目录

1.数据库概念

- 数据库：本质是一个文件，图书馆保存书籍，数据库保存数据。
- 数据库简介

  `SQL Server Oracle` 微软

  `MySQL`
- 使用DOS界面操作数据库(`cmd`)
  - DOS界面报错： `不是内部或外部命令`
    - 解决方法：配置环境变量
      1. 登录 `localhost` ,输入密码，查看是哪个版本的数据库
      2. 进入数据库 `bin` 文件夹,拷贝当前路径
      3. 右击我的电脑-属性-高级系统设置-环境变量
      4. 打开 `Path` 变量(没有的话新建)，新建，把 `bin` 路径放进去

         【注】如果是一整串的话，在后边拼接%后插入路径
  - 连接数据库
    - 链接 `phpnow` 的 `5.1` 数据库 (在 `cmd` 界面操作)
      - `d:` //进入D盘
      - `cd D:\PHPnow\MySQL-5.1.50\bin`
      - `mysql -u root -p` ： 输入密码就进入5.1版本的数据库了(`123456`)
  - 两个数据库端口一样
    - 找到数据库文件 `my.ini` 修改 `port` 端口

2.MySQL 命令操作 (phpnow本机上的mysql服务为 MySQL5_pn)

- 登录 `mysql -h localhost -u root -p`
  - 说明： `mysql` 是一个程序， `-h` 主机 `-u` 用户 `-p` 密码
  - `ctrl + c` 终止当前程序

- 数据库命令 (接下来输入的 `mysql` 命令操作都要加分号)
  - 查看数据库： `show databases;`
  - 创建数据库： `create database 数据库名;`
  - 创建一个 `utf-8` 字符集的db2数据库：

```
create database db2 character set utf8;
```

- 创建一个使用`utf-8`字符集,并带校对规则(`utf8_general_ci`)的mydb3数据库
  - 校对规则：可以理解成排序规则，默认`utf8_general_ci`，`ci`代表大小写不敏感

```
create databases mydb3 character set outfit collate utf8_general_ci;
```

  - 校对规则了解即可，默认不区分大小写
- 删除数据库：

```
drop database [if exists] db_name;
```

  - `if exists`可选的。如果存在的话就删除，不存在就不删除。
- 选择数据库：`use 数据库名字`;
- 创建表（基本语句）`[]`内为可选项 （一定要设置ID）

```
create table 表名(
  字段1 数据类型 [列级约束条件],
  字段2 数据类型 [列级约束条件],
  字段n 数据类型 [列级约束条件],
  [表级约束条件]
)character set 字符集 collate 校对规则;
```

  - 表可以设置字符集，遵从以表为准。
  - 表的字符集 和 校对规则 默认继承数据库

- mysql 数据类型 （一字节 = 8bit）
  - 整型

数据类型	取值范围	大小
`tinyint`	`-128` 到 `127`	1字节
`smallint`	`-2^15` 到 `2^15 - 1`	2字节
`int`	`-2^31` 到 `2^31 - 1`	4字节
`bigint`	`-2^63` 到 `2^63 - 1`	8字节

  - 浮点型
    - `Float(M,D)`： 单精度浮点数
    - `double(M,D)` ： 双精度浮点数
      - `M`：精度，数据的总长度
      - `D`：标度，小数点后的长度
    - `double` 和 `float` 的区别是`double`精度高，有效数字`16`位，`float`精度`7`位。但`double`消耗内存是`float`的两倍，`double`的运算速度比`float`慢得多
  - 字符串 (一个中文三个字符)
    - `char(10)` ： 最多只能有十个字符

        - `varchar(30)`： 超过了也可以多一点

    - 日期
        - `date`： 日期 "2000-01-01"

    - 文本
        - `text`： 文本 (理论上无上限)
    - 数据单位
        - 比特位 : `Bit`
        - 字节(`byte`) : `1b = 8Bit`
        - 千字节(`kb`) : `1kb = 1024b`

3.案例 (创建一个表)

- 确定设置的元组及元组数据类型

元组名称	数据类型
`id`	整型
`name`	字符型
`sex`	字符型或 `bit` 型
`brithday`	日期型 生日
`job`	字符型
`salary`	小数型 工资
`resume`	大文本型 简历

