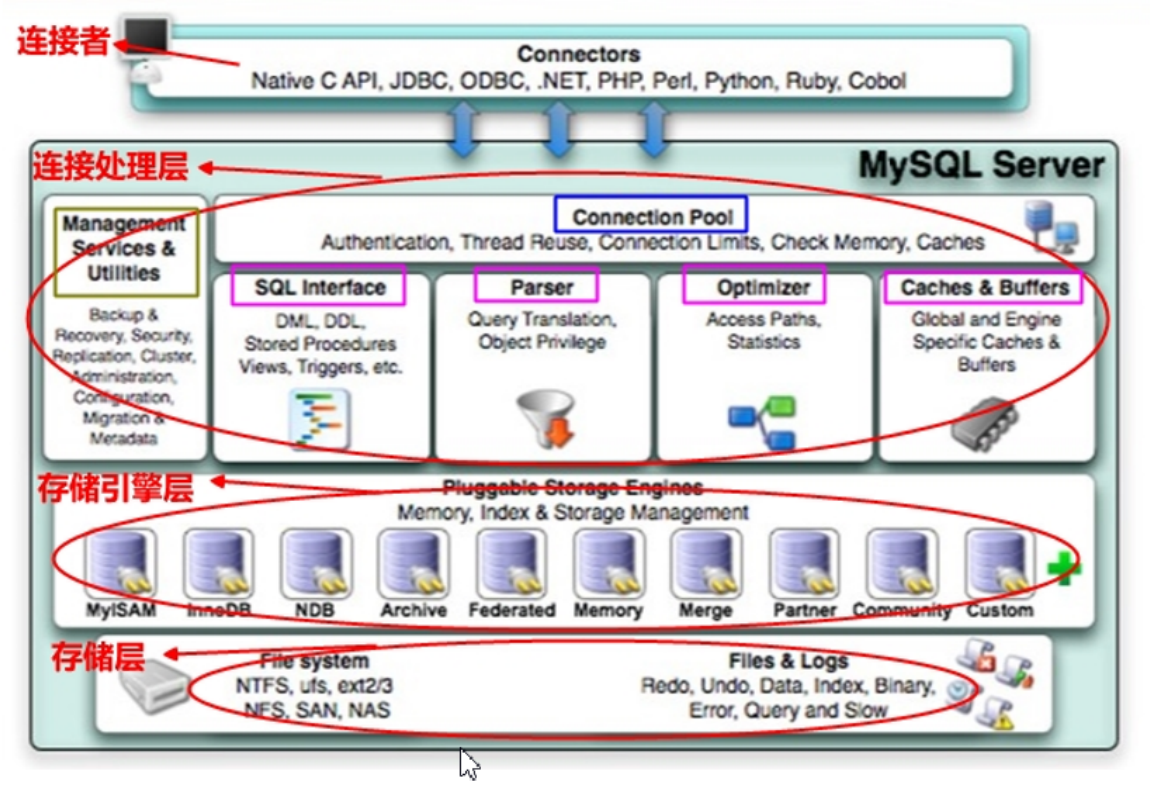
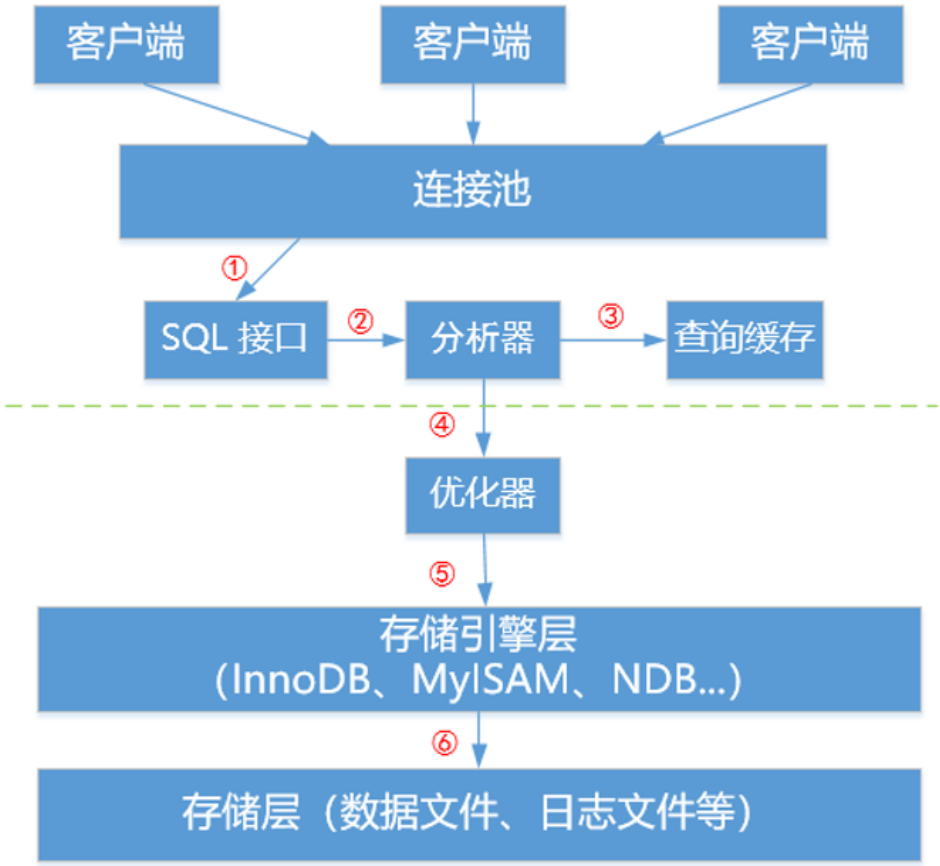
课程目标：

- 理解MySQL的基本体系结构
- 了解MySQL常见的日志文件及作用
- 熟悉基本的SQL语句（增删改查等)
- ==掌握MySQL的用户权限管理（重点）==

一、MySQL的体系结构



分解图:



