DBChat 项目报告

孙培艺，张煌昭

摘要—本次项目使用后端数据库服务和前端 WEB 服务，完成了简单的在线聊天系统。数据库使用 PowerDesigner 进行设计，使用 Mysql 进行实现，其上共有 6 张表，记录了用户(CLIENT)，聊天室(SESSION)，消息(MSG)及其相互关系等信息。数据库上所有操作均通过存储过程（PROCEDURE）实现，以参数化的方法避免注入攻击。前端 WEB 服务使用 Apache 服务器，所有网页均使用 Bootstrap 模板和 PHP 编写。最终 DBChat 聊天系统被部署于云端服务器，访问 http://47.94.138.231/进行试用。

I. 介绍

本次中期项目，使用 B/S 结构实现了一个简单的在线聊天系统。服务器端，即后端数据库使用 PowerDesigner 工具进行设计，基于 Mysql 数据库进行实现。数据库中共有 6 张表，分别为用户（CLIENT），好友(KNOWS)，好友申请(FRIENDAPPLICATION)，聊天室（SESSION），聊天室中的用户（INSESSION）和消息（MSG）。可能需要的所有操作均使用存储过程（PROCEDURE）实现，并且数据库用户仅有执行存储过程（EXECUTE）的权限，以此避免 SQL 注入等非法或攻击动作。WEB 服务使用 Apache 服务器和 Bootstrap 框架进行实现。

本次中期项目由小组完成，小组成员及信息见脚注。本项目所有源代码，开源于 GitHub 平台[1]。本次报告使用 Overeaf L^AT_EX 在线平台编写[2]。

II. 实体-关系设计

本节对 DBChat 数据库的实体-关系模式的设计进行说明。

一个典型的聊天/聊天室系统，应当具备至少 3 类实体——用户（CLIENT），会话（SESSION）和消息（MSG）。这 3 类实体之间也必然存在如下的关系：CLIENT 处于哪些 SESSION 之中，SESSION 的管理员是哪个 CLIENT，MSG 发送至哪个 SESSION，CLIENT 发送了哪些 MSG。考虑到目前常见的社交系统都具有好友功能，用户之间还应具有好友这一关系。此外，为了使用户可以向暂时不是好友的其他用户申请好友，添加一类特殊实体，即好友申请（FRIENDAPPLICATION，简写为 FA），该实体同用户具有如下两类关系：FA 发起者是哪个 CLIENT，向哪个 CLIENT 申请。

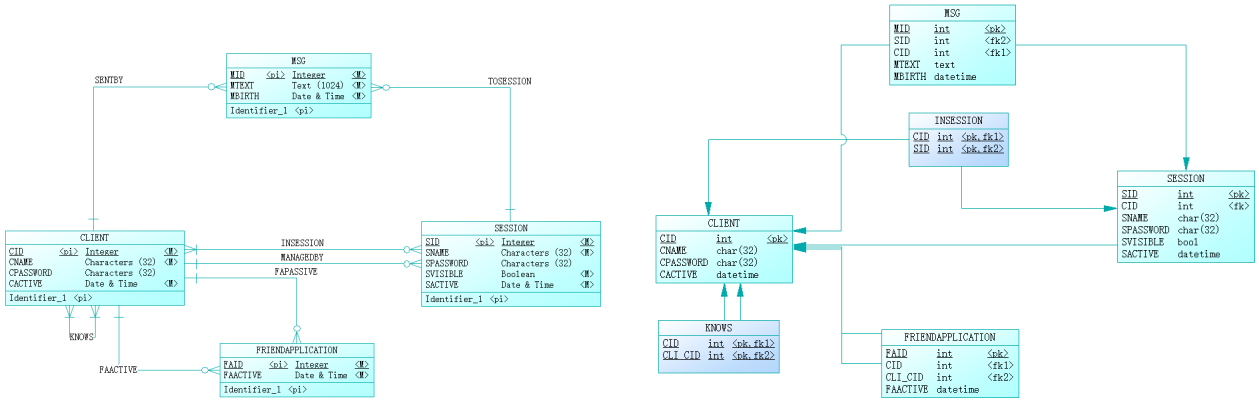
综上该实体-关系模型中，具有四类实体——CLIENT，SESSION，MSG 和 FA，它们之间的关系为——CLIENT 间的好友关系（KNOWS），CLIENT 和 SESSION 间的处于及管理员关系（MANAGER），CLIENT 和 MSG 间的发送关系（SENTBY），CLIENT 和 FA 间的发起方（FAACTIVE）和被申请方（FA-PASSIVE）的关系，MSG 和 SESSION 间的发送关系（TOSESSION）。

