

# 数据库工程作业

要求:

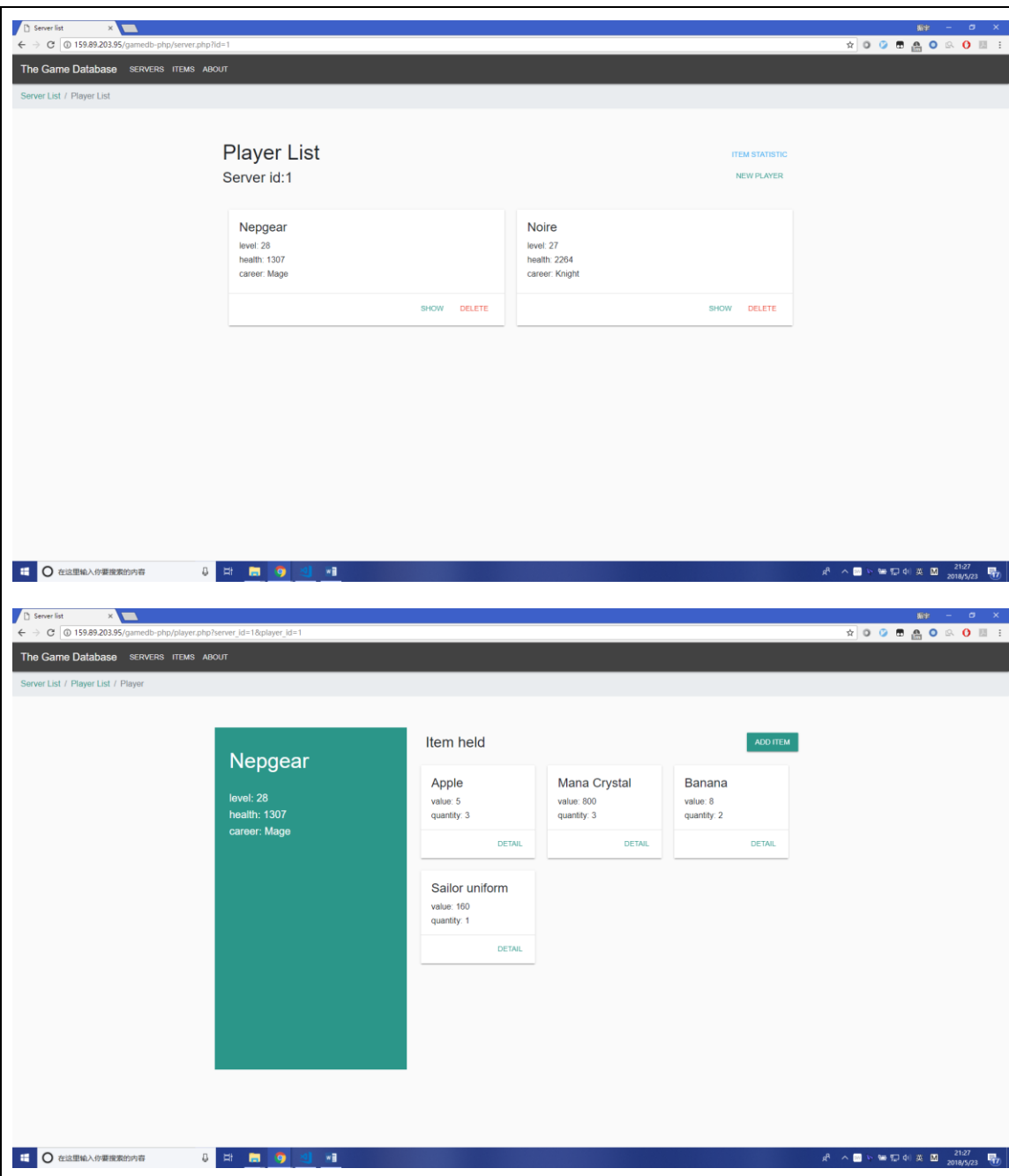
- 1. 完成一个小型的数据库信息管理系统（或部分功能），并填写工程作业报告；程序和报告请在规定时间之内上传。
- 2. 开发模式（B/S 或 C/S）、开发高级语言任选，后台数据库使用大型数据库管理系统（SQL Server、Oracle 等），不要使用桌面数据库。
- 3. 报告中所列举的四种操作，每种操作举一个例子即可。
- 4. 作业成绩按照报告中的标准评分，程序只实现报告中涉及的部分即可。

## 工程作业报告

1. 项目信息（10 分）

学号	1610842	姓名	左振宇	专业	计算机科学与技术
项目名称	Game Database - php				
必备环境	Apache 2.4, PHP 7.2, PostgreSQL 9.6				
系统主要功能简介（4分）	<p>系统可以管理某一 MMORPG 中服务器、玩家和物品之间的关系。并能够按照需要进行查询、增加、更新和删除操作。</p> <p>测试用的网页已部署，可以通过 <a href="http://159.89.203.95/gamedb-php/servers-list.php">http://159.89.203.95/gamedb-php/servers-list.php</a> 访问。</p>				

系统主要页面截图（6分）



Server list

159.89.203.95/gameadb.php/item-statistics.php?server\_id=1

The Game Database

SERVICESITEMSABOUT

Server List / Player List / Item Statistics

Item Statistics

Server id:1

Item ID	Name	Value	Total Quantity	Total Value
1	Apple	5	15	75
3	Mana Crystal	800	3	2400
5	Banana	8	2	16
6	Sailor uniform	160	1	160
7	Axe	80	0	0
8	Master Sword	6750	0	0
9	Black Sword	933	1	933

在这里输入你要搜索的内容

21:272018/5/23

Server list

159.89.203.95/gameadb.php/items-list.php

The Game Database

SERVICESITEMSABOUT

Item List

All items in this game will be displayed here.