1.1 客户端(连接者)

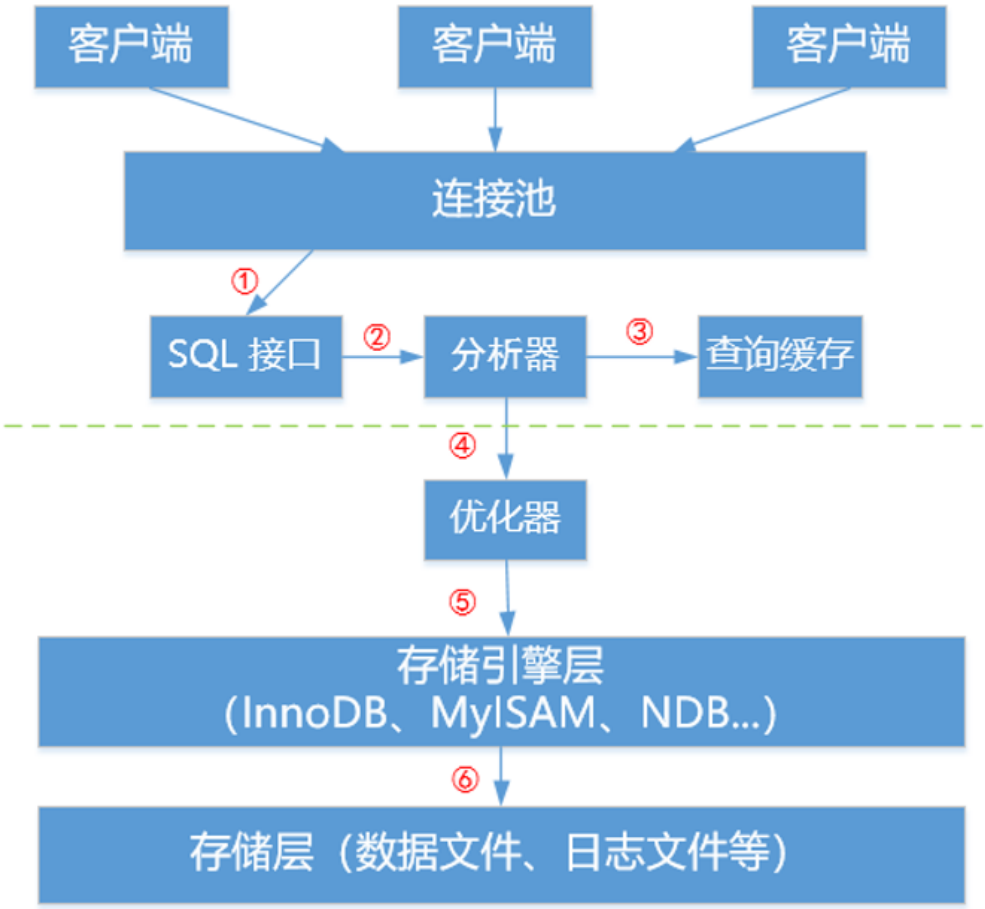
- MySQL的客户端并不单纯的指某个客户端软件
- MySQL的客户端可以是不同的编程语言(PHP/Python等)编写的应用程序
- MySQL的客户端还可以是一些API的接口

