### 1.MYSQL：

什么是数据库

数据库服务器：

安装的mysql服务器安装包，将mysql服务器软件安装在电脑上，就可以作为一台数据库服务器

数据库：

数据库服务器中的一个仓库，一个服务器可以创建多个仓库

一般一个网站的所有数据会放在一个数据库中

表：

一个数据库中可以创建多张表

通常情况下，一张表，存放一类数据

表记录：

一张表记录多条记录

数值类型:

tinyint：占用1个字节，相对于java中的byte

smallint：占用2个字节，相对于java中的short

int：占用4个字节，相对于java中的int

bigint：占用8个字节，相对于java中的long

其次是浮点类型即：float和double类型

float：4字节单精度浮点类型，相对于java中的float

double：8字节双精度浮点类型，相对于java中的double

字符串类型：

1、char类型 – 表示定长字符串类型

name char(10) char类型的范围，0~255个字符

char类型在存储数据时，如果存储的数据长度小于最大长度，不足会用空格来补全，因 此char类型可能会造成空间的浪费。

但是，char类型在存储数据的效率上比varchar类型略高一些！

2、varchar类型 – 表示不定长字符串类型

name varchar(200)

varchar类型的范围：mysql5.0之前，0~255个字符

mysql5.0之后， 0 < 字符数\*字节 <= 65535

varchar类型在存储数据时，如果存储的数据长度小于最大长度，剩余的空间会留给别 的数据使用，因此varchar类型不会有空间的浪费！！

但是，varchar类型存储数据的效率上比char类型略低一些！

日期类型：

date 年月日

time 时分秒

datetime 年月日 时分秒

timestamp 时间戳，格式上和datetime相同，都是：年月日 时分秒，设置默认从1970开始才行

datetime和timestamp二者之间的差异：

相同点：都是日期类型，显示的日期格式是相同的

不同点：

(1)二者存储的内容不同：datetime存储的就是年月日时分秒，但是，timestamp存储的一个日期到1970年1月1日之间的时间毫秒值。

(2)二者的范围不同：datetime范围是 1000~9999，timestamp的范围是 1970~2038

(3)timestamp指定的列，可以设置自动更新时间为当前时间。但是datetime则不可以。

字段约束

主键约束 primary key 一般和自增一起用auto\_increment

唯一约束 unique

非空约束 not null

外键约束：foreign key dept\_id references dept(id)

注释：

—— 注释内容 注意中间有空格

#注释内容

Mysql --default-character-set=gbk -uroot -proot //打开mysql时设置编码

show variables like 'char%'; //查看MySQL数据库使用的编码

**create database 库名 charset utf8 //创建库时指定编码**

drop database if exists 库名; //判断是否存在,都能用

在插入(写数据)前设置编码:  **set names gbk;**

insert into 表名( ) values ( ); //插入

update 表名 set 列=值, .....; //修改

delete from 表名; //删除,单id还在

truncate table emp; //删除表中所有记录(将表摧毁再重建),防止id的自增

select distinct dept,job from emp; //剔除重复的记录

select name,sal+ifnull(bonus,0) as 总薪资 from emp //as取别名

where sal+ifnull(bonus,0)>3500; //ifnull判断是否有空,用第二个值替换

select name,sal from emp where sal between 3000 and 4500; //在...和...之间的

select name,sal from emp where sal in (1400,1600,1800); //在集合里的

not in //不在集合中的

select \* from emp where dept is null; //判断是否为空 /is not null

select name from emp where name like '刘%'; //like模糊查询,%：0到多个字符,\_：一个字符

select name,sal from emp order by sal asc; //排序asc升序

desc降序

select \* from emp group by dept; //分组

/\* 聚合函数:

(1)max(colname)、min(colname) -- 求某一列中的最大值、最小值

(2)sum(colname) -- 统计某一列的数据之和

(3)avg(colname) -- 求某一列的平均值

(4)count(\*) -- 统计查询结果的记录行数

-- 聚合函数会自动剔除null值(即null值不会参数计算或统计)

-- 当没有分组时,聚合函数是对整个查询结果进行计算或者统计(可以将整个查询结果看作是一个组)

-- 如果进行了分组,聚合函数是对每一个组的记录进行计算或者统计

-- 聚合函数不能用在where子句中!!

