数据库分类

关系型数据库：

- MySQL、Oracle、Sql Server、DB2、SQLite
- 通过表和表之间，行和列之间的关系进行数据的存储

非关系型数据库：

- Redis、MongDB
- 对象存储，通过对象的自身的属性来决定

安装MySQL

1. 解压
2. 配置环境变量 path MySQL bin目录
3. MySQL根目录新建配置文件`my.ini`
4. 编写配置文件

```
[mysqld]
basedir=D:\MySQL根目录\
datadir=D:\MySQL根目录\data\
port=3306
skip-grant-tables
```

5. 安装MySQL服务，MySQL bin目录，管理员CMD

```
mysqld -install
```

6. 初始化数据文件，MySQL bin目录，管理员CMD

```
mysqld -initialize-insecure --user=mysql
```

7. 启动MySQL服务

```
net start mysql
```

8. 登录(第一次登录密码为空即可)

```
mysql -u root -p
```

9. 修改密码

```
update mysql.user set
authentication_string=password('123456')
where user='root' and Host = 'localhost';
```

10. 退出

```
exit
```

11. 修改 my.ini文件，删除最后一句skip-grant-tables
12. 重启服务

```
net stop mysql
net start mysql
```

数据库字段类型

数值

类型		字节
tinyint	十分小的数据	1
smallint	较小的数据	2
mediumint	中等大小的数据	3
int	标准的整数	4
bigint	较大的数据	8
float	浮点数	4
double	浮点数	8
decimal	用于金融计算	

字符串

类型		长度
char	字符串固定长度	0~255
varchar	可变字符串	0~26635
tinytext	微型文本	2^8-1
text	文本串	2^16-1

日期

类型	
date	YYYY-MM-DD
time	HH:mm:ss
datetime	YYYY-MM-DD HH:mm:ss
timestamp	时间戳，1970.1.1到现在的毫秒数
year	年份

数据库字段属性

Unsigned(无符号的整数)：声明该列不能为负数
zerofill(0填充)：不足的位数，用0填充，int(3) 5--005
自增：默认自动在上一条记录值+1，必须是整数类型，通常用于主键
非空：not null，不赋值就会报错
默认值：不赋值会自动用默认值填充

MYISAM与INNODB

MYISAM是mysql5.5之前的默认引擎，INNODB是之后版本的默认引擎

	MYISAM	INNODB
事务	不支持	支持
数据行锁定	不支持	支持
外键约束	不支持	支持
全文索引	支持	不支持
表空间大小	较小	较大，为前者的两倍

各自优势

- MYISAM：节约空间，速度快
- INNODB：安全性高，支持事务，多表多用户操作

存储区别：
所有的数据库文件都在data目录下，一个文件夹对应一个数据库

- INNODB：在本地存储文件夹中只有一个*.frm文件，以及上级目录中的ibdata1文件
- MYISAM：在本地存储文件夹下有三个文件
  - *.frm 表结构的定义文件
  - *.MYD 数据文件
  - *.MYI 索引文件

基本命令行操作

连接数据库

```
mysql -uroot -p
```

修改用户密码(连接之后)

```
update mysql.user set
authentication_string=password('123456')
where user='root' and Host='localhost'
```

刷新用户权限

```
flush privileges
```

查看所有数据库

```
show databases
```

切换数据库

```
use student
```

查看数据库中所有的表

```
show tables
```

查看数据库中某一张表的结构

```
describe student
desc student
```

创建数据库

```
create database demo
```

退出连接

```
exit
```

数据库四大语言

DDL定义语言

> 操作库

创建数据库

```
create database [if not exists] student
```

删除数据库

```
drop database [if exists] student
```

使用数据库

```
-- 如果表名或者字段名是一个关键字，需要用``
use `u`
```

查看数据库

```
-- 查看所有数据库
show databases
```

查看数据库创建语句

```
show create database student
```

查看表的创建语句

```
show create table student
```

查看表结构

```
describe student
desc student
```

查看表的索引信息

```
show index from student
```

> 操作表

创建表

```
-- 表名字段尽量使用``包裹
-- primary key 主键 auto_increment 自增 comment字段注释
-- 字符串使用单引号包裹
-- engine=innodb 表类型 charset=utf8 字符集设置，默认为Latin1，不支持中文
create table if not exists `student` (
    `id` int(4) not null auto_increment comment '学号',
    `name` varchar(30) not null default '匿名' comment '姓名',
    `pwd` varchar(20) not null default '123456' comment '密码',
```

```
    `gender` varchar(2) not null default '女' comment '性别',
    `birthday` datetime  default null comment '出生日期',
    `addres` varchar(100)  default null comment '家庭住址',
    `email` varchar(50)  default null comment '邮箱',
    primary key(id)
)engine=innodb default charset=utf8
```