MySQL体系结构2-连接池

1.2 连接处理层

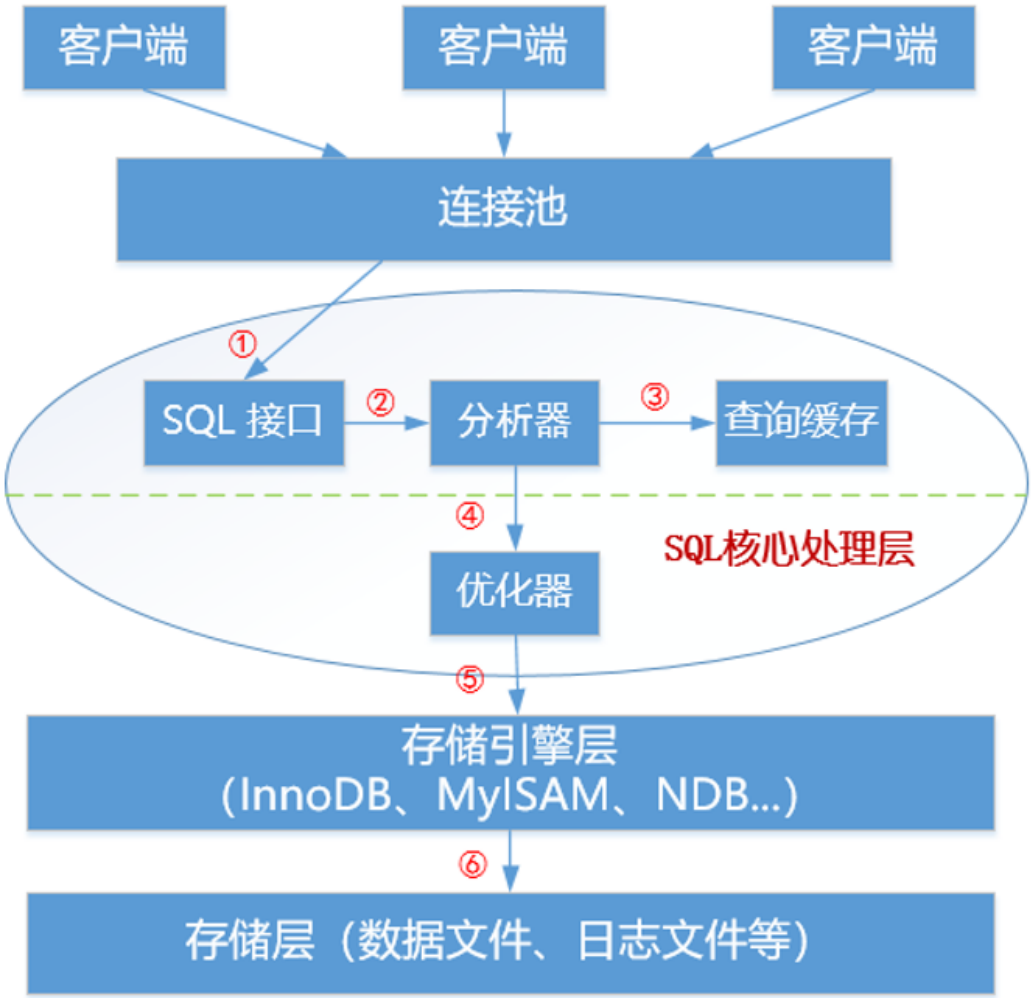
- **服务层(连接池)**

作用：管理和缓冲用户连接，为客户端请求做连接处理。



- **核心处理层（SQL处理）**

作用:接受用户的SQL请求，查询分析，权限处理，优化，结果缓存等。



1.3 存储引擎层

- 什么是存储引擎?

1. 存储引擎说白了就是==如何管理操作数据==（存储数据、如何更新、查询数据等）的==一种方法和机制==。
2. 在MySql数据库中提供了==多种存储引擎，==各个存储引擎的优势各不一样。
3. 用户可以根据不同的需求为**数据表**选择不同的存储引擎，也可以根据自己的需要编写自己的存储引擎。
4. 甚至一个库中不同的表使用不同的存储引擎，这些都是允许的。

- 常用的存储引擎有哪些?

  最常用的存储引擎是MyISAM和InnoDB。

MySQL5.5版本之前默认使用MyISAM引擎，它查询速度快，有较好的索引优化和数据压缩技术。但它不支持事务。适用于读多写少的应用场景，比如OLAP（在线分析处理系统）。
MySQL5.5版本开始默认使用InnoDB引擎，它是第一个支持拥有ACID特性事务的存储引擎，并且提供行级的锁定，应用相当广泛。为处理巨大数据量时的最大性能设计。比如OLTP(在线事务处理系统)。

InnoDB引擎由于该存储引擎在事务上具有优势，即支持具有提交、回滚及崩溃恢复能力等事务特性，所以，比MyISAM存储引擎占用更多的磁盘空间。因此当需要频繁的更新、删除操作，同时还对事务的完整性要求较高，需要实现并发控制，建议选择。

其他存储引擎：

```
NDB存储引擎：用于MySQL Cluster的集群存储引擎，提供数据层面的高可用性。
MEMORY存储引擎：MEMORY存储引擎存储数据的位置是内存，因此访问速度最快，但是安全上没有保障。适合于需要快速的访问或临时表。
BLACKHOLE存储引擎：黑洞存储引擎，写入的任何数据都会消失，应用于主备复制中的分发主库（中继slave）。
...
```

1.4 存储层

作用：

主要用来存储MySQL的一些数据文件，各种日志文件等物理文件。支持各种文件系统，如ext2~ext4，ntfs，nfs，nas等。

总结：

1. MySQL整体逻辑结构分为哪几层？
2. 每一层是如何工作的？
3. MySQL5.5版本以后默认的存储引擎是哪个？有什么特点？

二、MySQL数据库物理文件

1.1 常见的日志文件

日志类型	写入日志的信息
错误日志 error log	启动，运行或停止mysqld时遇到的问题
一般查询日志	建立客户端连接和从客户端收到的语句
二进制日志 binary log	更改数据的语句（也用于复制）
中继日志 relay log	从复制主服务器收到的数据更改
慢查询日志 slow query log	long_query_time执行时间超过几秒的查询
DDL日志（元数据日志）	由DDL语句执行的元数据操作

- **错误日志（erro log）**

  作用：存放数据库的启动、停止或运行时的错误信息，==用于排错==。

  - 默认是==开启==的，可以通过修改==my.cnf==文件自定义，如：

    `log_error=/path默认存在在$datadir/hostname.err`

- **一般查询日志（general query log）**

  作用：当客户端连接或断开时，服务器会将信息写入该日志，并记录从客户端收到的每一条SQL语句。当您怀疑客户端的错误并想知道客户端发送给mysqld的确切信息时，一般查询日志可能非常有用。

  - 默认情况下，一般查询日志是==被禁用==的。 如果要开启，可以使用以下参数：

```
general_log[={0|1}          0表示禁用，1表示开启.
```

- 默认情况下，系统会在数据目录下创建以host_name.log命名的一般查询日志。如果要自己指定，可以使用以下参数：

```
general_log_file=file_name      该参数用于指定一般查询日志的路径及文件名
log-output=[value,...]          该参数用于定义general log和slow log的输出目标

value=[TABLE|FILE|NONE],默认值为FILE
TABLE表示将日志记录到表中，general_log表或者slow_log表中
FILE表示将日志记录的文本文件中
NONE表示不记录到表或者文件

注意：
如果log-output=NONE，则即使启用了日志，也不会写入条目。
如果log-output不等于NONE,但是没有启用日志也不会写入条目。
```

- **慢查询日志(slow query log)**

  作用：慢查询日志记录的是一些SQL语句，可用于查找需要很长时间才能执行的查询，因此可用于优化。但是，检查一个很长的慢查询日志可能成为一项艰巨的任务。为了简化这一过程，您可以使用[mysqldumpslow](#)命令处理慢查询日志文件，以汇总日志中显示的查询。

  - 默认情况下慢查询日志是关闭的，可以使用以下参数开启

```
slow_query_log={1|0}        0代表关闭；1代表启用
slow_query_log_file=file_name
该参数用于指定慢查询日志的路径及文件名，默认$datadir/host_name-slow.log
log-output=[=value,...]          该参数用于定义日志的输出目标
long_query_time=n       单位是秒s，表示查询语句超过n秒被记录到slow log里，最小值为0，默认值为10
```