\*/

select \* from emp where month(curdate())=month(birthday);

year(date)、month(date)、day(date) //只获取日期中的年\月\日

curdate() 获取当前日期 年月日

curtime() 获取当前时间 时分秒

sysdate() 获取当前日期+时间 年月日 时分秒

now() 获取当前日期+时间 年月日 时分秒

/\* order by 排序的列 asc 升序(从低到高)

order by 排序的列 desc 降序(从高到低)

默认就是升序，所以asc可以省略不写 \*/

/\* 在mysql中，通过limit进行分页查询：

limit (页码-1)\*每页显示记录数, 每页显示记录数 \*/

select \* from emp limit 0,3; 第一页

select \* from emp limit 3,3; 第二页

一般用来求前几名：如前五名 limit 5;

关联查询

select \* from dept,emp where emp.dept\_id = dept.id;

外连接查询（左left右right外连接）

select \* from dept left join emp on emp.dept\_id=dept.id;

/\* where和having都用于筛选过滤，但是：

where用于在分组之前进行筛选, having用于在分组之后进行筛选

where中不能使用列别名和聚合函数, having中可以使用别名和聚合函数

因为where子句比select先执行!!\*/

SQL语句的书写顺序：

select...

from...

where...

group by...

having...

order by...

limit...

...

SQL语句的执行顺序：

from... -- 确定要查询的是哪张表 (定义表别名)

where... -- 从整张表的数据中进行筛选过滤

select... -- 确定要显示哪些列 (定义列别名)

group by... -- 根据指定的列进行分组

having... -- 对分组之后的数据进行筛选过滤

order by... -- 根据指定的列进行排序

...

JDBC：

Java DataBase Connectivity Java数据库路连接

JDBC的六个步骤：

1. 注册数据库驱动

Class.forName(“com.mysql.jdbc.Driver”);

1. 获取数据库连接

Connection conn = DriverManager.getConnection(

“jdbc:mysql://localhost:3306/jt\_db?characterEncoding=utf-8”,

“root”,”root”);

1. 获取传输器

Statement stat = conn.createStatement();

1. 执行sql语句，返回执行结果

String sql = "select \* from account";

ResultSet rs = stat.executeQuery(sql); //执行查询操作时用

Int rows = stat.executeUpdate(sql); //执行加删改操作时用

1. 处理结果

while(rs.next()) { //使指向数据行的索引向下移动一行

int id = rs.getInt("id");

String name = rs.getString("name");

1. 释放资源

放在finally代码块中，每一个对象都有单独的一个try/catch/finally

SQL注入攻击：由于SQL语句是拼接而来的。其中的参数是由用户提交的，如果用户在提交参数时，在其中掺杂了一些SQL关键字或者特殊符号，就会跳过验证

如：加注释 aa’# 加或 a or ‘1=1

使用preparedStatement对象，是Statement的子接口，比Statement更安全

String sql = "select \* from user where username=? and password=?"; //用？占位符

preparedStatement ps = conn.prepareStatement(sql); //获取传输器，并发送sql语句骨架

ps.setString(1, user); //设置参数

ps.setString(2, pwd);

rs = ps.executeQuery(); //提交

c3p0数据库连接池

创建数据库连接池：

ComboPooledDataSource pool = new ComboPooledDataSource();

方式一：直接写

pool.setDriverClass("com.mysql.jdbc.Driver");

pool.setJdbcUrl("jdbc:mysql:///jt\_db?characterEncoding=utf-8");

pool.setUser("root");

pool.setPassword("root");