修改表名

```
alter table student rename as student1
```

添加表字段

```
alter table student add age int
```

修改表字段

```
-- modify仅仅只能修改字段约束
alter table student modify age varchar(11)

-- change不仅可以修改字段约束，还可以给字段重命名
alter table student change age age1 int(11)
```

删除表字段

```
alter table student drop age1
```

删除表

```
-- 如果存在就删除
drop table [if exists] student
```

添加外键：物理外键，数据库级别的不建议使用，因为多表关联了，影响表操作，现在一般外键都是通过代码实现逻辑外键，数据库表值存储数据，不关联

```
alter table student add constraint
`FK_gradeid` foreign key(`gradeid`)
references `grade`(`gradeid`);
```

DML操作语言

插入数据

```sql
-- 不写字段名，必须保证数据顺序与字段顺序一一对应，自增值直接用null代替即可
insert into student values (null,'张三','123456','男','2020-09-10 10:00:00','四川','2036786419@qq.com')

-- 也可以根据字段名选择性插入
insert into student (`id`,`name`,`addres`) values ('8','李四','北京')

-- 批量插入
insert into student values
(null,'张三','123456','男','2020-09-10 10:00:00','四川','2036786419@qq.com'),
(null,'李四','123456','男','2020-09-10 10:00:00','四川','2036786419@qq.com'),
(null,'王五','123456','男','2020-09-10 10:00:00','四川','2036786419@qq.com')
```

修改数据

```sql
update student set name='赵六' where name='张三'
-- 一次修改多个属性值，用逗号分隔
update student set name='赵六',age=18 where id = 1
```

删除数据

```sql
-- 删除表中某一行记录
delete from 表名 [where 条件]

-- 删除表中所有记录,不跟条件，将会逐条删除表数据
delete from 表名

-- 删除整表，复制出有相同结构的表
truncate table user

-- truncate重新设置自增列，计数器会归零。不会影响事务
-- 关于自增计数，重启数据库后，INNODB引擎计数器清零(存储在内存中，断电即失)，MYISAM引擎计数器不会清零(存储在文件中)
```

DQL 查询语言

测试数据

```sql
create database if not exists `school`;
-- 创建一个school数据库
use `school`;-- 创建学生表
drop table if exists `student`;
create table `student`(
    `studentno` int(4) not null comment '学号',
    `loginpwd` varchar(20) default null,
    `studentname` varchar(20) default null comment '学生姓名',
```

```
    `sex` tinyint(1) default null comment '性别，0或1',
    `gradeid` int(11) default null comment '年级编号',
    `phone` varchar(50) not null comment '联系电话，允许为空',
    `address` varchar(255) not null comment '地址，允许为空',
    `borndate` datetime default null comment '出生时间',
    `email` varchar (50) not null comment '邮箱账号允许为空',
    `identitycard` varchar(18) default null comment '身份证号',
    primary key (`studentno`),
    unique key `identitycard`(`identitycard`),
    key `email` (`email`)
)engine=myisam default charset=utf8;

-- 创建年级表
drop table if exists `grade`;
create table `grade`(
    `gradeid` int(11) not null auto_increment comment '年级编号',
  `gradename` varchar(50) not null comment '年级名称',
    primary key (`gradeid`)
) engine=innodb auto_increment = 6 default charset = utf8;
```

```
28
29  -- 创建科目表
30  drop table if exists `subject`;
31  create table `subject`(
32      `subjectno`int(11) not null
    auto_increment comment '课程编号',
33      `subjectname` varchar(50) default
    null comment '课程名称',
34      `classhour` int(4) default null
    comment '学时',
35      `gradeid` int(4) default null
    comment '年级编号',
36      primary key (`subjectno`)
37  )engine = innodb auto_increment = 19
    default charset = utf8;
38
39  -- 创建成绩表
40  drop table if exists `result`;
41  create table `result`(
42      `studentno` int(4) not null comment
    '学号',
43      `subjectno` int(4) not null comment
    '课程编号',
44      `examdate` datetime not null comment
    '考试日期',
45      `studentresult` int (4) not null
    comment '考试成绩',
46      key `subjectno` (`subjectno`)
47  )engine = innodb default charset = utf8;
48
```

```sql
-- 插入学生数据 其余自行添加 这里只添加了2行
insert into `student`
(`studentno`,`loginpwd`,`studentname`,`sex`,`gradeid`,`phone`,`address`,`borndate`,`email`,`identitycard`)
values
(1000,'123456','张伟',0,2,'13800001234','北京朝阳','1980-1-1','text123@qq.com','123456198001011234'),
(1001,'123456','赵强',1,3,'13800002222','广东深圳','1990-1-1','text111@qq.com','123456199001011233');

-- 插入成绩数据  这里仅插入了一组，其余自行添加
insert into `result`(`studentno`,`subjectno`,`examdate`,`studentresult`)
values
(1000,1,'2013-11-11 16:00:00',85),
(1000,2,'2013-11-12 16:00:00',70),
(1000,3,'2013-11-11 09:00:00',68),
(1000,4,'2013-11-13 16:00:00',98),
(1000,5,'2013-11-14 16:00:00',58);

-- 插入年级数据
```

```
insert into `grade`
(`gradeid`,`gradename`) values(1,'大一'),
(2,'大二'),(3,'大三'),(4,'大四'),(5,'预科
班');

-- 插入科目数据
insert into
`subject`(`subjectno`,`subjectname`,`cla
sshour`,`gradeid`)values
(1,'高等数学-1',110,1),
(2,'高等数学-2',110,2),
(3,'高等数学-3',100,3),
(4,'高等数学-4',130,4),
(5,'C语言-1',110,1),
(6,'C语言-2',110,2),
(7,'C语言-3',100,3),
(8,'C语言-4',130,4),
(9,'Java程序设计-1',110,1),
(10,'Java程序设计-2',110,2),
(11,'Java程序设计-3',100,3),
(12,'Java程序设计-4',130,4),
(13,'数据库结构-1',110,1),
(14,'数据库结构-2',110,2),
(15,'数据库结构-3',100,3),
(16,'数据库结构-4',130,4),
(17,'C#基础',130,1);
```