接下来对实体的属性进行说明。**CLIENT** 具有编号(CID)，名字(CNAME)和密码(CPASSWORD)属性，其中 CID 作为 CLIENT 的唯一标识，不可重复且非空，CNAME 非空，CPASSWORD 允许空值（允许空密码存在）。**SESSION** 具有编号（SID），名字（SNAME），密码(SPASSWORD)和是否可见(SVISIBLE)属性，其中 SID，SNAME 和 SPASSWORD 设置同 CLIENT。SVISIBLE 是该 SESSION 是“群聊”还是“私聊”的标志，若设置为 TRUE 则表示群聊，该 SESSION 可以被搜索到，允许新的 CLIENT 加入或被邀请，允许已在 SESSION 中的 CLIENT 退出；否则 SVISIBLE 被设置为 FALSE，表示私聊，那么该 SESSION 中有且仅有两个 CLIENT，SESSION 无法被主动搜索到，不允许新的 CLIENT 加入或被邀请加入，不允许已在 SESSION 中的两个 CLIENT 退出，即 SVISIBLE 标明了 SESSION 是否对外界可见，可见的即为群聊，不可见的即为私聊。**MSG** 具有编号（MID）和文本内容（MTEXT），其中 MID 的设置同 CLIENT 和 SESSION，MTEXT 为该消息的文本内容，要求非空（不允许发送空消息）。

按字母序排序。
孙培艺，1500012899，信息科学技术学院，spy@pku.edu.cn
张煌昭，1400017707，元培学院，zhang_hz@pku.edu.cn

[1]项目源码可通过以下 git 命令获得，
git clone git@github.com:LC-John/DBChat.git

[2]报告源码可通过以下 git 命令获得，
git clone https://git.overleaf.com/16484156pkncjbyfdnrv



(a) 逻辑数据模型　　(b) 物理数据模型

图 1. DBChat 数据库的实体-关系模型。图 1(a)为 DBChat 的逻辑数据模型，方框代表实体，其中各行为该实体的属性，方框间的连线代表实体间的关系。关系的基数关系如下——三角 + 横线表示零或多个，三角 + 圆圈表示多个且至少一个，直线 + 横线表示零或一个，直线 + 圆圈表示仅一个且至少一个。例如 MANAGEBY 关系中每个 SESSION 对应一个且仅一个 CLIENT，每个 CLIENT 对应零或一个 SESSION。图 1(b)为自动生成的 DBChat 的物理数据模型，方框代表表，箭头代表表间的引用关系，绿色方框对应逻辑数据模型中的实体，蓝色方框对应逻辑数据模型中的关系（无法纳入实体属性中）。

FA 具有编号（FAID）属性，设置同上。

使用 PowerDesigner 工具进行设计，设计文件均在附件 model 目录下，逻辑数据模型如图 1(a)。各个实体均添加时间属性（BIRTH 或 ACTIVE），用于记录创建时间（BIRTH）或最近一次操作时间（ACTIVE），该属性在数据库实现时被用于过期清理。

将逻辑数据模型转换为物理数据模型，如图 1(b)，之后生成 Mysql 代码，得到附件中 build_db.sql 文件，用于构建数据库。数据库共有 6 张表，分别为 CLIENT，SESSION，MSG，FRIENDAPPLICATION(简写为 FA)，KNOWS，INSESSION，其中 KNOWS 和 INSESSION 是对应了实体-关系模型中的两个多对多关系，它们无法被纳入实体的属性之中，因此单独成表。最终数据库各表及各列如表 I所示，注意数据库结构和各列属性同图 1(b)略有不同，因为在实现过程中对其进行了略微的调整，该部分将于第 III中进行说明。

