表，表B是从表 右外连接：表A right [outer] join 表B on 关联条件，表B是驱动表，表A是从表 全外连接：表A full [outer] join 表B on 关联条件，两张表的数据不管满不满足条件，都做显示。

- 自连接：在多张表进行关联查询时，这些表的表名是同一个。即自连接。
- 外键： 占用空间少，方便修改数据

1、一对多

1、语法

```
constraint 外键名 foreign key (被约束的字段) references 约束的表(约束的字段)
```

2、实例

```
mysql> create table dep(
id int auto_increment primary key,
name varchar(32) not null default ''
```

```
)charset=utf8;

mysql> insert into dep (name) values ('研发部'),('运维部'),('行政部'),('市场部');

mysql> create table userinfo (
id int auto_increment primary key,
name varchar(32) not null default '',
depart_id int not null default 1,
constraint fk_user_depart foreign key (depart_id) references dep(id)
)charset utf8;

mysql> insert into userinfo (name, depart_id) values ('qfedu a',1);
mysql> insert into userinfo (name, depart_id) values ('qfedu b',2);
mysql> insert into userinfo (name, depart_id) values ('qfedu c',3);
mysql> insert into userinfo (name, depart_id) values ('qfedu d',4);
mysql> insert into userinfo (name, depart_id) values ('qfedu e',1);
mysql> insert into userinfo (name, depart_id) values ('qfedu f',2);
mysql> insert into userinfo (name, depart_id) values ('qfedu g',3);
# 以上7行符合外键要求，所以能插入不报错，但下边一行插入时会报错
mysql> insert into userinfo (name, depart_id) values ('qfedu h',5);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
'mysql> insert into userinfo (name, depart_id) values ('qfedu h',5)' at line 1
```

2、多对多

1、创建男生表

```
mysql> create table boy(
id int auto_increment primary key,
bname varchar(32) not null default ''
)charset=utf8;

mysql> insert into boy (bname) values ('xiaoming'),('xiaogang'),('xiaoqiang');
```

2、创建女生表

```
mysql> create table girl(
id int auto_increment primary key,
gname varchar(32) not null default ''
)charset=utf8;

mysql> insert into girl (gname) values ('xiaohong'),('xiaoli'),('xiaojiao');
```

3、创建关联表

```
mysql> create table b2g(
id int auto_increment primary key,
bid int not null default 1,
gid int not null default 0,
constraint fk_b2g_boy foreign key (bid) references boy(id),
constraint fk_b2g_girl foreign key (gid) references girl(id)
)charset utf8;

mysql> insert into b2g (bid, gid) values (1,1),(1,2),(2,3),(3,3),(2,2);
```

4、用到 left jion

```
mysql> select * from boy left join b2g on boy.id=b2g.bid left join girl on
girl.id=b2g.gid;
+----+-----------+------+------+------+------+----------+
| id | bname     | id   | bid  | gid  | id   | gname    |
+----+-----------+------+------+------+------+----------+
|  1 | xiaoming  |    1 |    1 |    1 |    1 | xiaohong |
|  1 | xiaoming  |    2 |    1 |    2 |    2 | xiaoli   |
|  2 | xiaogang  |    5 |    2 |    2 |    2 | xiaoli   |
|  2 | xiaogang  |    3 |    2 |    3 |    3 | xiaojiao |
|  3 | xiaoqiang |    4 |    3 |    3 |    3 | xiaojiao |
+----+-----------+------+------+------+------+----------+
5 rows in set (0.01 sec)
mysql> select bname, gname from boy left join b2g on boy.id=b2g.bid left join
girl on girl.id=b2g.gid;
+-----------+----------+
| bname     | gname    |
+-----------+----------+
| xiaoming  | xiaohong |
| xiaoming  | xiaoli   |
| xiaogang  | xiaoli   |
| xiaogang  | xiaojiao |
| xiaoqiang | xiaojiao |
+-----------+----------+
5 rows in set (0.00 sec)
```