1. 当我们在 `int` 整型后面增加了 `unsigned` 后,就是一个**无符号数**。
2. 我们创建 `char` 或者 `varchar` 类型字段，默认值设置成 `""` 。

```
name char(3) not null default ""
```

3. 如果我们使用小数，可以使用 `float` 。大文本使用 `text`

```
create table employee(
  id int unsigned unique,
  name char(30) not null default "",
  sex char(3) not null default "",
  birthday date,
  job varchar(30) not null default "",
  salary float not null default 0.0,
  resume text
)character set utf8;
```

- 列级约束条件

约束关键字	描述

约束关键字	描述
unsigned	无符号，既为非负数，用此类型可以增加数据长度 `tinyint` 为 `-128~127` ，设置了 `unsigned` 范围就变成 `0~255`
not null	不许为空
default 值;	默认值
unique	唯一性约束（每条记录的指定字段的值不能重复）
primary key	主键约束 (列不能为空,且不能重复)
auto_increment	设置指定字段的值自动增加
check(条件表达式)	用于检验输入值，拒绝接受不满足条件的值 id int check(id=1 or id=2)
enum(1,2)	枚举，和 `check` 相似,设置的值需要匹配enum给定值 id int enum(1, 2)

- 表级约束条件

约束关键字	描述
primary key	主键约束 (列不能为空,且不能重复) `constraint <约束名> primary key [CLUSTERED](字段名1,字段名2,…)`
foreign key	外键约束，用于设置参照完整性规则 `constraint <约束名> forreign key <外键> references <被参照表(主键)>`
unique	既可用于列级完整性约束，也可用于表级完整性约束 `constraint <约束名> unique(<字段名>)`

- 扩展命令：

功能	代码
查看数据库	`show create database 数据库名称;`
查看表	`show create table 表名;`
查看当前数据库中所有表	`show tables;` (要先进入数据库)
删除数据库	`drop database [if exists] 数据库名称;` `if exists` 存在删除，不存在也不会报错
删除表 (删除结构)	`drop table 表名1,[表名2];` （删除表结构和数据）
复制表 (复制结构)	`create table 新表名 select from * 旧表名;`
查看表结构	`desc 表名;`
修改数据库字符集	`alter database 数据库名 character set utf8;`

功能	代码
修改表字符集	`alert table 表名 default character set utf8 collate utf8_general_ci;`
修改表名	`alter table 表名 rename [TO] 新表名;`
修改字段名	`alter table 表名 change 旧字段名 新字段名 新数据类型 列级约束;`
修改字段数据类型	`alter table 表名 modify 字段名 新数据类型 列级约束;`

- 添加字段:

```
alter table 表名 ADD 新字段名 数据类型 [列级约束条件] [first|after 已存在字段名];

alter table student ADD class char(10) unique AFTER ssex ;
// 在student表 ssex字段后添加 class 字段
```

- 删除字段:

```
alert table 表名 drop 字段名;
```

- 注意区分 JS中的`alert` 和 sql`alter`

4.数据库的crud语句 (insert、update、delete、select)

- `insert`语句： 插入数据

```
insert into 表名(字段) values(值)
```

- `update`语句： 更新数据

```
update 表名 set 字段 = 值 where 条件;
```

- `delete`语句： 删除数据

```
delete from 表名 [where];
```

  - `delete`删除表数据，`drop`会删除 表数据和表结构 (表结构理解为表)
- `select`语句： 查找数据(重要)

```
select * from 表名
```

  - 查询多个字段并相加

```
select c1 + c2 from table where Id = 1
```

    - 避免其中一个字段为`null`,结果就为`null`:

```
select IFNULL(c1,0) + IFNULL(c2,0) from table where Id = 1
```

- `insert`语句，添加数据

- 对指定字段名添加数据

```
insert into 表名(字段名1, 字段名2, ……)  value(值1, 值2, ……);
```

- 不指定字段名，添加全部字段 (值要一一对应)

```
insert into 表名 values(值1, 值2, ......)
```

- 添加多条记录（指定字段名和不指定都可以）

```
insert into 表名[(字段名1, 字段名2, ……)] values(值1,值2, ……), (值1,值2, ……), ...;
```