NEW ITEM

Item ID	Name	Value	Operation
1	Apple	5	DELETE
3	Mana Crystal	800	DELETE
5	Banana	8	DELETE
6	Sailor uniform	160	DELETE
7	Axe	80	DELETE
8	Master Sword	6750	DELETE
9	Black Sword	933	DELETE

在这里输入你要搜索的内容

21:272018/5/23

## 2. 系统配置（10 分）

说明		(2 分) 请说明系统配置情况（后台数据库，高级语言）； (8 分) 请使用连接串连接高级语言和数据库，并分析字符串的各个部分。			
配置 步骤 2 分	DBMS	PostgreSQL 9.6			
	高级 语言	1. PHP			
		2. HTML5			
连接串 分析 (6 分)		序号	名称	功能说明	取值
		1	\$host	指定数据库地址	"host=127.0.0.1"
		2	\$port	指定数据库端口	"port=5432"
		3	\$dbname	指定数据库名称	"dbname=gamedb"
		4	\$credentials	指定用户名和密码	"user=gamedbuser password=gPassword"
连接串代码 (截屏) (2 分)		<pre>\$host      = "host=127.0.0.1"; \$port      = "port=5432"; \$dbname    = "dbname=gamedb"; \$credentials = "user=gamedbuser password=gPassword";</pre>			
备注					

### 3. 数据库设计（14 分）

说明	<p>（10 分）按照数据表的创建顺序，依次给出所涉及数据表的信息，其中参照字段以“（字段 1，字段 2，……，字段 n）”的形式给出，被参照字段以“表名（字段 1，字段 2，……，字段 n）”的形式给出；</p> <p>（4 分）一般 DBMS 都可以为数据库生成关系图，请将该图片截屏并粘贴到表格中。</p>				
数据表 (10)	创建顺序	数据表名称	主键	参照属性	被参照表及属性
	1	servers	id		
	2	players	server_id, id	server_id	servers(id)
	3	items	id		
	4	holds	server_id, player_id, item_id	server_id, player_id, item_id	players(server_id, id), items(id)
关系图 (4)	<pre> erDiagram     servers   --o{ players : "server_id"     servers   --o{ items : "id"     players   --o{ holds : "server_id, id"     items   --o{ holds : "item_id"     </pre>				
备注	PostgreSQL 没有提供生成关系图的功能，故使用了 SQL Server 来生成关系图。				

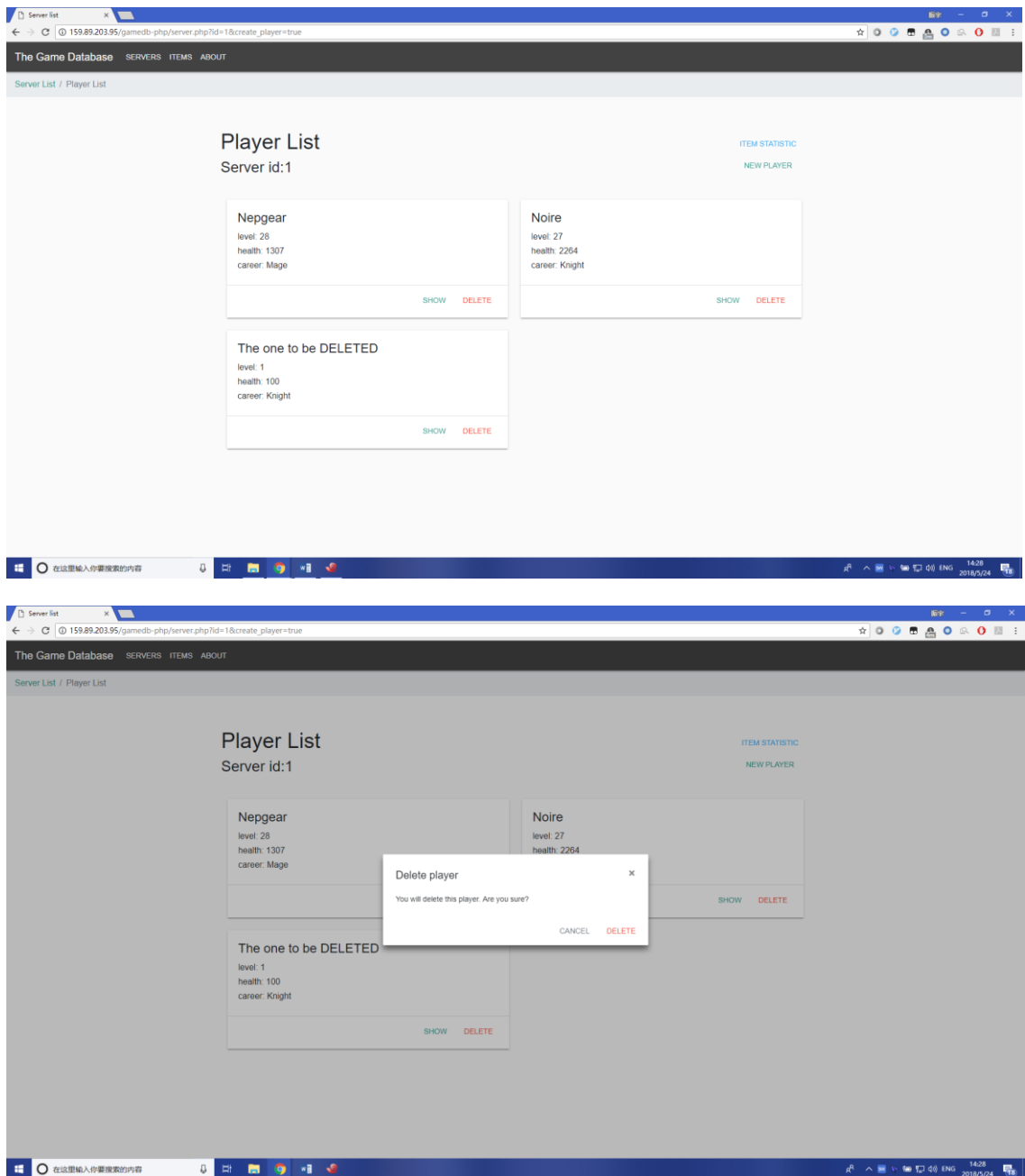
#### 4. 含有事务应用的删除操作（13 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能;</p> <p>(2 分) 该操作会涉及的表 (必须含有两张或两张以上的关系表, 同时以“表名”的形式给出)</p> <p>(1 分) 表连接涉及字段描述 (描述方式为“表 1. 属性=表 2. 属性”)</p> <p>(1 分) 删除条件涉及的字段描述 (以“表名. 属性=? ”形式给出)</p> <p>(4 分) 实现该操作的关键代码 (高级语言、SQL), 截图即可; (其中如果删除语句中不包含任何形式的事务应用将扣除 3 分)</p> <p>(4 分) 如何执行该操作, 按所述方法能够正常演示程序则给分。</p>	
功能描述 (1 分)	删除一个 player 时, 需要同时删除 holds 表中相应的数据。使用事务可以避免在删除操作执行失败时导致的数据库不一致。	
涉及的表 (2 分)	players, holds	
表连接涉及字段 (1 分)	holds.player_id = players.id, holds.server_id = players.server_id	
删除条件字段描述 (1 分)	字段	规则
	players.id	players.id = \$player_id
	players.server_id	players.server_id = \$id
	holds.player_id	holds.player_id = \$player_id
	holds.server_id	holds.server_id = \$id