3、一对一

1、创建员工信息表

```
mysql> create table user(
id int auto_increment primary key,
name varchar(32) not null default ''
)charset=utf8;

mysql> insert into user (name) values ('xiaoming'),('xiaogang'),('xiaoqiang');

mysql> select * from user;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | xiaoming  |
|  2 | xiaogang  |
|  3 | xiaoqiang |
+----+-----------+
3 rows in set (0.00 sec)
```

2、创建员工工资表

```
mysql> create table priv(
id int auto_increment primary key,
salary int not null default 0,
uid int not null default 1,
constraint fk_priv_user foreign key (uid) references user(id),
unique(uid)
```

```
)charset=utf8;

mysql> insert into priv (salary, uid) values (2000, 1),(2500,2),(3000,3);

mysql> select * from priv;
+----+--------+-----+
| id | salary | uid |
+----+--------+-----+
|  1 |   2000 |   1 |
|  2 |   2500 |   2 |
|  3 |   3000 |   3 |
+----+--------+-----+
3 rows in set (0.00 sec)
```

14、多表联查

```
mysql>  select userinfo.name as uname, dep.name as dname from userinfo left join
dep on depart_id=dep.id;
+---------+-----------+
| uname   | dname     |
+---------+-----------+
| qfedu a | 研发部    |
| qfedu e | 研发部    |
| qfedu b | 运维部    |
| qfedu f | 运维部    |
| qfedu c | 行政部    |
| qfedu g | 行政部    |
| qfedu d | 市场部    |
+---------+-----------+
7 rows in set (0.00 sec)

mysql> select userinfo.name as uname, dep.name as dname from userinfo left join
dep on depart_id=dep.id;
+---------+-----------+
| uname   | dname     |
+---------+-----------+
| qfedu a | 研发部    |
| qfedu e | 研发部    |
| qfedu b | 运维部    |
| qfedu f | 运维部    |
| qfedu c | 行政部    |
| qfedu g | 行政部    |
| qfedu d | 市场部    |
+---------+-----------+
7 rows in set (0.00 sec)
```

15、单表多表查询练习

1、创建班级表

```
mysql> create table class1(
cid int auto_increment primary key,
caption varchar(32) not null default ''
)charset=utf8;

mysql> insert into class1 (caption) values ('三年二班'),('一年三班'),('三年一班');
```

```
mysql> select * from class1;
+-----+--------------+
| cid | caption      |
+-----+--------------+
|   1 | 三年二班     |
|   2 | 一年三班     |
|   3 | 三年一班     |
+-----+--------------+
3 rows in set (0.00 sec)
```

2、创建学生表

```
mysql> create table stu(
sid int auto_increment primary key,
sname varchar(32) not null default '',
gender varchar(32) not null default '',
class_id int not null default 1,
constraint fk_stu_class1 foreign key (class_id) references class1(cid)
)charset=utf8;

mysql> insert into stu (sname, gender, class_id) values ('小薇','女',1),('小
雪','女',1),('小明','男',2);

mysql> select * from stu;
+-----+--------+--------+----------+
| sid | sname  | gender | class_id |
+-----+--------+--------+----------+
|   1 | 小薇   | 女     |        1 |
|   2 | 小雪   | 女     |        1 |
|   3 | 小明   | 男     |        2 |
+-----+--------+--------+----------+
3 rows in set (0.00 sec)
```

3、创建老师表

```
mysql> create table tea(
tid int auto_increment primary key,
tname varchar(32) not null default ''
)charset=utf8;

mysql> insert into tea (tname) values ('qfedu'),('Frank'),('Julie');

mysql> select * from tea;
+-----+-------+
| tid | tname |
+-----+-------+
|   1 | qfedu |
|   2 | Frank |
|   3 | Julie |
+-----+-------+
3 rows in set (0.00 sec)
```

4、创建课程表

```
mysql> create table cour(
cid int auto_increment primary key,
cname varchar(32) not null default '',
tea_id int not null default 1,
constraint fk_cour_tea foreign key (tea_id) references tea(tid)
)charset=utf8;

mysql> insert into cour(cname, tea_id) values ('生物',1),('体育',1),('物理',2);

mysql> select * from cour;
+-----+--------+--------+
| cid | cname  | tea_id |
+-----+--------+--------+
|   1 | 生物   |      1 |
|   2 | 体育   |      1 |
|   3 | 物理   |      2 |
+-----+--------+--------+
3 rows in set (0.00 sec)
```