- **二进制日志(binary log)**

  作用：二进制日志记录==数据库的所有更改==操作（DDL/DML/DCL），不包含select或者show这类语句。

  - 用于==主从复制==中，master主服务器将二进制日志中的更改操作发送给slave从服务器，从服务器执行这些更改操作是的和主服务器的更改相同。
  - 用于==数据的恢复==操作。
  - 默认二进制日志是==关闭==的，可以使用`log-bin=xxx`参数开启
  - mysqlbinlog工具查看

- **中继日志（relay log）**

  作用：用于主从复制，master主服务器将自己的二进制日志发送给slave从服务器，slave先保存在自己的==中继日志中==，然后再执行自己本地的relay log里的sql达到数据库更改和master保持一致。

  - 默认中继日志没有开启，可以使用`relay-log`参数开启

1.2 常见的数据文件

1. **.frm文件:** 不论是什么存储引擎，每一个表都会有一个以表名命名的.frm文件，与表相关的==元数据（meta）信息==都存放在此文件中，包括==表结构的定义==信息等。
2. **.MYD文件:** myisam存储引擎专用，存放myisam表的数据（data）。每一个myisam表都会有一个.MYD文件与之呼应，同样存放在所属数据库的目录下 .

3. **.MYI文件:** 也是myisam存储引擎专用，存放myisam表的索引相关信息。对于myisam存储引擎来说，可以被缓存(cache)的内容主要就是来源于.MYI文件中。 每一个myisam表对应一个.MYI文件.
4. **.ibd文件和ibdata文件:** 存放innoDB的数据文件（包括索引）。innoDB存储引擎有两种表空间方式：独享表空间和共享表空间。
    - 独享表空间：使用.ibd文件来存放数据，且每个表一个.ibd文件。
    - 共享表空间：使用.ibdata文件，所有表共同使用一个（或多个，自行配置）.ibdata文件。
5. **db.opt文件:** 此文件在每一个==自建的库==里都会有，记录这个库的默认使用的==字符集和校验规==。

三、MySQL数据库概念名词

1. 什么是MySQL数据库?

数据库（database）： 操作系统或存储上的==数据文件的集合==。mysql数据库中，数据库文件可以是*.frm、*.MYD*、.MYI、*.ibd等结尾的文件， 不同存储引擎文件类型不同。

2. 什么是MySQL数据库实例?

数据库实例（instance）： 由==后台进程或者线程==以及一个==共享内存区==组成。共享内存可以被运行的后台线程所共享。 数据库实例才是真正操作数据库的。

注意：MySQL的后台守护程序mysqld是单进程多线程的工作模式。

3. 什么是MySQL数据库服务器?

数据库服务器(database server)：部署==安装数据库实例==的服务器。

4. 数据库和数据库实例之间的关系是什么?

通常情况下，数据库实例和数据库是一一对应的关系，也就是==一个数据库实例对应一个数据库==； 但是，在集群环境中存在==多个数据库实例共同使用一个数据库==。oracle RAC

四、MySQL中的SQL语句

1. 什么是SQL?

SQL 是 Structure Query Language(==结构化查询语言==)的缩写,它是使用==关系模型的数据库应== ==用语言==,由 IBM 在 20 世纪 70 年代开发出来,作为 IBM 关系数据库原型 System R 的原型关 系语言,实现了关系数据库中的信息检索。

20 世纪 80 年代初,美国国家标准局(ANSI)开始着手制定 SQL 标准,最早的 ANSI 标准于 1986 年完成,就被叫作 SQL-86。标准的出台使 SQL 作为标准关系数据库语言的地位得到了 加强。SQL 标准目前已几经修改更趋完善。

正是由于 SQL 语言的标准化,所以大多数关系型数据库系统都支持 SQL 语言,它已经发展 成为多种平台进行交互操作的底层会话语言。

2. SQL语句的分类

- DDL(Data Definition Languages)语句:

  ==数据定义语言==,这些语句定义了不同的数据段、数据库、表、列、索引等数据库对象的定义。常用的语句关键字主要包括 **create、drop、alter、rename、truncate**。

- DML(Data Manipulation Language)语句:

==数据操纵语句==,用于添加、删除、更新和查询数据库记录,并检查数据完整性,常用的语句关键字主要包括 **insert、delete、update**等。

- DCL(Data Control Language)语句:

  ==数据控制语句==,用于控制不同数据段直接的许可和访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的语句关键字包括 **grant、revoke** 等。

- DQL(Data Query Language)语句:

  数据查询语句，用于从一个或多个表中检索信息。主要的语句关键字包括 **select**

3. MySQL中如何求帮助

- 亘古不变的官当

  MySQL5.6官方文档

  MySQL5.7官方文档

- man文档

  man文档可以对mysql的一些基本工具及后台命令求帮助，比如：

```
[root@misshou ~]# man mysql
[root@misshou ~]# man mysql_install_db
[root@misshou ~]# man mysqldump
[root@misshou ~]# man mysqld
```

- MySQL的命令行求帮助

```
mysql> help;
mysql> ?
mysql> help create table;
mysql> ? contents       根据内容进行查找帮助
   Account Management
   Administration
   Data Definition
   Data Manipulation
   Data Types
   Functions
   Functions and Modifiers for Use with GROUP BY
   Geographic Features
   Language Structure
   Storage Engines
   Stored Routines
   Table Maintenance
   Transactions
   Triggers

mysql> ? Account Management 寻求账户管理的帮助（一级一级的向内部查）
mysql> ? CREATE USER
```