代码  
(4分)

```
if(isset($_POST['destroy'])) {
    $player_id = $_POST['destroy'];
    pg_query($db, "BEGIN TRANSACTION;");
    $query1 = pg_query($db, "DELETE FROM holds WHERE server_id = $id AND player_id = '$player_id' ;");
    $query2 = pg_query($db, "DELETE FROM players WHERE server_id = $id AND id = '$player_id' ;");
    if ($query1 and $query2) {
        pg_query($db, "COMMIT;");
        header("Location: " . 'http://' . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'] . '?' . 'id=' . $id . '&delete_player=true');
        exit;
    } else {
        pg_query($db, "ROLLBACK;");
        header("Location: " . 'http://' . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'] . '?' . 'id=' . $id . '&delete_player=false');
        exit;
    }
}
```

程序演示  
(4分)



	<div><div>Server list</div><div>159.89.203.95/gamedb-php/server.php?id=1&amp;delete_player=true</div><div>The Game Database SERVERS ITEMS ABOUT</div><div>Server List / Player List</div><div><div>Player List</div><div>Server id:1</div><div><a href="#">ITEM STATISTIC</a> <a href="#">NEW PLAYER</a></div><div>Successfully deleted player.</div><div><div><div>Nepgear</div><div>level: 28 health: 1307 career: Mage</div><div>SHOW DELETE</div></div><div><div>Noire</div><div>level: 27 health: 2264 career: Knight</div><div>SHOW DELETE</div></div></div></div></div>
备注	



## 5. 触发器控制下的添加操作（20 分）

说明	(1 分) 简要说明该操作所要完成的功能; (2 分) 简要说明该触发器所要完成的功能 (1 分) 该操作会涉及的表 (以 “表名” 的形式给出)。 (2 分) 该操作输入数据以及输入数据应该满足的条件, 如: 数值范围、是否为空; (6 分) 实现该操作的关键代码 (高级语言、SQL), 截图即可; (8 分) 如何执行该操作, 按所述方法能够正常演示程序则给分。	
功能描述 (1 分)	players 表中的条目代表角色的属性, 其中 level 和 health 不能小于或等于 0。该触发器的功能是在向 players 表中添加数据时检查上述属性是否小于或等于 0, 若如此则拒绝此次操作。	
触发器描述 (2 分)	依次判断 level 和 health 是否为空或是否小于或等于零, 若是, 则抛出异常; 否则返回新插入的条目。	
涉及的表 (1 分)	players	
输入数据 (2 分)	字段	规则
	NEW.level	NEW.level != NULL && NEW.level >= 0
	NEW.health	NEW.health != NULL && NEW.health >= 0
插入操作源码 (3 分)	<pre> if(isset(\$_POST['new_player'])) {     \$new_name = \$_POST['new_name'];     \$new_level = \$_POST['new_level'];     \$new_health = \$_POST['new_health'];     \$new_career = \$_POST['new_career'];     \$query = pg_query(\$db, "INSERT INTO players(server_id, name, level, health, career) VALUES(\$id, '\$new_name', \$new_level, \$new_health, '\$new_career');");     //Header("Location: " . 'http://' . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . '?' . 'id=' . \$id . '&amp;create_player=true');     if (\$query) {         header("Location: " . 'http://' . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . '?' . 'id=' . \$id . '&amp;create_player=true');         exit;     } else {         header("Location: " . 'http://' . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . '?' . 'id=' . \$id . '&amp;create_player=false');         exit;     } } </pre>	