本节的最后，对 DBChat 数据库的模式进行分析如下。

数据库中共有 6 个表，每个表中都有唯一的主键，除去主键并没有其他的唯一的键，因此候选码唯一，即为主码。满足 1-NF 范式，每个分量都是不可再分的数字或者字符串，此处不考虑将日期时间分割的情况，因为在使用过程中，并没有割离。满足 2-NF 范式，每个非主属性完全依赖于码，大多数表中都只有一个主属性，只有 INSESSION 和 KNOWS 表中有两个属性作为主键，但是没有非主属性，故显然非主属性完全依赖于码。满足 3-NF 范式，不存在非主属性对码的传递依赖，在每一个表中，非主属性都完全取决于主键，故不存在非主属性对码的传递依赖。满足 BCNF 范式，不存在主属性之间的部分依赖及传递函数依赖，大多数表中都只有一个主属性，只有 INSESSION 和 KNOWS 表中有两个属性作为主键，且不存在依赖关系。满足 4-NF 范式，不存在非平凡的多值依赖。

综上所述，DBChat 数据库满足 4-NF 范式的要求。由于数据库中 6 张表中的 4 张（CLIENT，SESSION，MSG，FRIENDAPPLICATION）均对应实体，剩余两张表（KNOWS，INSESSION）对应多对多关系，因而没有办法进行合并。此外，DBChat 的业务中并不会有很多三张表以上的查询，多数增删改查操作均可以通过两表完成，因而可以使用该模式。

III. 数据库实现

本节对于 DBChat 数据库的搭建和实现进行介绍。

数据库后端的 Mysql 脚本均位于附件 sql_script 目录下，对各个脚本的详细说明和解释请见目录下 README.md 以及各脚本中的注释。按如下顺序执行脚本即完成数据库后端的实现：create_db.sql，build_db.sql，build_db_alter.sql，auto_event.sql，user_procedure.sql，session_procedure.sql，msg_procedure.sql。

A. 数据库创建

数据库各表结构如表 I所示。搭建数据库时，按如下顺序执行附件中的 SQL 脚本：create_db.sql，build_db.sql，build_db_alter.sql。create_db.sql 脚本

表 I
DBChat 数据库结构

表名	列名	主键	外键	其它
CLIENT	CID	Y	N	自增长
	CNAME	N	N	非空
	CPASSWORD	N	N	CLIENT 密码
	CACTIVE	N	N	非空，CLIENT 最近一次操作时间
KNOWS	CID	Y	CLIENT.CID	非空，好友关系主动方
	CLI_CID	Y	CLIENT.CID	非空，好友关系被动方
FRIENDAPPLICATION	FAID	Y	N	自增长
	CID	N	CLIENT.CID	非空，FA 发起方
	CLI_CID	N	CLIENT.CID	非空，FA 被动方
	FAACTIVE	N	N	非空，FA 最近一次被操作时间
SESSION	SID	Y	N	自增长
	SNAME	N	N	非空，FA 发起方
	SPASSWORD	N	N	SESSION 密码
	CID	N	CLIENT.CID	非空，SESSION 管理员
	SVISIBLE	N	N	非空，SESSION 类别，标记是否对外可见
	SACTIVE	N	N	非空，SESSION 最近一次被操作时间
INSESSION	CID	Y	CLIENT.CID	非空，处于关系中的 CLIENT
	SID	Y	SESSION.SID	非空，处于关系中的 SESSION
	SPASSWORD	N	N	非空，记录该的 SESSION 密码
SESSION	MID	Y	N	自增长
	SID	N	SESSION.SID	非空，发送至 SESSION
	CID	N	CLIENT.CID	非空，发送自 CLIENT
	MTEXT	N	N	非空，消息的文本内容
	MBIRTH	N	N	非空，MSG 被发送的时间

创建和初始化 DBChat 数据库。build_db.sql 为 PowerDesigner 工具自动生成的脚本，完成创建数据库中各表的工作。build_db_alter.sql 对自动生成的数据库进行一些调整，完成数据库的创建。

下面简单说明对自动生成的数据库进行了哪些调整: 向 INSESSION 表添加 SPASSWORD 列, 用于保存 SESSION 的密码；将 restrict 的外键改为 cascade，使得删除记录时，与其相关的各行被自动删除；将所有的 BIRTH 和 ACTIVE 列的类型更换为 timestamp，使得可以方便地使用 CURRENT_TIMESTAP 对其进行赋值，并且可以利用 Mysql 提供的 TIMESTAMPDIFF() 函数计算时间差值。