5、成绩表

```
mysql> create table sco(
sid int auto_increment primary key,
stu_id int not null default 1,
cou_id int not null default 1,
number int not null default 0,
constraint fk_sco_stu foreign key (stu_id) references stu(sid),
constraint fk_sco_cour foreign key (cou_id) references cour(cid)
)charset utf8;

mysql> insert into sco (stu_id, cou_id, number) values (1,1,60),(1,2,59),
(2,2,100);

mysql> select * from sco;
+-----+--------+--------+--------+
| sid | stu_id | cou_id | number |
+-----+--------+--------+--------+
|   1 |      1 |      1 |     60 |
|   2 |      1 |      2 |     59 |
|   3 |      2 |      2 |    100 |
+-----+--------+--------+--------+
3 rows in set (0.00 sec)
```

6、查询所有大于60分的学生的姓名和学号 (DISTINCT: 去重)

```
mysql> select distinct stu.sid,stu.sname from sco left join stu on
stu.sid=stu_id where number>=60;
+------+--------+
| sid  | sname  |
+------+--------+
|    1 | 小薇   |
|    2 | 小雪   |
+------+--------+
2 rows in set (0.00 sec)
```

五、MySQL 数据库导入导出

1、数据导出方式

```
mysqldump -h IP -u 用户名 -p -d 数据库名 > 导出的文件名
```

1、数据库的导出

1、导出所有数据

```
[root@qfedu.com ~]# mysqldump -uroot -pQfedu.123com --all-databases >
/data/mysql.sql
```

2、导出数据库

```
[root@qfedu.com ~]# mysqldump -uroot -pQfedu.123com --databases qfedu >
/data/mysql1.sql
```

3、导出某张表

```
[root@qfedu.com ~]# mysqldump -uroot -pQfedu.123com mysql user >/data/mysql.sql
```

4、参数说明

```
--all-databases  , -A

导出全部数据库。

mysqldump  -uroot -p --all-databases

--all-tablespaces , -Y

导出全部表空间。

mysqldump  -uroot -p --all-databases --all-tablespaces

--no-tablespaces  , -y

不导出任何表空间信息。

mysqldump  -uroot -p --all-databases --no-tablespaces

--add-drop-database

每个数据库创建之前添加drop数据库语句。

mysqldump  -uroot -p --all-databases --add-drop-database

--add-drop-table

每个数据表创建之前添加drop数据表语句。(默认为打开状态，使用--skip-add-drop-table取消选项)
```

```
mysqldump  -uroot -p --all-databases  (默认添加drop语句)

mysqldump  -uroot -p --all-databases -skip-add-drop-table  (取消drop语句)

--add-locks

在每个表导出之前增加LOCK TABLES并且之后UNLOCK  TABLE。(默认为打开状态，使用--skip-add-locks取消选项)

mysqldump  -uroot -p --all-databases  (默认添加LOCK语句)

mysqldump  -uroot -p --all-databases -skip-add-locks   (取消LOCK语句)

--allow-keywords

允许创建是关键词的列名字。这由表名前缀于每个列名做到。

mysqldump  -uroot -p --all-databases --allow-keywords

--apply-slave-statements

在'CHANGE MASTER'前添加'STOP SLAVE'，并且在导出的最后添加'START SLAVE'。

mysqldump  -uroot -p --all-databases --apply-slave-statements

--character-sets-dir

字符集文件的目录

mysqldump  -uroot -p --all-databases  --character-sets-dir=/usr/local/mysql/share/mysql/charsets

--comments

附加注释信息。默认为打开，可以用--skip-comments取消

mysqldump  -uroot -p --all-databases  (默认记录注释)

mysqldump  -uroot -p --all-databases --skip-comments   (取消注释)

--compatible

导出的数据将和其它数据库或旧版本的MySQL 相兼容。值可以为ansi、mysql323、mysql40、postgresql、oracle、mssql、db2、maxdb、no_key_options、no_tables_options、no_field_options等，

要使用几个值，用逗号将它们隔开。它并不保证能完全兼容，而是尽量兼容。

mysqldump  -uroot -p --all-databases --compatible=ansi

--compact

导出更少的输出信息(用于调试)。去掉注释和头尾等结构。可以使用选项：--skip-add-drop-table  --skip-add-locks --skip-comments --skip-disable-keys

mysqldump  -uroot -p --all-databases --compact

--complete-insert,  -c
```