触发器源码  
(3分)

```
CREATE FUNCTION player_check() RETURNS trigger AS $player_check$
BEGIN
    IF NEW.level IS NULL THEN
        RAISE EXCEPTION 'level cannot be null';
    END IF;

    IF NEW.level <= 0 THEN
        RAISE EXCEPTION '% cannot have a <=0 level', NEW.level;
    END IF;

    IF NEW.health IS NULL THEN
        RAISE EXCEPTION 'health cannot be null';
    END IF;

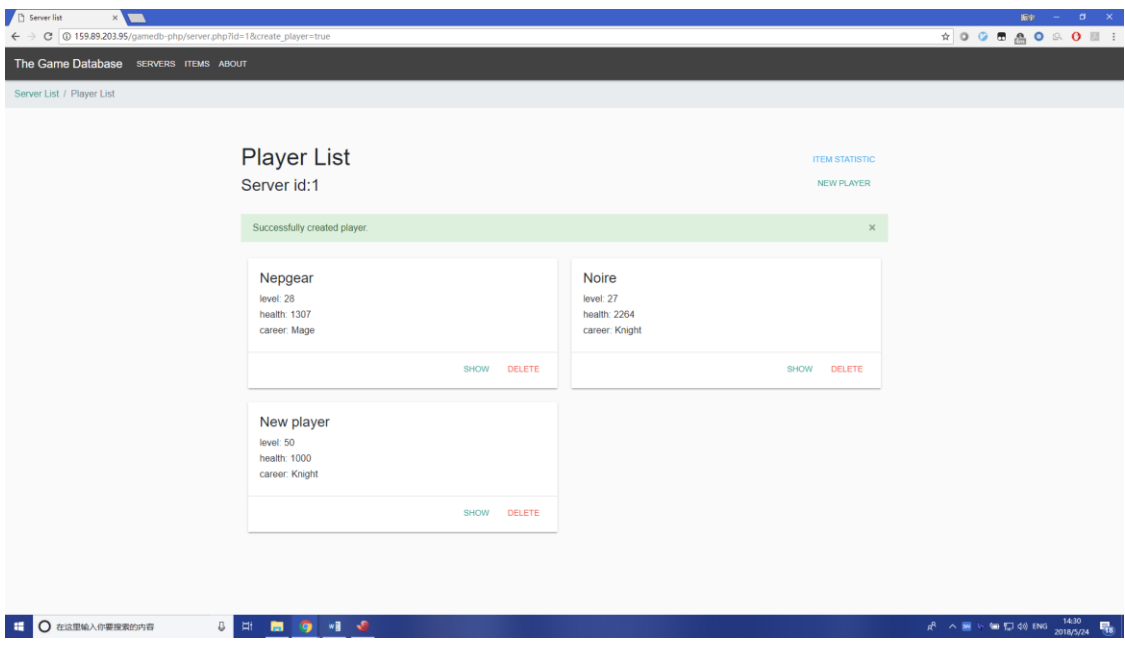
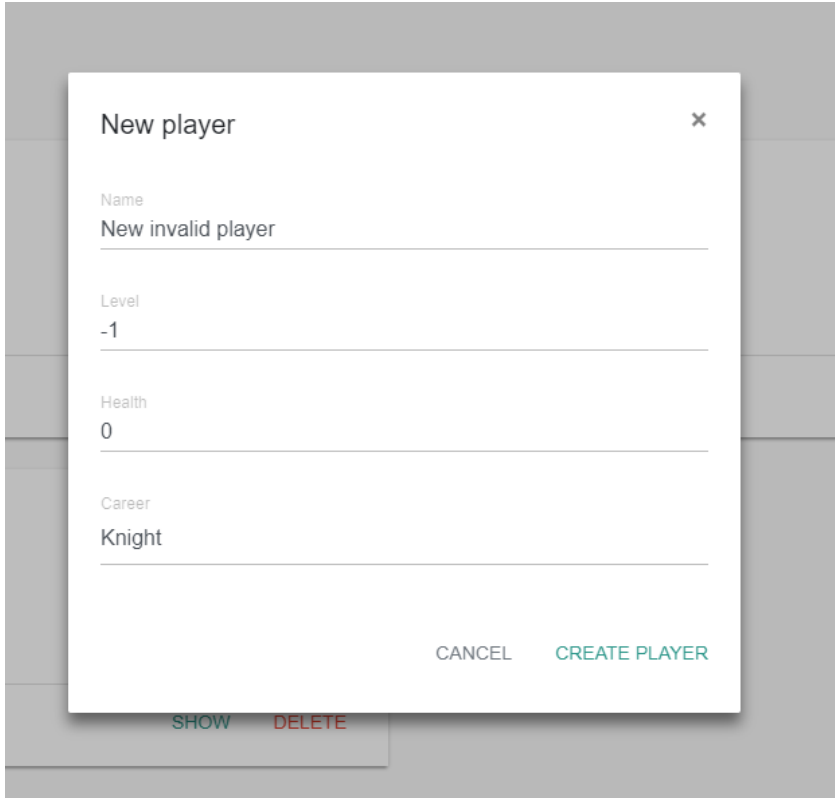
    IF NEW.health <= 0 THEN
        RAISE EXCEPTION '% cannot have a <=0 health', NEW.health;
    END IF;

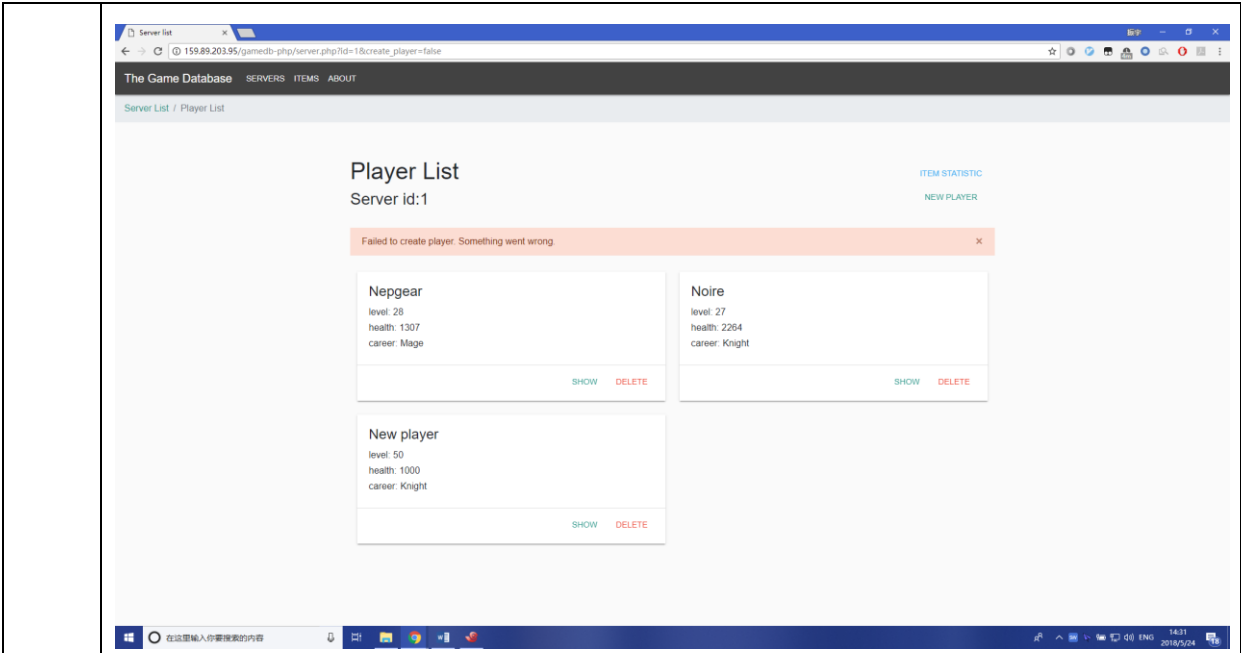
    RETURN NEW;
END;
$player_check$ LANGUAGE plpgsql;

CREATE TRIGGER player_check BEFORE INSERT OR UPDATE ON players
FOR EACH ROW EXECUTE PROCEDURE player_check();
```