B. 自动清理

完成数据库创建后，执行 auto_event.sql 脚本，创建每日例程清理的事件（EVENT），该事件自创建时刻起，每天（每 24 小时）被执行一次，对长期不活跃的 CLIENT 和 SESSION，长期未被处理的 FA，以及发送时间久远的 MSG 进行清理，删除这些记录。

目前 CLIENT 的不活跃时间定为一个月，SESSION 的不活跃时间定为一周，FA 和 MSG 的清理时间定为一周。若需要更改，则直接更改脚本后重新执行即可。

C. PROCEDURE

为了防止 SQL 注入攻击，以及加快数据库查询速度，我们将所有有可能用到的增删改查操作包装在存储过程（PROCEDURE）之中，WEB 前端对数据库的任何操作都通过 “CALL SOME_PROCEDURE()” 的方式进行。WEB 前端连接数据库所用的 Mysql 用户，仅在该数据库上具有执行存储过程（EXECUTE）的权限，而不具有其它任何权限（该用户无法使用自定的增删改查的功能）。通过存储过程参数化和用户权限的保护，DBChat 在数据库后端对注入攻击进行了防护，减少 WEB 前端的实现难度。

下面对 Mysql 存储过程的定义方式，以及 DBChat 中使用的存储过程的共性进行说明。

Mysql 中的 PROCEDURE 允许使用参数的方式传入或传出变量，此外还可以返回一张查询结果的表。参数的声明方法为 “IN/OUT/INOUT PARAM_NAME PARAM_TYPE”，其中通过 IN，OUT，INOUT 来选择该参数为输入，输出或输入且输出。标准的

PROCEDURE 的定义如下所示，输出参数通过 set 进行赋值，除去输出参数，还可以通过 SELECT 返回一张查询结果表，若需要在一个 PROCEDURE 中返回多张表，则需要创建临时表，将结果的多张表在临时表中拼接，最终返回临时表这一张大表。注意到由于 PROCEDURE 中会有多行，即会出现多次 “;”，需要更换结束符号，当 PROCEDURE 创建完成后，再将结束符号换回 “;”。

```
drop procedure if exists p;
delimiter $$
create procedure p(IN p_in int, OUT p_out int)
    begin
        set p_out=1;
        select * from SOME_TABLE;
    end $$
delimiter ;
```

使用 “CALL p(SOME_INT, @tmp_var);” 命令调用如上的 PROCEDURE。调用后会立刻返回 “select * from SOME_TABLE;” 的查询结果表，再使用 “SELECT @tmp_var” 命令获得输出参数 p_out 的值。

在 DBChat 中，所有的 PROCEDURE 中都会使用事务（TRANSACTION），并且设定为一旦出现任何异常，立即回滚。此外，DBChat 中几乎所有 PROCEDURE 都需要验证身份（即验证 CID 和 CPASSWORD），仅有合法的 CLIENT 可以进行操作，因此几乎所有 PROCEDURE（除去注册）都会有 user_id 和 user_pswd 两个输入参数和 granted 输出参数，若身份验证以及后续的验证通过，则 granted 非 0，否则验证不通过，granted 为 0，若 PROCEDURE 执行过程中出错，则 TRANSACTION 回滚，granted 为空值，通过检查输出参数 granted 的值即可确定该 PROCEDURE 执行状态。DBChat 中所用的 PROCEDURE 的基本形式如下。

```
drop procedure if exists p;
delimiter $$
create procedure p(IN user_cid int,
                   IN user_pswd char(32),
                   OUT granted int)
    begin
        declare EXIT HANDLER for SQLEXCEPTION rollback;
        start transaction;
        set granted=(...); # 身份验证
        if granted>0 then
            # Do something here ...
        end if;
        commit;
    end $$
delimiter ;
```

D. CLIENT PROCEDURE

执行 user_procedure.sql 脚本创建所有 CLIENT 需要用到的 PROCEDURE，包括注册，登录，注销，个人信息获取和更改，好友相关操作等。下面对各个 PROCEDURE 进行简单的介绍，详细说明请见附件 sql_script.sql/README.md。

USER_SIGN_UP 注册新用户，输入参数为新用户的 CNAME 和 CPASSWORD，输出参数为为该用户注册的 CID。注意，新用户注册时会默认和自己是好友，并且该好友关系禁止解除。