使用完整的insert语句(包含列名称)。这么做能提高插入效率，但是可能会受到max_allowed_packet参数的影响而导致插入失败。

```
mysqldump  -uroot -p --all-databases --complete-insert
```

--compress, -C

在客户端和服务器之间启用压缩传递所有信息

```
mysqldump  -uroot -p --all-databases --compress
```

--create-options, -a

在CREATE TABLE语句中包括所有MySQL特性选项。(默认为打开状态)

```
mysqldump  -uroot -p --all-databases
```

--databases, -B

导出几个数据库。参数后面所有名字参量都被看作数据库名。

```
mysqldump  -uroot -p --databases test mysql
```

--debug

输出debug信息，用于调试。默认值为：d:t:o,/tmp/mysqldump.trace

```
mysqldump  -uroot -p --all-databases --debug
```

```
mysqldump  -uroot -p --all-databases --debug=” d:t:o,/tmp/debug.trace”
```

--debug-check

检查内存和打开文件使用说明并退出。

```
mysqldump  -uroot -p --all-databases --debug-check
```

--debug-info

输出调试信息并退出

```
mysqldump  -uroot -p --all-databases --debug-info
```

--default-character-set

设置默认字符集，默认值为utf8

```
mysqldump  -uroot -p --all-databases --default-character-set=latin1
```

--delayed-insert

采用延时插入方式（INSERT DELAYED）导出数据

```
mysqldump  -uroot -p --all-databases --delayed-insert
```

--delete-master-logs

master备份后删除日志. 这个参数将自动激活--master-data。

```
mysqldump  -uroot -p --all-databases --delete-master-logs
```

--disable-keys

对于每个表，用/*!40000 ALTER TABLE tbl_name DISABLE KEYS */;和/*!40000 ALTER TABLE tbl_name ENABLE KEYS */;语句引用INSERT语句。这样可以更快地导入dump出来的文件，因为它是在插入所有行后创建索引的。该选项只适合MyISAM表，默认为打开状态。

```
mysqldump  -uroot -p --all-databases
```

--dump-slave

该选项将导致主的binlog位置和文件名追加到导出数据的文件中。设置为1时，将会以CHANGE MASTER命令输出到数据文件；设置为2时，在命令前增加说明信息。该选项将会打开--lock-all-tables，除非--single-transaction被指定。该选项会自动关闭--lock-tables选项。默认值为0。

```
mysqldump  -uroot -p --all-databases --dump-slave=1

mysqldump  -uroot -p --all-databases --dump-slave=2
```

--events, -E

导出事件。

```
mysqldump  -uroot -p --all-databases --events
```

--extended-insert, -e

使用具有多个VALUES列的INSERT语法。这样使导出文件更小，并加速导入时的速度。默认为打开状态，使用--skip-extended-insert取消选项。

```
mysqldump  -uroot -p --all-databases

mysqldump  -uroot -p --all-databases--skip-extended-insert   (取消选项)
```

--fields-terminated-by

导出文件中忽略给定字段。与--tab选项一起使用，不能用于--databases和--all-databases选项

```
mysqldump  -uroot -p test test --tab="/home/mysql" --fields-terminated-by="#"
```

--fields-enclosed-by

输出文件中的各个字段用给定字符包裹。与--tab选项一起使用，不能用于--databases和--all-databases选项

```
mysqldump  -uroot -p test test --tab="/home/mysql" --fields-enclosed-by="#"
```