五、MySQL数据库基本操作

1. 常用的一些命令

```
1. 事务控制语句:
DML:
insert
update
delete

commit      提交
rollback        回滚

insert into t1...
insert into t2...
commit
rollback


mysql> show variables like 'autocommit';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| autocommit    | ON    |           //autocommit=ON表示自动提交
+---------------+-------+
1 row in set (0.00 sec)
mysql> set autocommit=OFF;          //临时关闭
Query OK, 0 rows affected (0.00 sec)

思考:mysql数据库不要自动提交,永久更改?
vim my.cnf
...
autocommit=0
或者
autocommit=OFF

2. 查看数据库的基本信息相关语句
mysql> \s 或者status      查看数据库的基本信息
--------------
mysql  Ver 14.14 Distrib 5.6.35, for linux-glibc2.5 (x86_64) using  EditLine wrapper

Connection id:      4
Current database:
Current user:       root@localhost
....

常用函数:
查看mysql支持字符加密函数:
select password('123');
select md5('123');
select sha1('123');
select encrypt('123');      基本上不用了

使用select来调度mysql中的常见函数:
select version();               当前数据库版本
```

```
select current_user();      当前用户
select current_time();      当前时间
select current_date();      当前日期
select now();                   当前日期时间

mysql> show databases;      查看所有数据库
mysql> show schemas;            查看所有数据库
mysql> show variables;      查看变量
mysql> show variables like '%char%';
mysql> show variables like '%data%';
mysql> show engines;            查看存储引擎


更改默认字符集为utf8mb4:
临时更改:
mysql> set character_set_database=utf8mb4;
Query OK, 0 rows affected (0.01 sec)

mysql> set character_set_server=utf8mb4;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables like '%char%';
mysql> show variables like 'collation_server';
+------------------+--------------------+
| Variable_name    | Value              |
+------------------+--------------------+
| collation_server | utf8mb4_general_ci |
+------------------+--------------------+

永久更改:
vim my.cnf
[mysqld]
...
character_set_server=utf8mb4
collation_server=utf8mb4_general_ci (可省略)
```

2. 数据库基本操作

```
create database db1;                                        创建db1库
create database db1 default charset gbk;                创建db1库并指定默认字符集
说明:不能创建相同名字的数据库!
create database if not exists db1 default character  set utf8;      如果存在不报错

show create database db1;               查看创建db1库的详细语句
alter database db1 default character set gbk;       更改db1库的默认字符集
drop database db1;          删除db1库
```

3. 数据表基本操作

```
use db1;                使用db1库,类似于进入到db1库里进行下面操作

show tables;        查看当前库有哪些表
```

```
create: 创建表
create table 表名 (字段1 数据类型(字符长度), 字段2, ...)
create table 表名 (字段1 数据类型(字符长度) 约束条件, 字段2, ...)

create table t1 (id int,name varchar(10),.....);
create table t1 (id int key,name varchar(20) not null,......)

desc t1;        查看表的结构
describe t1;

insert: 插入/增加记录
insert into t1 set id=3,name='li';
insert into t1 values(1,'zhang'),(2,'wang');
insert into t1 (id,name) values(3,'li');
insert into t1 select * from t2;

查询数据:
select * from t1;

delete: 删除记录
delete from t1;
delete from t1 where id>3;

update: 更新/修改记录
update 表名 set 字段1=新值,字段2=新值,... where  条件;

UPDATE table_anem SET column_name1 =new value1, column_name2 = value2, ...WHERE ... ;

mysql> update t3 set name='harry' where id=1;
mysql> update t3 set name='BBB',id=444 where id=4;


alter table: 修改表的属性
alter table t2 add id int first;            增加一列成为第一列
alter table t2 add id2 int after id;            在id后面增加一列叫id2
alter table t2 drop id2;                    删除id2这个列
alter table t2 change id ID bigint;         修改列名和数据类型
alter table t2 modify ID int;                   修改列的数据类型
show engines;                               查看数据库有哪些存储引擎
alter table t20 engine MyISAM;              修改表的存储引擎
show create table t20;                      查看修改存储引擎是否成功
alter table t20 default charset=utf8;       修改表的语言编码

RENAME TABLE:  重命名或者移动表
rename table db01.t1 to db02.t1;        移动表到另一个库里并重命名
alter table db01.t1 rename db02.t1
rename table tt1 to tt2;                    只重命名表名
alter table tt1 rename  tt2;

删除表:
drop table t1;
```

```
DDL(数据定义语言)：当执行时，会自动提交上面未提交的事务。create drop alter
```

3.1 常见的数据类型

3.1.1 数值类型

常见的数值类型

- **整数类型（精确值）**

数据类型-整数

- **定点类型（精确值）**
  - DECIMAL和NUMERIC

`DECIMAL`和`NUMERIC`类型的存储==精确的数值数据==。使用这些类型时，重要的是要保留==精确的精度==，例如使用**货币数据**。在MySQL中，`NUMERIC`被作为`DECIMAL`来应用，所以下面的举例`DECIMAL`同样适用于`NUMERIC`。

举例：

```
salary DECIMAL(5,2)
在上面的例子中，薪水字段的类型为decimal，表示精确的数字。其中，5代表精度，2代表刻度。
精度表示数值存储的有效位数，刻度表示小数点后面可存储的位数。
DECIMAL(5,2)能够存储五位数和两位小数的任何值，因此可以存储在salary 列中的值的范围-999.99是 999.99

特殊注意：
DECIMAL(M)和DECIMAL(M,0)是相等的，存储长度取决于M的值，默认情况下M值为10.刻度为0表示没有小数。
```

- **浮点类型（近似值）**
  - FLOAT和DOUBLE

`FLOAT`和`DOUBLE`类型代表近似数字数据值。MySQL对于单精度值使用四个字节，对于双精度值使用八个字节。`FLOAT`单精度浮点数精确到约7位小数，`DOUBLE`双精度浮点数精确到大约15位小数。`FLOAT`**类型会随着数值的增大精度会减小。**

举例：

```
FLOAT(M,D)，其中，M表示存储的有效位数，D代表小数点后面的位数；即整数位数+小数部分不能超过M值。
colum1 FLOAT(7,4)
上面例子中，如果你插入为999.00009到cloum1列，那么mysql在存储时会四舍五入变为999.0001插入。
```

总结：

数值类型	精度	存储空间	精确性
FLOAT	单精度	4字节	低
DOUBLE	双精度	8字节	低，比float高
DECIMAL	高精度	变长	高

3.1.2 字符串类型

常见的字符类型

- **CHAR**

CHAR类型的字符串为**定长**.长度范围是0到255之间的任何值.占用定长的存储空间,不足的部分用==空格==填充；读取时删掉后面的空格。

name char(10) harry-----

存储空间:

CHAR(==M==)**类型的存储空间和字符集有关系**，一个中文在utf8字符集中占用3个bytes、gbk占用2个bytes、数字和字符统一用一个字符表示。 **存储机制：**

在不够M长度时，MySQL在存储数据时，需要填充特殊的空格.