方式二：通过c3p0-config.xml文件

<?xml version="1.0" encoding="UTF-8"?>

<c3p0-config>

<default-config>

   <property name="driverClass">

      com.mysql.jdbc.Driver

    </property>

  <property name="jdbcUrl">

     jdbc:mysql:///jt\_db?useUnicode=true&amp;characterEncoding=utf-8

  </property>

  <property name="user">

     root

  </property>

  <property name="password">

     root

  </property>

</default-config>

</c3p0-config>

方式三：通过c3p0.properties文件

c3p0.driverClass=com.mysql.jdbc.Driver

c3p0.jdbcUrl=jdbc:mysql:///jt\_db

c3p0.user=root

c3p0.password=root

从连接池获取连接

conn = pool.getConnection();

释放资源

conn.close 不是关闭连接，而是别c3p0连接池封装为还连接

数据库事务(Database Transaction)，是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。

事务的四大特性(ACID)是：

(1)原子性（Atomicity）：事务中所有操作是不可再分割的原子单位。事务中所有操作要么全部执行成功，要么全部执行失败。

(2)一致性（Consistency）：事务执行后，数据库状态与其它业务规则保持一致。如转账业务，无论事务执行成功与否，参与转账的两个账号金额之和应该是不变的。

(3)隔离性（Isolation）：隔离性是指在并发操作中，不同事务之间应该隔离开来，使每个并发中的事务不会相互干扰。也就是说，在事务中查看数据更新时，数据所处的状态要么是另一事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看到中间状态的数据。例如：在A事务中，查看另一B事务(正在修改张三的账户金额)中张三的账户金额，要查看到B事务之前的张三的账户金额，要么查看到B事务之后张三的账户金额。

(4)持久性（Durability）：一旦事务提交成功，事务中所有的数据操作都必须被持久化到数据库中，即使提交事务后，数据库马上崩溃，在数据库重启时，也必须能保证通过某种机制恢复数据。

Mysql中的事务：

开启事务：start transaction；

结束事务：commit（提交事务）或rollback（回滚事务）。

事务并发读问题：多个事务对相同的数据同时进行操作，这叫做事务并发。

(1)脏读（dirty read）：读到另一个事务的未提交更新数据，即读取到了脏数据；

例如：A给B转账100元但未提交事务，在B查询后，A做了回滚操作，那么B查询到了A未提交的数据，就称之为脏读。

(2)不可重复读（unrepeatable read）：对同一记录的两次读取不一致，因为另一事务对该记录做了修改（是针对修改操作）

例如：在事务1中，前后两次查询A账户的金额，在两次查询之间，另一事物2对A账户的金额做了修改，此种情况可能会导致事务1中，前后两次查询的结果不一致。这就是不可重复度

1. 幻读（虚读）（phantom read）：对同一张表的两次查询不一致，因为另一事务插入了一条记录(是针对插入或删除操作)；

事务隔离级别：

1、READ UNCOMMITTED（读未提交数据）

安全级别最低, 可能出现任何事务并发问题(比如脏读、不可以重复读、幻读等)

性能最好（不使用!!）

2、READ COMMITTED（读已提交数据）（Oracle默认）

防止脏读，没有处理不可重复读，也没有处理幻读；

性能比REPEATABLE READ好

3、REPEATABLE READ（可重复读）（MySQL默认）

防止脏读和不可重复读，不能处理幻读问题；

性能比SERIALIZABLE好

4、SERIALIZABLE（串行化）

不会出现任何并发问题，因为它是对同一数据的访问是串行的，非并发访问的；

性能最差；

MySQL的默认隔离级别为REPEATABLE READ，即可以防止脏读和不可重复读

Mysql设置事务级别

set tx\_isolation='read-uncommitted';

### 2.HTML+CSS：

HTML：超文本标记语言

结构：

<!DOCTYPE html> <!--文档声明声明当前html的版本为5.0 -->

<html>

    <head>

<meta charset="utf-8" />

       <title>网页的标题</title>

    </head>

    <body>

       Hello CGB1808...