USER_SIGN_IN 验证用户身份，输入参数为 CID 和 CPASSWORD，输出参数为 granted，即是否通过验证。由于 DBChat 数据库中并不记录登录状态，因此该 PROCEDURE 仅仅进行身份验证，登录状态记录于前端，若通过了验证，则前端自动保存 CID 和 CPASSWORD 用于之后所有操作的验证，而不需要用户每一次操作都输入 CID 和 CPASSWORD。此外，由于上述原因，DBChat 不需要退出登录的 PROCEDURE，网页一旦关闭，则前端保存的 CID 和 CPASSWORD 自动删除，相当于用户退出登录。

USER_SIGN_OUT 注销用户，将其永久删除，由于外键被设置为 cascade，与之相关的所有 INSESSION，KNOWS，FRIENDAPPLICATION，MSG 表项全部被删除。

USER_GET_NAME 和 USER_RENAME 获取用户名字和对用户重命名。由于在登录时使用 CID 和 CPASSWORD 进行验证，并没有 CNAME 信息，因此前端必须使用该 PROCEDURE 来获取登录了的 CLIENT 的名字。

USER_SEARCH 查询 CLIENT 的 CNAME，获得所有叫该名字的 CLIENT 的 CID，该 PROCEDURE 中使用第III-C节中提及的 SELECT 返回结果表的方法。由于 CLIENT 对其它 CLIENT 的好友申请等操作都需要对方的 CID，而名字对于人而言相对直观，因此提供该 PROCEDURE 来进行查询。

USER_CHANGE_PASSWORD 更改用户密码。

USER_GET_FRIEND 获取用户的好友的 CID 和名字 CNAME。

USER_FRIEND_APPLICATION 向某个其他用户发起好友申请，该 PROCEDURE 中除了身份验证，还需要进行是否已发送好友申请和是否已经是好友的验证。

USER_GET_FRIEND_APPLICATION 获得所有向该用户发送的好友申请。

USER_ACCEPT_FRIEND_APPLICATION 和 USER_REFUSE_FRIEND_APPLICATION 分别通过 FAID 来接受或拒绝好友申请。注意好友关系是对称的，因此接受好友后，KNOWS 表中需要插入两行，分别表示 “A 是 B 的好友” 和 “B 是 A 的好友”。

USER_DELETE_FRIEND 解除同某用户的好友关系，注意禁止解除同自己的好友关系。

E. SESSION PROCEDURE

执行 session_procedure.sql 脚本创建所有 SESSION 相关的 PROCEDURE，包括创建，退出，解散，加入等。下面对各个 PROCEDURE 进行简单的介绍，详细说明请见附件 sql_script.sql/README.md。

首先再次强调两种不同的 SESSION，一种为 “私聊”，SVISIBLE 设置为 FALSE，私聊允许两个且仅两个用户在其中，不在私聊中的用户无法感知其存在，也无法影响其（即不可能查找也不可能加入私聊），私聊内的用户只可以解散 SESSION，而不能退出；另一种为 “群聊”，SVISIBLE 设置为 TRUE，群聊允许任意数量的用户在其中，但不可以为空，群聊可以被外界感知，也允许加入或退出。

此外，需要说明管理员的概念和特权。SESSION 的管理员不可以退出 SESSION，但具有解散 SESSION 的特权，若管理员希望退出，则必须转交管理员给 SESSION 中的其他某个用户，将自己降级为普通用户。私聊之中不存在管理员，或者说私聊中的两个用户都是管理员；而群聊之中 SESSION.CID 对应的用户即为管理员。

SESSION_CREATE_FRIEND 和 SESSION_CREATE 分别创建 “私聊” 和 “群聊”。私聊创建时需要提供好友 CID，不需要提供 SNAME 和 SPASSWORD，由于私聊对于外界不可见，因而没有必要使用密码，私聊默认 SNAME 为 “A and B”，其中 A 和 B 为两个用户的 CNAME。群聊创建时，需要提供 SNAME 和 SPASSWORD，其中 SPASSWORD 用于不在该 SESSION 中的用户在未被邀请的情况下主动加入 SESSION 的身份验证。