普通查询

```
-- 查询全部
select * from student
```

```sql

-- 查询指定字段
select `StudentNo`,`StudnetName` from student

-- 给查询列取别名显示
select `StudentNo` 学号,`StudentName` 学生姓名 from student s

-- Concat函数简单修改查询出的结果
select Concat('姓名: ',StudentName) '新名字' from student

-- 对查询出的结果去掉重复值显示
select distinct `studentNo` from result
```

数据库中的表达式

```sql
-- 查询mysql版本
select version()

-- 计算
select 100*3-1 '计算结果'

-- 查询自增步长
select @@auto_increment_increment

-- 对查询结果进行计算后显示
select `StudentNo`,`StudentResult`+1 '提分后' from result
```

模糊查询

```sql
-- 占位符 _代表一个字符 %代表任意个数字符
select `StudentNo`,`StudentName` from `student`
where StudentName like '张%'

select `StudentNo`,`StudentName` from `student`
where StudentNo in (1001,1002,1003)

select `StudentNo`,`StudentName` from `student`
where StudentNo is not null

select `StudentNo`,`StudentName` from `student`
where StudentNo is null
```

连接查询

- 交叉查询 cross join

```
查询到两个表的笛卡尔积

select * from 表1 cross join 表2

selext * from 表1,表2
```

- 内连接 inner join(inner可省略)

```
显式内连接
select * from 表1 inner join 表2 on 关联条件

隐式内连接
select * from 表1,表2 where 关联条件
```

- 外连接 outer join(outer可省略)

```
左外连接
select * from 表1 left outer join 表2 on 关联条件

右外连接
select * from 表1 right outer join 表2 on 关联条件
```

- 子查询

```
带in的子查询
select * from 表1 where 字段 in (子查询语句)

带exists的子查询
select * from 表1 where exists (子查询语句)  //子查询语句成立，显示查询结果

带any的子查询
select * from 表1 where 字段 > any (子查询语句)

带all的子查询
select * from 表1 where 字段 > all (子查询语句)
```

- 合并查询结果

```
union 合并后去重
select column_name(s) from table_name1
union
select column_name(s) from table_name2

union all 只合并，不去重
select column_name(s) from table_name1
union all
select column_name(s) from table_name2
```

排序

```
-- 根据哪个字段升序/降序排列，写于where条件之后
order by 字段名 asc/desc
```

分页

```
select * from `subject`
limit 0,2

-- 第一页 limit 0,5         (1-1)*5
-- 第二页 limit 5,5         (2-1)*5
-- 第三页 limit 10,5        (3-1)*5
-- 第N页 limit M,5          M = (N-1)*pageSize

-- 【N】当前页
-- 【pageSize】页面大小
-- 【M】起始值下标
-- 【总页数 = 数据总数/页面大小】
```

分组与过滤

```
-- group by 分组字段 having 过滤条件
select studentNo,studentClass from student group by studentClass having studentClass = '软件19-1班'
```

数据库级别的md5加密

```sql
-- 加密函数
md5()

-- 在插入时完成加密
insert into student
(`name`,`pwd`,`addres`) values('张三',md5('123456'),'北京')
```