- 注意事项
  1. 插入的数据应与字段的数据类型相同
  2. 数据的大小应在列的规定范围内
  3. 在 `values` 中列出的数据位置,必须与 字段 的位置相对应
  4. 字符 和 日期型 数据应包含在单引号(`''`)中。

- `select` 语句，查询 (单表/多表) `[]` 可选项 `<>` 必填项
  - 单表无条件查询

```
select [all/distinct] <选项>[as 显示列名] [,<选项2>[as 显示列名],...]
from <表名/视图名> [where条件表达式 and where条件2] [limit [m,]n ];
```

    - `all` : 表示输出所有记录，包括重复记录。默认值为 `all` 。
    - `distinct` ： 表示在查询结果中去除掉重复的值。
    - `limit [m,]n` :
      - `limit n` : 返回查询结果中的前N行
      - `limit[m,]n` : 加 `[m]` , 返回 `m-n` 行, 包括 `m` 和 `n`
        - `M` 默认从 `0` 开始， `N` 的取值范围由表中的记录数决定。
    - 选项：查询结果中的**输出列**。可为**字段名、表达式或函数**。用 `*` 表示表中的所有字段。若选项为表达式或函数，输出的列名系统自动给出，不是原字段名，故用 `AS` 重命名。
    - 显示列名：在输出结果中，设置选项显示的列名。用引号定界或不定界。
    - 表名：要查询的表。表不需要打开，到当前路径下寻找所对应的文件。
    - 查询全体学生的详细信息： `select * from 表名;`
    - 输出学生表中的前十条记录

```
select * from 表名 limit 10;
select * from 表名 limit 0,10; // 是等价关系
```

- `update` 语句, 修改数据

```
update 表名 set 字段名1=值1，[字段名2=值2，...]  where条件表达式;
// 将Tom原来的工资增加1000;
update employee set 工资 = 工资+1000 where name = 'Tom';
```

- `delete` 语句, 删除数据

```
delete from 表名 [WHERE 条件表达式]; // 删除元组数据
```

- delete注意事项
  1. 在 `DELETE` 语句中如果没有使用 `WHERE` 子句，则会将表中的所有记录都删除（删除表里的所有记录，表还存在）。
  2. `delete` 只能删除元组/表, 不能删除列的数据，
     - 可以使用 `alter table 表名 drop 字段名;` 删除这个字段
     - 或使用 `update` 将列数据换成 `null`
- 删除表中所有的记录

```
truncate [table] 表名;
```

- `truncate` 语句和 `delete` 语句区别
  - `truncate` 语句和 `delete` 语句都能实现删除表中的所有数据 的功能，但两者有一定的区别。
    1. `delete` 语句是 `DML` 语句， `truncate` 语句通常被认为是 `DDL` 语句。
    2. `delete` 语句后面可以跟 `where` 子句，通过指定 `where` 子句中的条件表达式只删除满足条件的部分记录，而 `truncate` 语句只能用于删除表中的所有记录。
    3. 使用 `truncate` 语句删除表中的数据，
       - 在向表中添加记录时，自动增加字段的默认初始值重新由1开始，
       - 使用 `delete` 语句删除表中所有记录，
       - 再向表中添加记录时，自动增加 字段的值为删除时该字段的最大值加1。
    4. 使用 `delete` 语句时，每删除一条记录都会在 日志中记录，
       - 而使用 `truncate` 语句时，不会 在日志中记录删除的内容，
       - 因此 `truncate` 语句的执行效率比 `delete` 语句高。

5.where子句常用运算符

- 比较运算符： `=、<、>、<=、>=、<>(不等于)、!=(不等于)、!<、!>`
- 范围运算符： `between and` 、 `not between and`

```
where 成绩 [not] between 最小值 and 最大值; // 包括最小值和最大值
```

- 列表运算符： `in` 、 `not in`

```
where 系别 [not] in('软件工程系','计算机','信息工程系'); // 可以理解为集合
```

- 字符匹配符： `like` 、 `not like`

```
where 字段名 [not] like '字符串' escape '转义字符'; // 参考下文‘通配符’
```

- 空值： `is null` 、 `is not null`

```
where 成绩 is [not] null;  // 注：这里的 is 不能用=替换
```