--fields-optionally-enclosed-by

输出文件中的各个字段用给定字符选择性包裹。与--tab选项一起使用，不能用于--databases和--all-databases选项

```
mysqldump  -uroot -p test test --tab="/home/mysql"  --fields-enclosed-by="#" --fields-optionally-enclosed-by  ="#"
```

--fields-escaped-by

输出文件中的各个字段忽略给定字符。与--tab选项一起使用，不能用于--databases和--all-databases选项

```
mysqldump  -uroot -p mysql user --tab="/home/mysql" --fields-escaped-by="#"
```

--flush-logs

开始导出之前刷新日志。

请注意：假如一次导出多个数据库(使用选项--databases或者--all-databases)，将会逐个数据库刷新日志。除使用--lock-all-tables或者--master-data外。在这种情况下，日志将会被刷新一次，相应的所以表同时被锁定。因此，如果打算同时导出和刷新日志应该使用--lock-all-tables 或者--master-data 和--flush-logs。

```
mysqldump  -uroot -p --all-databases --flush-logs
```

--flush-privileges

在导出mysql数据库之后，发出一条FLUSH PRIVILEGES 语句。为了正确恢复，该选项应该用于导出mysql数据库和依赖mysql数据库数据的任何时候。

```
mysqldump  -uroot -p --all-databases --flush-privileges
```

--force

在导出过程中忽略出现的SQL错误。

```
mysqldump  -uroot -p --all-databases --force
```

--help

显示帮助信息并退出。

```
mysqldump  --help
```

--hex-blob

使用十六进制格式导出二进制字符串字段。如果有二进制数据就必须使用该选项。影响到的字段类型有BINARY、VARBINARY、BLOB。

```
mysqldump  -uroot -p --all-databases --hex-blob
```

--host, -h

需要导出的主机信息

```
mysqldump  -uroot -p --host=localhost --all-databases
```

--ignore-table

不导出指定表。指定忽略多个表时，需要重复多次，每次一个表。每个表必须同时指定数据库和表名。例如：--ignore-table=database.table1 --ignore-table=database.table2 ……

```
mysqldump  -uroot -p --host=localhost --all-databases --ignore-table=mysql.user
```

--include-master-host-port

在--dump-slave产生的'CHANGE MASTER TO..'语句中增加'MASTER_HOST=<host>，MASTER_PORT=<port>'

```
mysqldump  -uroot -p --host=localhost --all-databases --include-master-host-port
```

--insert-ignore

在插入行时使用INSERT IGNORE语句.

```
mysqldump  -uroot -p --host=localhost --all-databases --insert-ignore
```

--lines-terminated-by

输出文件的每行用给定字符串划分。与--tab选项一起使用，不能用于--databases和--all-databases选项。

```
mysqldump  -uroot -p --host=localhost test test --tab="/tmp/mysql"  --lines-terminated-by="##"
```

--lock-all-tables， -x

提交请求锁定所有数据库中的所有表，以保证数据的一致性。这是一个全局读锁，并且自动关闭--single-transaction 和--lock-tables 选项。

```
mysqldump  -uroot -p --host=localhost --all-databases --lock-all-tables
```

--lock-tables， -l

开始导出前，锁定所有表。用READ LOCAL锁定表以允许MyISAM表并行插入。对于支持事务的表例如InnoDB和BDB，--single-transaction是一个更好的选择，因为它根本不需要锁定表。

请注意当导出多个数据库时，--lock-tables分别为每个数据库锁定表。因此，该选项不能保证导出文件中的表在数据库之间的逻辑一致性。不同数据库表的导出状态可以完全不同。

```
mysqldump  -uroot -p --host=localhost --all-databases --lock-tables
```

--log-error

附加警告和错误信息到给定文件

```
mysqldump  -uroot -p --host=localhost --all-databases  --log-error=/tmp/mysqldump_error_log.err
```

--master-data

该选项将binlog的位置和文件名追加到输出文件中。如果为1，将会输出CHANGE MASTER 命令；如果为2，输出的CHANGE MASTER命令前添加注释信息。该选项将打开--lock-all-tables 选项，除非--single-transaction也被指定（在这种情况下，全局读锁在开始导出时获得很短的时间；其他内容参考下面的--single-transaction选项）。该选项自动关闭--lock-tables选项。

```
mysqldump  -uroot -p --host=localhost --all-databases --master-data=1;

mysqldump  -uroot -p --host=localhost --all-databases --master-data=2;
```