程序演示  
(4分)

The image shows a 'New player' dialog box with a close button (X) in the top right corner. It contains four input fields with labels and values: 'Name' with 'New player', 'Level' with '50', 'Health' with '1000', and 'Career' with 'Knight'. At the bottom, there are two buttons: 'CANCEL' and 'CREATE PLAYER'.

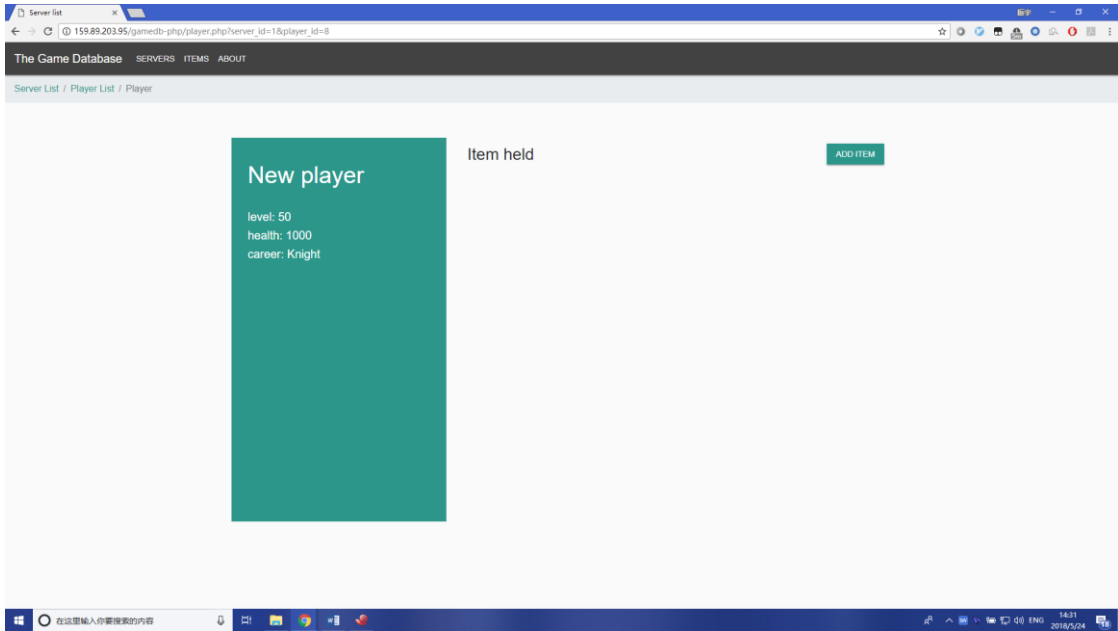
	
程序 演示 (4 分)	