举例说明：

`name CHAR(M),M表示字符数`

- **VARCHAR**

VARCHAR是==变长存储==,仅使用必要的存储空间.

name varchar(10) harry

存储空间：

VARCHAR(M)类型的存储空间和字符集有关系，一个中文在utf8字符集中占用3个bytes、gbk统一占用2个bytes、数字和字符一个字符表示。

存储机制：

VARCHAR(M)字段存储实际是从**第二个字节开始存储**,然后用1到2个字节表示实际长度,剩下的才是可以存储数据的范围,因此最大可用存储范围是65535-3=65532字节;

第一个字节标识是否为空.（长度小于255字节,使用一个字节来表示长度;大于255字节使用两个字节来表示长度)。

举例说明：

- **其他**

BLOB：保存二进制的大型数据（字节串），没有字符集，eg：图片、音频视频等。

TEXT：保存非二进制字符串（字符串）；有一个字符集。

BINARY和VARBINARY：类似CHAR和VARCHAR；保存字节字符串，而不是字符字符串，这意味着它们没有字符集

- **存储需求**

在下面的表中，M表示非二进制字符串类型和二进制字符串类型的字节数的声明列长度。L表示给定字符串值的字节的实际长度。

string-type

3.1.3 日期类型

日期类型

- **DATE**

The `DATE` type is used for values with a date part but no time part. MySQL retrieves and displays `DATE` values in `'YYYY-MM-DD'` format. The supported range is `'1000-01-01'` to `'9999-12-31'`.

- **DATETIME**

The `DATETIME` type is used for values that contain both date and time parts. MySQL retrieves and displays `DATETIME` values in `'YYYY-MM-DD HH:MM:SS'` format. The supported range is `'1000-01-01 00:00:00'` to `'9999-12-31 23:59:59'`.

- **TIMESTAMP**

The `TIMESTAMP` data type is used for values that contain both date and time parts. `TIMESTAMP` has a range of `'1970-01-01 00:00:01'` UTC to `'2038-01-19 03:14:07'` UTC.

注意：

Invalid `DATE`, `DATETIME`, or `TIMESTAMP` values are converted to the “zero” value of the appropriate type (`'0000-00-00'` or `'0000-00-00 00:00:00'`).

无效的日期，日期时间等会被替换成‘0000-00-00’或‘0000-00-00 00:00:00’

- **TIME**

MySQL retrieves and displays `TIME` values in `'HH:MM:SS'` format (or `'HHH:MM:SS'` format for large hours values). `TIME` values may range from `'-838:59:59'` to `'838:59:59'`.

说明：小时部分可以是这么大，因为可以使用TIME类型不仅代表一个时间(必须小于24小时),而且可以表示运行时间或两个事件之间的时间间隔(可能大于24小时,甚至负数)。

==注意：==

TIME这一列如果存储缩写，需要注意mysql的解释方式。**无效的时间值会被转换成'00:00:00'** 。

`'11:12'` means `'11:12:00'`, not `'00:11:12'`.

`'12'` and `12` are interpreted as `'00:00:12'`.

- **YEAR**

`YEAR(4)` and `YEAR(2)` differ in display format, but have the same range of values.

For 4-digit format, MySQL displays `YEAR` values in `YYYY` format, with a range of `1901` to `2155`, or `0000`.

For 2-digit format, MySQL displays only the last two (least significant) digits; for example, `70` (1970 or 2070) or `69` (2069).

无效的值将会被转换成'0000'.

4. 用户权限管理

4.1 创建用户

- 通过使用用于创建帐户和建立其权限的帐户管理语句，例如 CREATE USER和GRANT。这些语句使服务器对基础授权表进行适当的修改。

```
创建用户的语法:
create user 'user'@'localhost';          5.7.17 不允许       创建用户不设置密码
create user user@host identified by 'password';      创建用户设置密码


'user01'@'%'                 user01从任意主机
'user01'@ip                  指定ip登录
'user01'@'10.1.1.0/24'     指定一个网络

说明:用户的信息保存在mysql数据库中的user表中,验证用户是否创建成功如下:
select user,host,password from mysql.user;

-h
-P
```

- 通过SQL语句直接操作MySQL授权表INSERT， UPDATE或 DELETE。 (==不建议==)

4.2 用户授权

- 权限分类

```
USAGE                                  无权限,只有登录数据库,只可以使用test或test_*数据库
ALL                                  所有权限
select/update/delete/super/slave/reload  指定的权限
with grant option                  允许把自己的权限授予其它用户或者从其他用户收回自己的权限
```

- 作用范围

```
*.*                    全库、全表    (mysql.user)
db01.*                 db01库下所有表(某库中的所有表)(mysql.db)
db01.t1                db01库中t1表(单表)(mysql.table_priv)
db01.t1.id             db01库中t1表的id列    (mysql.columns_priv)
```

- 权限保存位置

```
mysql.user             所有mysql用户的账号和密码,以及对用户对全库全表权限(*.*)
mysql.db                   非mysql库的授权都保存在此(db.*)
mysql.table_priv       某库某表的授权(db.table)
mysql.columns_priv  某库某表某列的授权(db.table.col1)
mysql.procs_priv       某库存储过程的授权
```

- 主机登录权限

```
user@localhost                          表示user只能在本地通过socket登录服务器
user@192.168.0.1                        表示user用户只能在192.168.0.1登录数据库服务器
user@192.168.0.0/255.555.255.0          指定某个子网的主机可以登录数据库
user@%                                  表示user用户能在所有的机器上登录数据库服务器
```