--max_allowed_packet

服务器发送和接受的最大包长度。

```
mysqldump  -uroot -p --host=localhost --all-databases --max_allowed_packet=10240
```

```
--net_buffer_length
```

TCP/IP和socket连接的缓存大小。

```
mysqldump  -uroot -p --host=localhost --all-databases --net_buffer_length=1024
```

```
--no-autocommit
```

使用autocommit/commit 语句包裹表。

```
mysqldump  -uroot -p --host=localhost --all-databases --no-autocommit
```

```
--no-create-db,  -n
```

只导出数据，而不添加CREATE DATABASE 语句。

```
mysqldump  -uroot -p --host=localhost --all-databases --no-create-db
```

```
--no-create-info,  -t
```

只导出数据，而不添加CREATE TABLE 语句。

```
mysqldump  -uroot -p --host=localhost --all-databases --no-create-info
```

```
--no-data, -d
```

不导出任何数据，只导出数据库表结构。

```
mysqldump  -uroot -p --host=localhost --all-databases --no-data
```

```
--no-set-names,  -N
```

等同于--skip-set-charset

```
mysqldump  -uroot -p --host=localhost --all-databases --no-set-names
```

```
--opt
```

等同于--add-drop-table, --add-locks, --create-options, --quick, --extended-insert, --lock-tables, --set-charset, --disable-keys 该选项默认开启, 可以用--skip-opt禁用.

```
mysqldump  -uroot -p --host=localhost --all-databases --opt
```

```
--order-by-primary
```

如果存在主键，或者第一个唯一键，对每个表的记录进行排序。在导出MyISAM表到InnoDB表时有效，但会使得导出工作花费很长时间。

```
mysqldump  -uroot -p --host=localhost --all-databases --order-by-primary
```

```
--password, -p
```

连接数据库密码

--pipe(windows系统可用)

使用命名管道连接mysql

```
mysqldump  -uroot -p --host=localhost --all-databases --pipe
```

--port, -P

连接数据库端口号

--protocol

使用的连接协议，包括：tcp, socket, pipe, memory.

```
mysqldump  -uroot -p --host=localhost --all-databases --protocol=tcp
```

--quick, -q

不缓冲查询，直接导出到标准输出。默认为打开状态，使用--skip-quick取消该选项。

```
mysqldump  -uroot -p --host=localhost --all-databases
```

```
mysqldump  -uroot -p --host=localhost --all-databases --skip-quick
```

--quote-names,-Q

使用（`）引起表和列名。默认为打开状态，使用--skip-quote-names取消该选项。

```
mysqldump  -uroot -p --host=localhost --all-databases
```

```
mysqldump  -uroot -p --host=localhost --all-databases --skip-quote-names
```

--replace

使用REPLACE INTO 取代INSERT INTO.

```
mysqldump  -uroot -p --host=localhost --all-databases --replace
```

--result-file, -r

直接输出到指定文件中。该选项应该用在使用回车换行对（\\r\\n）换行的系统上（例如：DOS，Windows）。该选项确保只有一行被使用。

```
mysqldump  -uroot -p --host=localhost --all-databases --result-file=/tmp/mysqldump_result_file.txt
```

--routines, -R

导出存储过程以及自定义函数。

```
mysqldump  -uroot -p --host=localhost --all-databases --routines
```

--set-charset

添加'SET NAMES default_character_set'到输出文件。默认为打开状态，使用--skip-set-charset关闭选项。

```
mysqldump  -uroot -p --host=localhost --all-databases
```

```
mysqldump  -uroot -p --host=localhost --all-databases --skip-set-charset
```

```
--single-transaction
```