备注 PostgreSQL 中作为触发器的函数并没有显式输入的参数。

## 6. 存储过程控制下的更新操作（18 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能；</p> <p>(1 分) 简要说明该存储过程所要完成的功能；</p> <p>(2 分) 说明该操作涉及操作的表（必须包含两张或两张以上的关系表，以“表名形式”描述）</p> <p>(1 分) 表连接涉及字段描述（描述方式为“表 1. 属性=表 2. 属性”）</p> <p>(2 分) 该操作会修改字段（以“表名. 字段名”的形式给出），以及修改规则，如新数值的计算方法、在何种条件下予以修改等；</p> <p>(6 分) 实现该操作的关键代码（高级语言、SQL），截图即可；</p> <p>(5 分) 如何执行该操作，按所述方法能够正常演示程序则给分。</p>	
功能描述 (1 分)	<p>在给玩家添加物品时，需要判断物品 id 是否存在于 items 中，否则拒绝操作；还需判断该玩家是否已持有该物品，若已持有，则更新 holds 表中相应的条目为数值之和，若未持有，则向 holds 表中新添加对应的条目。</p>	
存储过程功能描述 (1 分)	<p>输入四个参数 m_server_id, m_plsyer_id, m_item_id 和 m_quantity，首先判断 m_quantity 是否小于等于零，若是，抛出异常；然后判断 m_item_id 是否存在于 items 表中，若否，抛出异常；最后判断 holds 中是否已有相应的条目，若是，则更新条目的 quantity = quantity + m_quantity；若否，则新建对应的条目。</p>	
涉及的关系表 (2 分)	<p>holds, items</p>	
表连接涉及字段 (1)	<p>holds.item_id = items.id</p>	
更改字段 (2 分)	字段	规则
	quantity	<p>IF exist_in_holds(server_id, player_id, item_id)</p> <p>    quantity + m_quantity</p> <p>ELSE</p> <p>    quantity</p>
更新代码 (3 分)	<pre> &lt;?php if(isset(\$_POST['add_item'])) {     \$new_id = \$_POST['new_id'];     \$new_quantity = \$_POST['new_quantity'];     \$query = pg_query(\$db, "SELECT * FROM add_item(\$server_id, \$player_id, \$new_id, \$new_quantity);");     //Header("Location: " . "http://" . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . "?id=\$id&amp;create_player=true");     if (\$query) {         header("Location: " . "http://" . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . "?server_id=\$server_id&amp;player_id=\$player_id&amp;add_item=true");         exit;     } else {         header("Location: " . "http://" . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . "?server_id=\$server_id&amp;player_id=\$player_id&amp;add_item=false");         exit;     } } </pre>	

<p>创建 存储 过程 源码 (3 分)</p>	<pre>CREATE OR REPLACE FUNCTION add_item(m_server_id integer, m_plsyer_id integer, m_item_id integer, m_quantity integer) RETURNS boolean AS \$\$ BEGIN     IF m_quantity &lt;= 0 THEN         RAISE EXCEPTION 'Quantity must greater than 0.';     END IF;      IF NOT EXISTS(SELECT * FROM items WHERE id = m_item_id) THEN         RAISE EXCEPTION 'Item not exist.';     END IF;      IF EXISTS(SELECT * FROM holds WHERE m_server_id = server_id AND m_plsyer_id = player_id AND m_item_id = item_id) THEN         UPDATE holds SET quantity = m_quantity + quantity WHERE m_server_id = server_id AND m_plsyer_id = player_id AND m_item_id = item_id;     ELSE         INSERT INTO holds(server_id, player_id, item_id, quantity) VALUES(m_server_id, m_plsyer_id, m_item_id, m_quantity);     END IF;      RETURN TRUE; END; \$\$ LANGUAGE plpgsql;</pre>
<p>存储 过程 执行 源码 (1 分)</p>	<pre>&lt;?php if(isset(\$_POST['add_item'])) {     \$new_id = \$_POST['new_id'];     \$new_quantity = \$_POST['new_quantity'];     \$query = pg_query(\$db, "SELECT * FROM add_item(\$server_id, \$player_id, \$new_id, \$new_quantity);");     //Header("Location: " . "http://" . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . "?id=" . \$id . "&amp;create_player=true");     if (\$query) {         header("Location: " . "http://" . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . "?server_id=" . \$server_id . "&amp;player_id=" . \$player_id . "&amp;add_item=true");         exit;     } else {         header("Location: " . "http://" . \$_SERVER['HTTP_HOST'] . \$_SERVER['PHP_SELF'] . "?server_id=" . \$server_id . "&amp;player_id=" . \$player_id . "&amp;add_item=false");         exit;     } }</pre>
<p>程序 演示 (2 分)</p>	 <p>The screenshot shows a web browser window with the URL 159.89.203.95/gamesdb-php/player.php?server_id=1&amp;player_id=8. The page title is 'The Game Database' with links for SERVERS, ITEMS, and ABOUT. The breadcrumb trail is 'Server List / Player List / Player'. The main content area features a green card titled 'New player' with the following details: level: 50, health: 1000, and career: Knight. To the right of the card is the text 'Item held' and a green 'ADD ITEM' button.</p>