SESSION_SEARCH 通过 SNAME 查找群聊的 SID，注意该方法搜索不到私聊。

SESSION_JOIN 和 SESSION_INVITE 为新用户加入群聊提供两种方式。JOIN 允许新用户用 SID 和 SPASSWORD 的方式验证加入 SESSION，一旦加入 SESSION 后，SPASSWORD 会存入 INSESSION 表中的相关的行，此后便不需要密码。INVITE 则允许已在 SESSION 中的用户邀请自己的好友加入。

SESSION_GET_OTHERS_INSESSION 获取该 SESSION 中所有用户的信息，不限私聊或群聊。

SESSION_GET_MY_SESSION 获取该用户所在的所有 SESSION 的信息，不限私聊或群聊。

SESSION_CHANGE_MANAGER 更换管理员。该方法仅在群聊有效，私聊中使用该方法尽管可以改变 SESSION.CID，但并没有任何效果（因为私聊中双方都是管理员，而不由 SESSION.CID 指定）。

SESSION_DESTROY 解散群聊，为管理员特权，仅群聊管理院和私聊用户可以使用。

SESSION_LEAVE 退出群聊，仅群聊普通用户可以使用，群聊管理员和私聊用户禁止使用。

F. MSG PROCEDURE

执行 session_procedure.sql 脚本创建所有 MSG 相关的 PROCEDURE，包括发送，获取和撤回。下面对各个 PROCEDURE 进行简单的介绍，详细说明请见附件 sql_script.sql/README.md。

MSG_SEND 发送一条消息至某个 SESSION 中。

MSG_GET 获取该用户所在的该群聊中所有消息。

MSG_DELETE 使用 MID 删除该用户此前发送的某条消息。

IV. WEB 前端

网页使用 Bootstrap 框架编写，详情请见附件中 front_end 目录。

前端中会对用户密码（CPASSWORD）的明文进行加密，加密后发送至数据库，数据库中使用密码密文进行比对；SESSION 密码不进行加密，因为其真正起作用仅仅是在 SESSION_JOIN 之中，并且 SPASSWORD 被认为不具有特殊含义。

一旦登录成功，以及进入某个 SESSION 后，CID，密文 CPASSWORD 等信息就会保存在 php 的 $_SESSION 变量之中，并且可以在其他页面获取到。

前端网页具有 3 个关键的页面，分别为：首页（index.php），用户主页（main.php），聊天页（session_main.php）。下面围绕这三个页面及其上相关动作进行简单的介绍。

A. 首页

首页（index.php）为用户在浏览器输入网址后直接进入的页面，该页面上提供注册（SIGN UP），登录（SIGN IN）和跳转至 About us 三个操作。

aboutus.html 为已经制作好的静态网页，点击 About us 按钮后会链接跳转至该页面，aboutus.html 上 Home 按钮链接至首页，点击后跳转回首页。

SIGN UP 按钮，要求输入用户名和密码，之后执行 signup.php 脚本将名字和密文密码通过 USER_SIGN_UP 发送至数据库进行注册，返回用户的 CID 并弹出 alert 窗口进行通知。

SIGN IN 按钮，要求输入 ID 和明文密码，之后执行 signin.php 脚本进行身份验证。

SIGN UP 和 SIGN IN，在成功后都会将 CID 和密文 CPASSWORD 保存在 PH 的 $_SESSION 变量中，并跳转至用户主页。

B. 用户主页

用户主页（main.php）为用户登录之后进入的页面，该页面左侧浮动栏中显示该用户所在的 SESSION，点击便进入相应的聊天页。浮动栏最下两个按钮分别为创建群聊（Create group session）和创建私聊（Create private session），点击后进入分别创建群聊和私聊的页面（session_create.php），创建群聊需要输入群聊的名字和密码，创建私聊需要输入好友的 CID，确认输入后执行 session_create_action.php 文件，向数据库发送请求完成创建，这一过程中若出现错误则弹出警告窗口报错。

顶栏有个人信息（用户名按钮），“我的”（My），好友申请（Friend），加入群聊（Join）和搜索（Search）下拉菜单。

个人信息菜单下显示当前登录用户的 ID 和名字，提供改名（Rename），改密码（Password），退出（Leave），删除账号（Purge）按钮。点击 Rename 按钮，跳转至更名界面（user_rename.php），要求输入新的名字，确认输入后执行 user.php 文件中的 Rename 相关部分，向数据库发送更名请求；点击 Password 按钮，跳转至更换密码界面（user_password.php），要求输入旧密码和新密码，之后脚本使用加密算法将新旧密码都转为密文，比对旧密码是否正确，若正确则向数据库发出更换密码的请求，成功后更新 $_SESSION 变量中存储的密文密码；点击 Leave 按钮，直接退回首页并清空 $_SESSION 变量；点击 Purge 按钮，向数据库发送注销账户的请求，注销成功返回首页，且该用户不再存在。