- 逻辑运算符： `and` 、 `or` 、 `not` (与 或 非)

```
select s1 from t1 where s1 <> any (select s1 from t2);
```

  - `any` 任意一个，要跟比较运算符 ， `any all some` 要用在嵌套查询
  - `all` 全部，要跟比较运算符
  - `some` 任何一个，和ANY是近义词

- 通配符：
  - `%`：任意多个字符
    1. `m%` 查询以`m`开头的任意字符串
    2. `%m` 查询以`m`结尾的任意字符串
    3. `%m%` 查询在任何位置包含`m`的所有字符串
  - `_`：单个字符
    1. `m_` 查询以`m`开头，任意一个字符结尾的字符串
    2. `_m` 查询任意一个字符开头，以`m`结尾的字符串
- 正则表达式：`regexp`

```
where 字段名 regexp '操作符'
```

  - `^b` ：匹配以`b`开头的字符串
  - `st$` ：匹配以`st`结尾的字符串
  - `b.t` ：匹配`b`和`t`之间有一个字符的字符串
  - `*n` ：匹配`n`前面有任意个字符串的字符
    - ..... 具体参考书本132页。
- mysql 要用`\\转义字符` 例`\\.` = `.`

6.聚集函数

- 聚集函数：可做为**列标识符**出现在`select`子句中的目标列、`having`子句的条件中或`order by`子句中(上述三个都是已经查询好的结果，聚集函数可以作用在结果二次处理)

```
select  id,count(聚集函数)  from 表名 [where条件表达式];
```

函数名	写法	说明
count	`count( [distinct/all]* )`	统计元组个数，包括有NULL值的行
count	`count([distinct/all] <列名>)`	对指定列的行进行计数，忽略NULL行
sum	`sum([distinct/all] <列名>)`	计算一列值的总和（此列必须为整型）； `sum()` 用来返回指定列值的和 null略过 字符串的话直接为0
avg	`avg([distinct/all] <列名>)`	计算一列的平均值（此列必须为整型）； 求一个列的平均值 null会被略过
max	`max([distinct/all] <列名>)`	求一列之中的最大值; 返回指定列的最大值 如果是中文的话 则是字符串长度
min	`min([distinct/all] <列名>)`	求一列之中的最小值

  - 会忽略`null`值的聚集函数, 并且只能填入一个列
    - `count(列名)` ：注：`count(*)`统计元组个数，包括有`NULL`值的行
    - `avg()` `sum()` `min()` `max()`

7.order by 排序查询结果

- `order by`子句排序，升序（`ASC`）降序(`DESC`)

```
select * from table [order by <列名1> [ASC|DESC]][,<列名2> [ASC|DESC]]
```

  - 排序规则：
    1. 对于空值，如按升序排列，含空值的元组将最先显示;如按降序排列，空值的元组将最后显示。
    2. 中英文字符按其`ASCII`码大小进行比较。
    3. 数值型数据根据其数值大小进行比较。
    4. 日期型数据按年、月、日的数值大小进行比较。
    5. 逻辑型数据`false`小于`true`。
    - mysql没有布尔值，使用`tinyint(1)`，`0`代表`false`。

8.grop by 对查询结果进行分组

- `group by`子句分组 （对查询结果进行分组）

```
select * from table [group by 列名清单 [having条件表达式]] [order by ...]
```

  - 注意点：`where`**后不能使用聚集函数**，因为`where`是查询符合条件的数据，聚集函数是对结果进行筛选
  - `where`和`having`的区别
    - `WHERE`条件与`HAVING`条件的区别在于作用对象不同。
      - `HAVING`条件作用于结果组，选择满足条件的结果组;
      - 而`WHERE`条件作用于被查询的表，从中选择满足条件的记录。

9.select查询扩展(多表查询/嵌套查询)

- 多表查询

```
select 表名.列名,表名2.列名2 from 表名1 别名1, 表名2 别名2 where 条件表达式;
```

- 嵌套查询

```
select 列名 from 表 where 列 in(
  select ....
);

select student.id, math.sum
from student a,math b
where a.id = b.id AND a.name IN(
    select name from sss;
);
```

10.数据库可视化界面(phpNow)