Add Item

Item ID

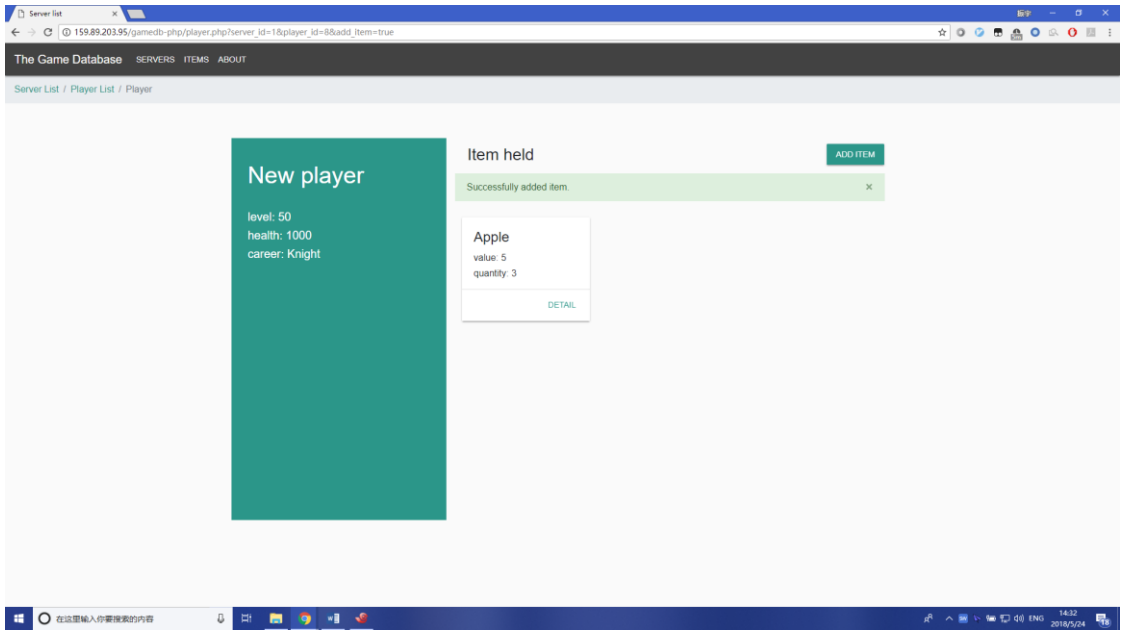
1

Quantity

3

CANCEL

ADD ITEM



Add Item

Item ID

1

Quantity

5

CANCEL

ADD ITEM

Server list

159.89.201.95/gamedb- php/player.php?server\_id=1&player\_id=8&add\_item=true

The Game Database

SERVICES ITEMS ABOUT

Server List / Player List / Player

New player

level: 50  
health: 1000  
career: Knight

Item held

Successfully added item.

Apple  
value: 5  
quantity: 8

DETAIL

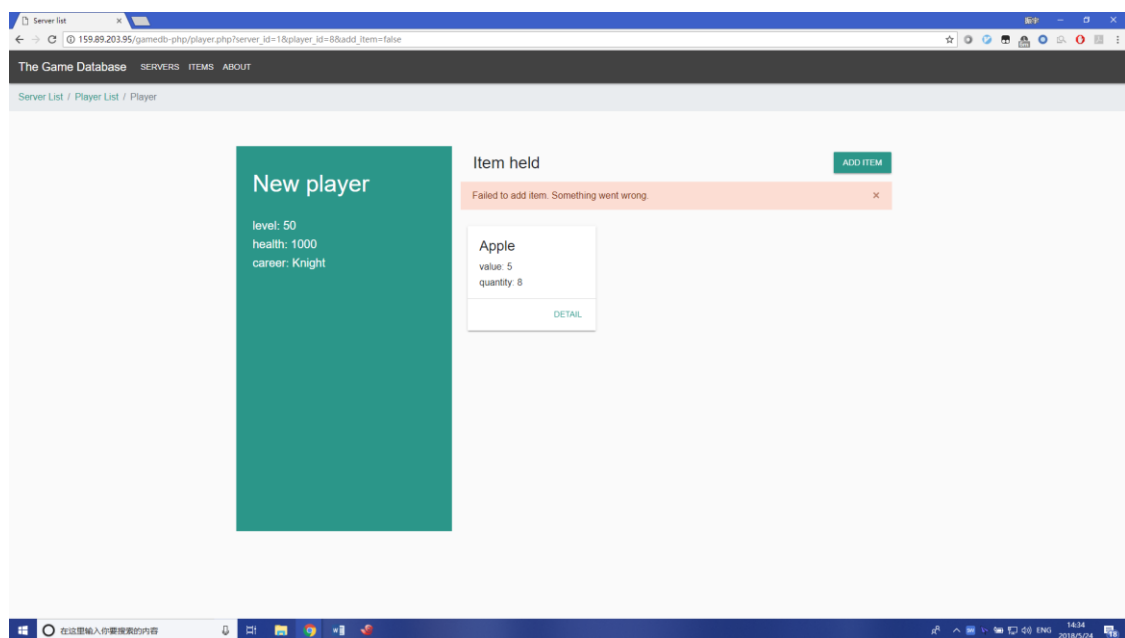
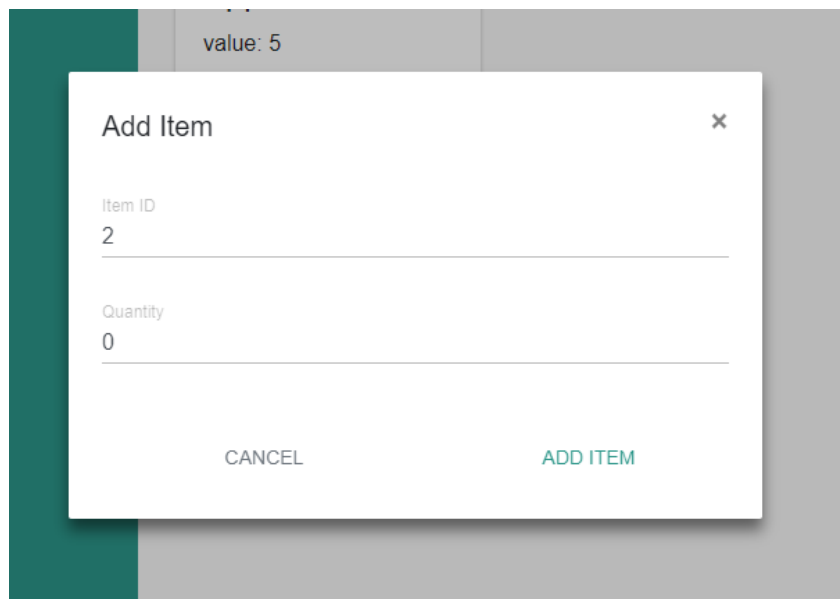
ADD ITEM