My 菜单下有我的好友（Friend），我的聊天（Group）和我的好友申请（Appliacation）三个查询按钮。点击后跳转至查询结果页面（my.php），该页面中使用数据库的相应的 PROCEDURE，并根据查询的内容以表的形式展示查询结果。

Friend 菜单下允许申请（Friend application）或删除（Delete friend）好友，同时允许通过（Accept application）或拒绝（Delete application）别的用户发出的好友申请。以上所有操作均需要输入 ID（用户 CID 或 FA 的 FAID），完成输入并点击按钮，执行 friend.php，向数据库发起对应请求。

Join 使得用户可以通过 SID 和 SPASSWORD 验证的方式主动加入聊天，输入 SID 和 SPASSWORD 后，点击 Join 按钮，向数据库发起请求，验证通过则加入群聊。

Search 下有 User 和 Group 两个按钮，分别允许精确匹配搜索用户或群聊名称，获得用户的 CID 和 CNAME，或群聊的 SID，SNAME，管理员 CID 和 CNAME，显示在查询结果页面（search.php）中。

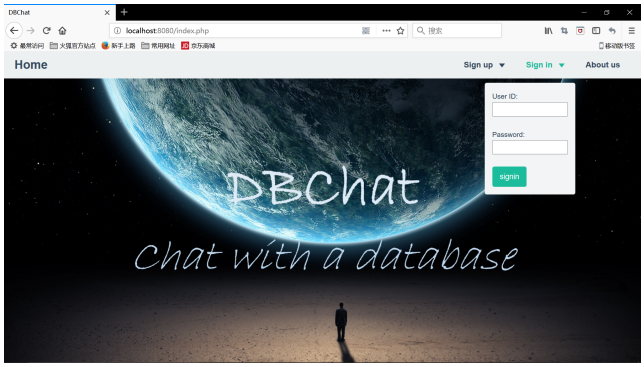
注意上述方法，成功或失败都会自动跳转至对应的页面并刷新，因此在用户主页上不需要自动刷新机制。

C. 聊天页

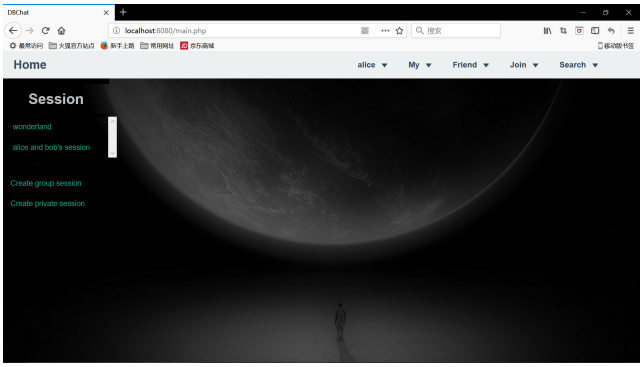
聊天页（session_main.php）为用户主页点击左侧聊天名称后进入的页面，该页面为聊天（群聊或私聊）的主页面。在首次加载如聊天页时，通过 POST 跳转，此时可以在 PHP 的 $_POST 变量中获取该 Session 的 SID 和 SNAME 信息，之后将其存储在 $_SESSION 变量中以便之后刷新页面使用。

页面左侧浮动栏显示该聊天的名字和参与聊天的各个用户的名字和 CID（CID 为名字后方括号中的内容），红色表示管理员用户，灰色表示普通用户。右侧为聊天界面，按时间顺序显示该聊天室内的各个消息的发送人，发送时间和消息内容等。顶栏中有个人信息（用户名按钮），消息撤回（Message），邀请好友（Invite），和管理员与退出（Session）下拉菜单。