    </body>

</html>

标签：

开始标签和结束标签：<from></from>

自闭标签：<img /> 也可以写成上面那种

空格：&nbsp; &emsp;中文空格 换行<br/>

常用标签

图像标签：

<img src="img/1.jpg" width="70%" border="5px"/>

相对路径 宽度 边框

超链接：

<a href="http://www.baidu.com" target="\_blank">百度一下，你就不知道</a>

点击后跳转的地址 \_self：默认值，在当前窗口打开超链接

\_blank：在新的窗口中打开超链接

表格标签：

<table>

    <tr>

       <td>11</td>

    </tr>

    <tr rowspan="2">

       <td colspan="2">21</td>

    </tr>

</table>

<table> -- 用于定义一个表格

<tr> -- 用于定义表格中的行

<td> -- 用于定义表格行中的单元格

<th> -- 也是单元格, 是表头中的单元格

其中的colspan是设置单元格横跨的列数

另外rowspan是设置单元格竖跨的行数

表单标签：

<form action="#" method="GET"></form>

action 必须存在的属性，用来指定表单提交的目的地地址

method 可选属性，用来指定以何种方式来提交表单，如果不指定，默认是GET提交

提示1：在HTTP协议中规定了7种提交方式，其中5种都不常用，只用GET和POST。

提示2：只有使用表单，并且明确的指定了提交方式为POST时，才是POST提交，其他方式都是GET提交。

表单项标签：

input标签

(1)作为普通文本输入框

<input type="text" name="username"/>

(2)作为密码输入框

<input type="password" name="password"/>

(3)作为单选框

<input type="radio" name="gender" value="male" checked="checked"/>男

/\*name值相同表示为一个组，防止多选，value设置选中时提交的值，

checked设置为默认选中\*/

(4)作为复选框

<input type="checkbox" name="like" value="basketball"/>篮球

(5)提交按钮

<input type="submit" value="提交"/>

(6)普通按钮

<input type="button" value="点我"/>

select、option标签 -- 实现下拉选框

<select name=*"city"*>

    <option value=*"bj"*>北京</option> /\*改变提交时的值\*/

    <option>上海</option>

    <option selected=*"selected"*>广州</option> /\*设置为默认被选中\*/

    <option>深圳</option>

</select>

textarea标签 -- 多行文本输入框

<textarea rows=*"5"* cols=*"30"*  name=*"description"*

    placeholder=*"请输入描述信息..."*></textarea>

rows用于指定行数, clos用于指定列数，placeholder用于指定输入框中的提示消息

引用css的三种方式：

1. 通过style属性引入css
2. 通过style标签引入css
3. 通过link链接引入外部的css文件

<link href=*"demo.css"*  rel=*"stylesheet"*/>

选择器：

标签名选择器：

格式：标签名/元素名{ css属性... }

class选择器：

格式：.class值{ css属性... }

id选择器：

格式：#id{ css属性... }

后代选择器：

格式: 父选择器 后代选择器{ css样式… }

属性选择器

格式: 选择器[属性条件1][属性条件2]…{ css样式 }

### JavaScript+JQuery：

1、特点

(1)JS是一门直译式的语言，直接执行源代码（是一边解释一边执行，不像Java需要提前编译为class文件再运行）

(2)JS是一门弱类型的语言

2、优势：

(1)良好的交互性

(2)一定的安全性

(3)跨平台性（有浏览器的地方就可以执行JS）

JavaScript获取标签的六种方法：

document.getElementById('id名');

document.getElementsByName('name名'); 集合----数组选择[0]

document.getElementsByTagName(‘标签名’); 集合----数组选择[0]

document.getElementsByClassName('class名'); 集合----数组选择[0]

document.querrySelector('可以写任何选择器.或# 只能选择一个');

document.querrySelectorAll('可以写任何选择器.或# 可以选择多个');

document.body-- 获取当前文档中的body元素

obj.parentNode-- 获取当前元素的父元素。obj表示当前元素