DCL 控制语言

函数

常见函数

```sql
-- 数学函数
select abs(-8) -- 绝对值
select ceiling(9.4) -- 向上取整
select floor(9.4) -- 向下取整
select rand() -- 0~1之间的随机数
select sign(-1) -- 返回参数符号，0返回0，负数返回-1，正数返回1

-- 字符串
select char_length('代码书写人生') -- 返回字符串长度
select concat('Hello',' ','World') -- 拼接字符串
select lower('JDKASDAbhfjas') -- 转小写
select upper('JDKASDAbhfjas') -- 转大写
```

```sql
select instr('HelloHHH','H') -- 返回第一次出现子串的位置
select replace('失败是成功之母','之母','他妈') -- 替换指定字符串
select substr('sdnaldnaldsd',4,5) -- 从第4个位置，返回长度为5的字符串
select reverse('赵兄托我帮你办点事') -- 字符串反转

-- 时间和日期
select current_date() -- 获取当前日期
select current_date -- 获取当前日期
select now() -- 获取当前时间
select localtime() -- 本地时间
select sysdate() -- 系统时间

select year(now()) -- 年
select month(now()) -- 月
select day(now()) -- 日
select hour(now()) -- 时
select minute(now()) -- 分
select second(now()) -- 秒

-- 系统
select system_user()
select user()
select version()
```

聚合函数(常用)

函数名	作用
count()	统计记录数
sum()	求和
avg()	平均值
max()	最大值
min()	最小值

- count(*)与count(1)作用差不多
- count(列名)忽略NULL值

事务

什么是事务

要么都成功，要么都失败，将一组SQL放在一个批次中执行

ACID

原子性：要么都成功，要么都失败

一致性：保持业务数据前后一致

隔离性：多事务执行，相互之间隔离

持久性：事务一旦提交，不可逆，会持久到数据库中

隔离所导致的问题

脏读：事务A读取到了事务B未提交的数据

不可重复读：事务A读取到了事务B修改后的数据(一个事务中多次读取产生不同结果，这不一定是错误，也许是场合不对)

幻读(虚读)：事务A读取到了事务B提交的数据

事务操作

1. mysql默认开启事务自动提交，先手动关闭提交

```
-- 关闭
set autocommit = 0

-- 开启
set autocommit = 1
```

2. 开启事务

```
start transaction
```

3. 事务提交

```
commit
```

4. 事务回滚

```
rollback
```

索引

> 概念：MySQL官方对索引的定义为：索引（Index）是帮助MySQL高效获取数据的数据结构。
>
> 提取句子主干，就可以得到索引的本质：**索引是数据结构。**

索引的分类

主键索引(primary key)：唯一的标识，主键不可重复，只能有一个列作为主键

唯一索引(unique key)：避免列中重复的行出现，唯一索引可以重复，多个列都可以标识唯一索引

常规索引(index/key)：默认的，index，key关键字来设置

全文索引(fulltext)：在特定的数据库引擎下才有，MyISAM，快速定位数据

主键索引只有一个，唯一索引可以存在多个

```
-- 索引的使用
-- 1. 在创建表的时候给字段增加索引
-- 2. 创建完毕后，修改表增加索引
-- 3. 创建索引 id_表名_字段
```

```
create index id_student_name on student_name(`name`)

-- 查看表所有的索引信息
show index from student

-- 增加一个全文索引〔索引名〕列名
alter table school.student add fulltext index `studentname` ('studentname');

-- explain分析sql执行的状况
explain select * from student; --非全文索引

explain select * from student where match(studentname) against('刘');
```

索引使用原则

- 索引不是越多越好
- 数据量少的情况先不考虑索引
- 不要给经常变动的数据加索引
- 给经常查询的字段添加索引

MySQL索引背后的数据结构及算法原理

权限管理和备份

```
-- 创建用户 create user 用户名 identified by '密码'
```

```
create user JohnCena identified by '123456'

-- 修改当前用户密码
set password = password('123456')

-- 修改指定用户密码
set password for xxx = password('123456')

-- 重命名 rename user 用户名 to 新用户名
rename user JohnCena to demo

-- 给用户授权
grant all privileges on *.* to 用户名

-- 查询权限
show grants for 用户名

-- 撤销用户权限
revoke all privileges on *.* from 用户名

-- 删除用户
drop user 用户名
```

三大范式

第一范式(1NF)：是指关系中的所有属性都是不可再分的数据项，同一列中不能有多个值（消除属性多值）

第二范式(2NF)：数据库表满足第一范式，并且每一列都依赖主键，联合主键时每一列都完全依赖于主键（消除部分依赖）

第三范式(3NF)：如果一个关系满足2NF，并且除了主键以外的其他列都不传递依赖于主键列，则满足第三范式（消除传递依赖）