该选项在导出数据之前提交一个BEGIN SQL语句，BEGIN 不会阻塞任何应用程序且能保证导出时数据库的一致性状态。它只适用于多版本存储引擎，仅InnoDB。本选项和--lock-tables 选项是互斥的，因为LOCK TABLES 会使任何挂起的事务隐含提交。要想导出大表的话，应结合使用--quick 选项。

```
mysqldump  -uroot -p --host=localhost --all-databases --single-transaction
```

```
--dump-date
```

将导出时间添加到输出文件中。默认为打开状态，使用--skip-dump-date关闭选项。

```
mysqldump  -uroot -p --host=localhost --all-databases
```

```
mysqldump  -uroot -p --host=localhost --all-databases --skip-dump-date
```

```
--skip-opt
```

禁用-opt选项.

```
mysqldump  -uroot -p --host=localhost --all-databases --skip-opt
```

```
--socket,-S
```

指定连接mysql的socket文件位置，默认路径/tmp/mysql.sock

```
mysqldump  -uroot -p --host=localhost --all-databases --socket=/tmp/mysqld.sock
```

```
--tab,-T
```

为每个表在给定路径创建tab分割的文本文件。注意：仅仅用于mysqldump和mysqld服务器运行在相同机器上。

```
mysqldump  -uroot -p --host=localhost test test --tab="/home/mysql"
```

```
--tables
```

覆盖--databases (-B)参数，指定需要导出的表名。

```
mysqldump  -uroot -p --host=localhost --databases test --tables test
```

```
--triggers
```

导出触发器。该选项默认启用，用--skip-triggers禁用它。

```
mysqldump  -uroot -p --host=localhost --all-databases --triggers
```

```
--tz-utc
```

在导出顶部设置时区TIME_ZONE='+00:00' ，以保证在不同时区导出的TIMESTAMP 数据或者数据被移动其他时区时的正确性。

```
mysqldump  -uroot -p --host=localhost --all-databases --tz-utc
```

```
--user, -u
```

指定连接的用户名。

--verbose, --v

输出多种平台信息。

--version, -V

输出mysqldump版本信息并退出

--where, -w

只转储给定的WHERE条件选择的记录。请注意如果条件包含命令解释符专用空格或字符，一定要将条件引用起来。

```
mysqldump  -uroot -p --host=localhost --all-databases --where=" user='root'"
```

--xml, -X

导出XML格式.

```
mysqldump  -uroot -p --host=localhost --all-databases --xml
```

--plugin_dir

客户端插件的目录，用于兼容不同的插件版本。

```
mysqldump  -uroot -p --host=localhost --all-databases --
plugin_dir="/usr/local/lib/plugin"
```

--default_auth

客户端插件默认使用权限。

```
mysqldump  -uroot -p --host=localhost --all-databases --default-
auth="/usr/local/lib/plugin/<PLUGIN>"
```

2、数据库的导入

1、已经建好数据库，导入数据库文件

1、登录并进入数据库:

1、本地访问

```
[root@qfedu.com ~]# mysql -hlocalhost -uroot -pQfedu.123com
```

2、远程访问

```
[root@qfedu.com ~]# mysql -h192.168.182.120 -uroot -pQfedu.123com
```

3、参数解析

```
-h:表示host地址，本地直接使用localhost，远程需要使用ip地址
-u:表示user用户
-p:表示password密码
```

4、登录成功后执行导入命令source+文件路径

```
mysql> source /data/mysql.sql
```

2、不登录导入数据

```
[root@qfedu.com ~]# mysql -uroot -pQfedu.123com < /data/mysql.sql
```

2、没有数据库需要先创建数据库

```
[root@qfedu.com ~]# mysql -hlocalhost -uroot -pQfedu.123com # 进入mysql下面
mysql> create database qfedu;  # 创建数据库
mysql> show databases;         # 查看数据库列表
mysql> use qfedu;              # 进入qfedu数据库下面
mysql> show tables;            # 刚创建的数据库Qfedu.123com下面空没有表
mysql> source /data/mysql.sql  # 导入数据库表
mysql> show tables;            # 查看Qfedu.123com数据库下面的所有表,就可以看到表了
mysql> desc users;             # 查看表结构设计
mysql> select * from users;    # 查询所有的数据
mysql> exit;                   # 或者ctrl + c退出mysql
```