在这里输入你要搜索的内容

18:33  
2018/5/24



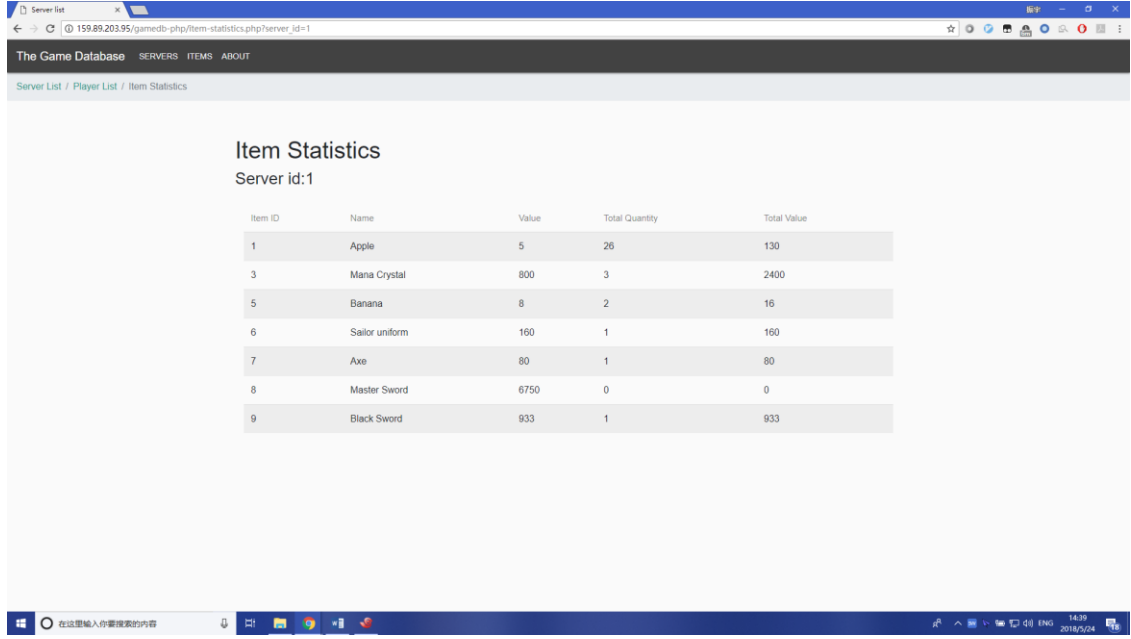
程序  
演示  
(2  
分)



备注 PostgreSQL 中，存储过程以函数的形式存储和执行。

## 7. 含有视图的查询操作（15 分）

说明	<p>(1 分) 简要说明该操作所要完成的功能;</p> <p>(1 分) 简要说明建立的该视图的功能;</p> <p>(2 分) 简要说明该操作涉及的关系数据表 (以 “表名” 的形式给出)</p> <p>(1 分) 简要说明表连接涉及的字段 (以 “表 1. 属性=表 2. 属性”)</p> <p>(6 分) 实现该操作的关键代码 (高级语言、SQL), 截图即可;</p> <p>(4 分) 如何执行该操作, 按所述方法能够正常演示程序则给分。</p>
操作功能描述 (1 分)	查询每个服务器内各种物品的总数以及总价值。
视图功能描述 (1 分)	从 holds 表中选择不重复的 (server_id, item_id) 行, 对 quantity 属性求和作为视图的 count 属性, 并添加 items 表中对应条目的 name 和 value。
涉及的关系表 (2 分)	holds, items
表连接字段 (1 分)	holds.item_id = items.id
创建视图代码	<pre>CREATE VIEW item_count AS SELECT h.server_id, h.item_id, i.name, i.value, Sum(h.quantity) AS "count" FROM holds h, items i WHERE h.item_id = i.id GROUP BY h.server_id, h.item_id, i.name, i.value;</pre>

码 ( 3 分)	
查 询 代 码 (3 分)	<pre>\$item_list = pg_query(\$db, "SELECT * FROM items;"); while (\$row = pg_fetch_row(\$item_list)) {     \$item_c = pg_query(\$db, "SELECT * FROM item_count WHERE server_id = \$server_id AND item_id = \$row[0];");     \$item_id = \$row[0];     \$item_name = \$row[1];     \$item_value = \$row[2];     \$total_quantity = 0;     \$total_value = 0;     while (\$row_c = pg_fetch_row(\$item_c)) {         \$item_id = \$row_c[1];         \$item_name = \$row_c[2];         \$item_value = \$row_c[3];         \$total_quantity = \$row_c[4];         \$total_value = \$total_quantity * \$item_value;     }     echo &lt;&lt;&lt;EOF     &lt;tr&gt;         &lt;td&gt;\$item_id&lt;/td&gt;         &lt;td&gt;\$item_name&lt;/td&gt;         &lt;td&gt;\$item_value&lt;/td&gt;         &lt;td&gt;\$total_quantity&lt;/td&gt;         &lt;td&gt;\$total_value&lt;/td&gt;     &lt;/tr&gt; EOF; }</pre>
程 序 演 示 (4 分)	
备 注	最终在表格中需要显示在服务器中数量为零的物品，所以使用如上所述的视图并非一个好的实现方法。但为了演示从两个不同表中选择属性，所以最终选择此种写法。