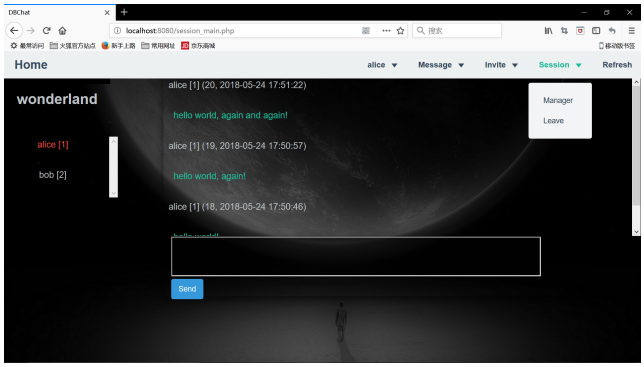
点击个人信息菜单，同个人主页相同显示该用户的 CID 和 CNAME，但不提供别的方法，若需使用那些方法，则必须返回个人主页。



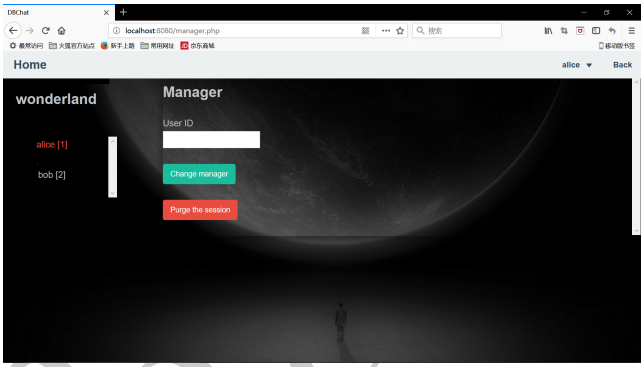
(a) 首页



(b) 用户主页



(c) 聊天页



(d) 管理员页面

图 2. DBChat 项目展示，使用 Chrome 浏览器访问部署于云服务器端的 Web 服务器。图 2(a)为浏览器访问 DBChat 主页，并点击登录按钮截图。图 2(b)为 alice 用户登陆后的用户主页截图，可见 alice 处于群聊 wonderland 中和和用户 bob 的私聊中。图 2(c)为 alice 在用户主页点击 wonderland 后进入 wonderland 群聊的聊天页，并点击 Session 菜单的截图，可以看到 alice 是管理员，此外还有一个普通用户 bob，聊天框中 alice 发送了若干 “Hello world” 消息。图 2(d)为 alice 点击 Manager 按钮后进入的管理员页面截图，该页面中允许更换管理员（输入 User ID 并点击 Change Manager 按钮），或直接解散群聊（点击 Purge the session 按钮）。

点击 Message 按钮，输入希望撤回的消息的 MID，执行 message.php 中的 Withdraw 相关部分，若通过验证则撤回该条消息。

点击 Invite 按钮，输入希望邀请的好友的 CID，执行 message.php 中的 Invite 相关部分，若通过验证则将好友邀请进入群聊中。

Session 菜单下有管理员界面（Manager）和退出（Leave）两个按钮。管理员只能点击 Manager 按钮进入管理员界面（manager.php），普通用户只能点击 Leave 按钮退出群聊。若管理员点击 Leave 或普通用户点击 Manager 按钮，则弹出 alert 报错。在管理员界面中有更换管理员（Change manager）和解散群聊（Purge）两个按钮，点击后分别执行 session_leave.php 中对应的部分，更换管理员或解散群聊。

在消息输入框中输入希望发送的消息，之后点击 Send 按钮发送，执行 message.php 中的 Send 相关的部分，将消息发送至数据库，之后页面刷新，最新发送的消息被加载在页面中。

由于并没有使用局部自动刷新机制，因此别的用户发送的消息，在不刷新页面的情况下，无法看到。

V. 系统部署和展示

DBChat 的数据库后端和 Web 服务前端，均部署在阿里云服务器端，分别监听 3306 和 80 端口，服务器 IP 地址为 47.94.138.231。Web 服务器登录数据库的用户为 user_dbchat，密码为空，该用户仅在 DBChat 数据库上有 EXECUTE 权限，无法进行其他任何操作。

图IV-C通过用户 alice 展示了 DBChat 的基本界面。alice 首先于主页登录（图 2(a)），进入用户主页（图 2(b)），之后进入群聊 wonderland 并发送若干消息（图 2(c)），最终进入管理员页面（图 2(d)）。