- 用户授权

```
show grants;    查看用户权限
grant 权限 on 库.表 to 用户@主机
grant 权限(列1,列2,...) on 库.表 to 用户@主机

mysql> grant select on db02.* to 'tom'@'10.1.1.6';
Query OK, 0 rows affected (0.01 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)


mysql> grant select,update(sal) on db02.t1 to 'jack'@'10.1.1.%';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

测试验证:
以tom用户登录:
mysql> select * from t1;
+------+-------+----------+
| id   | name  | sal      |
+------+-------+----------+
|    1 | harry | 12000.33 |
|    2 | tom   | 13000.57 |
+------+-------+----------+
2 rows in set (0.00 sec)

mysql> delete from t1;
ERROR 1142 (42000): DELETE command denied to user 'tom'@'db02.misshou.cc' for table 't1'
mysql> drop table t1;
ERROR 1142 (42000): DROP command denied to user 'tom'@'db02.misshou.cc' for table 't1'

以jack用户登录:
mysql> update t1 set name='aaa' where id=1;
ERROR 1143 (42000): UPDATE command denied to user 'jack'@'db02.misshou.cc' for column 'name' in
table 't1'
mysql> update t1 set sal=15000 where id=2;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from t1;
+------+-------+----------+
| id   | name  | sal      |
```

```
+------+-------+----------+
|    1 | harry | 12000.33 |
|    2 | tom   | 15000.00 |
+------+-------+----------+
2 rows in set (0.00 sec)


with grant option选项:
mysql> grant all on *.* to 'harry'@'%' identified by '123' with grant option;
Query OK, 0 rows affected (0.00 sec)

mysql> grant all on *.* to 'amy'@'%' identified by '123';
Query OK, 0 rows affected (0.00 sec)

测试harry用户和amy用户是否可以将自己的权限下放:
harry用户登录:
mysql> grant select on db02.t1 to 'haha'@'%' identified by '123';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

amy用户登录:
mysql> grant select on db02.t1 to 'haha'@'%' identified by '123';
ERROR 1142 (42000): GRANT command denied to user 'amy'@'db02.misshou.cc' for table 't1'


总结:
1. create user xxx  创建完后需要再次grant授权
2. grant xxxx           直接创建用户并授权

注意:从MySQL 5.7.6开始，不赞成使用grant修改密码。使用 ALTER USER来代替。
ALTER USER 'jeffrey'@'localhost' IDENTIFIED BY 'mypass';
ALTER USER USER() IDENTIFIED BY 'mypass';   如果当前连接是匿名用户可以用该语句修改密码
```

- 回收权限

```
revoke 权限 on 库.表 from 用户;
mysql> revoke update,select on mysql.user from stu6@localhost;  --撤消指定的权限
mysql> revoke all privileges,grant option from stu4@'%'; --撤消所有的权限
```

- 删除用户

```
mysql> drop user user01@'localhost';                          删除用户
mysql> select user from mysql.user where user='user01';     验证用户是否删除成功
mysql> drop user user;              默认删除该用户从任意主机登陆
mysql> rename user u01@'instructor.example.com' to u001@'localhost';  重命名用户名
mysql> show grants;                                             查看用户权限
mysql> show grants for user02@'%';                        查看指定用户的权限
mysql> drop user ''@'rhel6.example.com';                 删除一个匿名用户

mysql> delete from mysql.user where user='';              删除mysql中的匿名用户
```

```
mysql> delete from mysql.user where user='root' and host='::1';
mysql> flush privileges;

注意：如果tcp/ip登录，服务器端口不是默认3306，则需要加端口号
问题：可不可以给一个新的用户授权?

更改user02在所有机器上都能登录数据库.
mysql> update mysql.user set host='%' where user='user02';
mysql> flush privileges;
```

5. 补充总结

思考:

delete、truncate、drop有什么区别?

```
delete：删除数据
1、数据操作语言（DML）在事务控制里，DML语句是需要commit，不提交的话可以rollback；删除大量记录速度慢，只删除数据不回收高水位线
2、可以带条件删除

truncate：删除数据
1、数据定义语言（DDL）清大量数据速度快，高水位线（high water mark）下降
2、不能带条件truncate

drop：对象表、库、用户等；数据定义语言
```

六、查询语句

```
通配符:
    %       匹配0个或任意多个字符
    _       下划线，匹配一个字符
    =       精确匹配
    like    模糊匹配
    regexp 使用正则表达式来匹配

排序:
    order by    排序，默认升序
    asc         升序排列结果
    desc            降序排列结果

其他:
    group by    聚合
    having      筛选
    distinct        去除重复的行
    limit           限制

比较运算符:

> ,  < ,=  , != 或者< > ,>=   ,   <=
```

```
in ( values1,values2.... )
between v1 and v2    在v1至v2之间（包含v1,v2）

//取500-1000或者3000-5000的值
select * from emp where sal>=3000 and sal <= 5000 or sal >=500 and sal <=1000;

//不在3000与5000之间的值
select * from emp where sal not between 3000 and 5000

逻辑运算符:
not ( ! )  逻辑非
or ( || )    逻辑或
and ( && )  逻辑与
```

- 基本查询语句

```
select * from emp;      查看所有数据
select empno,ename,job,sal from emp;        查看指定列
select empno 工号,ename 姓名,job 工作,sal 工资 from emp;    给列别名增加可读性
或者
select empno as 工号,ename as 姓名,job as 工作,sal as 工资 from emp;
增加where条件查询:
mysql> select * from t1 where english >60 && english <90;
mysql> select * from t1 where name regexp '^l';
mysql> select * from t1 where name regexp '.*[0-9]' ;            --支持正则表达式
mysql> select * from t1 where name regexp '.*[0-9]+.*' ;        任意数字一次或一次以上
mysql> select * from t1 where name regexp '.*[0-9]{2}.*' ;
```