- 登录phpAdmin：http://localhost/phpMyadmin/ (root 123456)
- 创建数据库
  - 点击数据库 -> 下方新建数据库 -> 数据库编码选择`utf8_general_ci`
- `auto_increment`：自增，如果不填写数据，会自动增加

11.通过PHP操作数据库(建立在localhost目录)

- 链接数据库(天龙八部)

1. 链接数据库

```
mysql_connect('ip','use','password');
```

   - `ip`：链接数据库的IP/域名，可以跟端口号 `'localhost:3306'`
   - `use`：用户名(root)
   - `password`：密码

```
$link = mysql_connect('localhost','root','123456');
```

   - 返回值：链接成功返回`true`，链接失败返回`false`；

2. 判断是否连接成功

```
if(!$link){
    echo '链接失败';
    exit;  //终止后续所有代码
}
```

   - `exit`和`return`不同，`return`会终止当前作用域，`exit`终止当前脚本(文件)。

3. 设置字符集

```
mysql_set_charset("utf8");
```

4. 选择数据库

```
mysql_select_db("yyy");
```

5. 准备sql语句

```
$sql = "SELECT * FROM student where ...";
```

6. 发送sql语句 (返回的是mysql结果,不能直接输出)

```
$res = mysql_query($sql);
// var_dump($res); 不能直接输出
```

7. 处理结果

```
mysql_fetch_assoc(要处理的数据);
```

   - 每调用一次就依次返回一行mysql数据。
   - 利用循环，输出所有的查询结果

```
while( $newres = mysql_fetch_assoc($res) ){
  echo $newres;
  echo "<br/>";
}
```

8. 关闭数据库

```
mysql_close($link);
```

- PHP中准备`$sql`语句的注意点
  1. 如果数据类型是字符串，要用`''`包裹起来`'{$username}'`,`'{$password}'`,`{$id}`
  2. 如果数据类型是整型或者浮点,不要用`''`

12.获取学员成绩（前后端分离）(bootStrap了解)

- 数组从前端到后台，从后台到前端，只能通过`json`字符串

13.添加学员成绩

- `form`表单点击提交数据以后，需要跳转页面(不写`action`也会刷新当前页面)
- `ajax` 异步数据传输,不会进行页面跳转
  - 两者选其一.

- `form`表单
  - `label`可以让 文字绑定文本框，点击文字，光标定位到文本框
  - `button`按钮的`type`选择`submit`，相当于表单的提交按钮。
- 在后端操作数据库时，向前台反馈消息，模拟`code` + `message`(码+内容)。

```php
// 在php页面先声明 统一返回格式
$responseData = array('code' => 0, 'message' => '');

// 数据库链接失败
if(!$link){
    $responseData['code'] = 1;
    $responseData['message'] = '数据库链接失败';
    // 返回到前台页面
    echo json_encode($responseData);
    exit;
}
```

  - `mysql_query()`发送`sql`各种语句的返回值;
    - `select` 查询数据：返回的是`sql`数据,用`mysql_fetch_assoc($res)`;
    - `insert into` 插入数据 / `update` 修改数据 / `delete` 删除数据: 返回的是布尔值, 成功返回true。

14. 注册前端页面的实现(MD5加密)

- 思路
  1. 创建一个用户表
  2. 表中创建 `id`, `username`, `password`, `create_time` 四个字段
- 数据库中不直接存储密码，存储的是`MD5`加密后的密码
  - 加密生成的一串数字是生成 32位字符 的`md5`码
    - 一层MD5加密：`md5(要加密的数据);`
    - 两层MD5加密：`md5(md5(要加密的数据)."xxx");`
    - 三层MD5加密：`md5(md5(md5(要加密的数据)."xxx")."yyy");`

15.获取用户列表

- 通过查询后解析后的用户数据是 键值数组类型.
- 从数据库取出的数据都是 字符串类型的数据。

16.用户数据的 删除 - 修改

- 事件委托实现
- 思路-删除
  1. 将`a`节点的父节点`td`设置`id`属性,值为当前用户的`id`
  2. 点击删除后，获取`id`,向`deleteUser.php`发送`id`,`get`请求
  3. `php`删除数据
  4. 如果删除成功,直接将删除按钮所在的这一行直接删除，并且删除成功不需要告诉用户
- 思路-修改:点击修改后，跳转到修改页面