- 逻辑运算

```
逻辑运算: and | or| not
 select * from t1 where math >= 60 and english >= 60;
 select * from t1 where math >= 60 or english >= 60;      其中一科及格
 select * from t1 where math >= 60 and not  english >= 60;
```

- 排序

```
select * from emp order by deptno;      默认按照deptno列升序排列
select * from emp order by deptno asc;  指定按照deptno列升序排列
select * from emp order by deptno desc; 指定deptno列按降序排列
```

- 去重

```
select distinct deptno from emp;
```

- group by 分组和having

```
group by:根据给定数据列的每个成员对查询结果进行分组统计，最终得到一个分组汇总表。
注：一般情况下group需与统计函数（聚合函数）一起使用才有意义
max：求最大值
min：求最小值
sum：求总数和
avg：求平均值
count：求总行数

查找每个部门的最高工资：
select deptno,max(sal) from emp group by deptno;
select deptno,max(sal),count(*) from emp group by deptno;
select deptno,max(sal),count(*) from emp where deptno in (10,20) group by deptno;

select deptno,max(sal),count(*) from emp group by deptno having count(*)>5;

select empno,ename,sal+ifnull(comm,0) sum  from emp having sum >2000;

mysql> select ename,empno,deptno,sal from emp where sal in (select max(sal) from emp group by
deptno);
+-------+-------+--------+---------+
| ename | empno | deptno | sal     |
+-------+-------+--------+---------+
| BLAKE |  7698 |     30 | 2850.00 |
| SCOTT |  7788 |     20 | 3000.00 |
| KING  |  7839 |     10 | 5000.00 |
| FORD  |  7902 |     20 | 3000.00 |
+-------+-------+--------+---------+
4 rows in set (0.00 sec)


总结：
having 与where区别：
1、having与where类似，根据条件对数据进行过滤筛选
2、where针对表中的列发挥作用，查询数据
3、having对查询结果中的列发挥作用，筛选数据

查询20和30号部门基本工资加提成大于2000的人员信息
select deptno,empno,ename,sal+ifnull(comm,0) sum  from emp where deptno in(30,20)having sum
>2000;


IFNULL()函数：
MySQL中的IFNULL函数类似于Oracle中的NVL函数，其表达式为：IFNULL(expr1,expr2)，如果第一个参数不为空，则
返回第一个参数，否则返回第二个参数。
ifnull(comm,0)

IFNULL(expr1,expr2)
If expr1 is not NULL, IFNULL() returns expr1; otherwise it returns expr2.
```

```
IF()函数:
IF(expr1,expr2,expr3)类似于Oracle中的NVL2函数，如果第一个表达式的值为TRUE（不为0或null），则返回第二个参数的值，否则返回第三个参数的值

IF(expr1,expr2,expr3)
If expr1 is TRUE (expr1 <> 0 and expr1 <> NULL), IF() returns expr2. Otherwise, it returns expr3.

NULLIF()函数:
NULLIF(expr1,expr2),如果expr1=expr2为真，返回null；否则返回expr1

NULLIF(expr1,expr2)
Returns NULL if expr1 = expr2 is true, otherwise returns expr1. This is the same as CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END.
```

- 合并列

```
select concat(user,'    ',password) as 用户名和密码 from mysql.user;
create table ta3(path1 varchar(255),homedir varchar(255));
insert into ta3 set path1='/www/itcast',homedir='u01';
select concat(path1,'/', homedir) from t3;
select concat(path1,'/', homedir) as '家目录' from t3;
```

- 分页函数

```
limit 起始位置，偏移量  起始位置从0开始
select * from t1 limit n;           显示前n行
select * from t1 limit n,m;     显示n+1行至表中的n+m行

select * from t1 limit 10;          显示前10行
select * from t1 limit 10,10;       显示11至20行
select * from t1 limit 1;           显示第1行
select * from t1 limit 0,2;     显示第1,2行
select * from t1 limit 2,2;     显示3,4行
select * from t1 limit 4,2;     显示5,6行
select * from t1 limit 6,2;     显示第7,8行
```

- 多表查询

```
常规查询表:
select * from emp,dept where emp.deptno=dept.deptno;
select dept.deptno,dept.dname,emp.ename,emp.job from emp,dept where emp.deptno=dept.deptno;

表的连接:纵向连接/内连接/左连接/右连接
左连接（以左表为标准连接右表）:
left join=left [outer] join
select emp.ename,emp.job,dept.deptno from dept left join emp on emp.deptno=dept.deptno;


右连接（以右表为标准连接左表）:
```

```
right join= right outer join
select emp.ename,emp.job,dept.deptno from dept right join emp on emp.deptno=dept.deptno;

内连接:取多表之间的交集
select emp.ename,emp.job,dept.deptno from emp inner join dept on emp.deptno=dept.deptno;

纵向连接:合并【表结构相同】
select name,math,english,'' as 'chinese' from t2 union select name,math,english,chinese from t3;
```

- 四则运算

```
$(())   $[]   let  expr     shell:整数运算
awk 'BEGIN{print 1+1.5}'    AWK: 支持小数

mysql运算符: + - * /
select 1+2;
select 1-3;
select 1*4;
select 1/4;
select 1 + 1, (10-1)/3, 2*2/2;
select 1 + 1 from dual;     dual表,俗称万能表
乘幂需要调用函数: -3表示2的三次幂取倒数
mysql> select pow(2,3),power(2,3),pow(2,-3),power(2,-3);
+----------+------------+-----------+-------------+
| pow(2,3) | power(2,3) | pow(2,-3) | power(2,-3) |
+----------+------------+-----------+-------------+
|        8 |          8 |     0.125 |       0.125 |
+----------+------------+-----------+-------------+
1 row in set (0.00 sec)
```

课堂练习:

查询英语前三名

查询总分，以降序排列

查询总分第一名

查询英语最高分的姓名及分数

查询总成绩的平均分第一名: