数据库实施阶段总结

一．建立数据库

create database MBG

create table figure

(Cno varchar(8) primary key,

charac varchar(20),

hair int,

attack int,

defend int,

lv int

)

create table player

(Cno varchar(8) primary key,

charac varchar(20),

hair int,

attack int,

defend int,

lv int

)

create table cha\_type

(Cno varchar(8) primary key,

ctype char(2) check (ctype in ('攻','防','智'))

)

create table type\_relation

(res char(2) check (res in ('攻','防','智')),

resed char(2)check (resed in ('攻','防','智')),

primary key (res,resed)

)

create table skill

(Sno varchar(8) primary key,

skillname varchar(20),

attack int,

defend int,

process char(2) check (process in ('+','\*')),

obj varchar(20) check (obj in('card','character'))

)

create table fs\_match

(Cno varchar(8),

Sno varchar(8),

primary key (Cno,Sno)

)

create table f\_card

(cardno varchar(20) primary key,

cardname varchar(20),

hair int,

attack int,

defend int

)

create table p\_card

(cardno varchar(20) primary key,

cardname varchar(20),

hair int,

attack int,

defend int

)

二．数据库完整性控制：

1.实体完整性：

每一个表中都设置了一个专门用来标志每一条记录的序号，并将其设置为了主键，确保了数据库的实体完整性，如figure、player、cha\_type表都以Cno为主键，skill表以Sno为主键，f\_card、p\_card表都以cardno为主键，fs\_match、type——relation表以各自表格的两个元素的组合为主键。

2.参照完整性：

为了便于使用数据框格式从数据库中提取数据，并且拆分表格改进数据库的范式，此数据库中没有定义参照完整性约束。

3.用户定义的完整性：

设置了属性上的约束条件。

在表格cha\_type和type\_relation中，对人物的类型做的限制，即人物只能拥有“攻”、“防”、“智”这三种类型，不允许使用除此之外的类型对表进行插入或修改操作。

在表skill中，对process和obj进行了限制。列process为对应技能的对数值的操作，限制为“+”或“\*”，即技能只能对数据进行加法或者乘法的操作。列obj表示技能作用的对象，限制为“card”或“character”，即技能作用的对象只能是一张卡牌或者一个游戏角色。

4.断言：

create assertion ass\_on\_figure check (100>=(select COUNT(\*)from figure))

create assertion ass\_on\_player check (100>=(select COUNT(\*)from player))

create assertion ass\_on\_type\_relation check (3>=(select COUNT(\*)from type\_relation))

对数据库附加了三条断言，前两条是为了避免使得游戏机制过于复杂，对人物玩家的数量进行了限制，即figure、player表的记录条数都不允许超过100。否则会有太多角色需要用户来记忆，降低游戏的可玩性。

第三条是对type\_relation表进行的限制，由于“攻”、“防”、“智”这三种类型人物相克关系是确定的，所以type\_relation表是不需要修改的表格，将这张表格的记录条数限制为原本就规定好的3条。

5．触发器

create trigger insert\_or\_update\_cha

before insert or update on figure

referencing new row as newrecord

for each row

begin

if (newrecord.lv<1)or(newrecord.lv>6)

then newrecord.lv=1

end if

end;

create trigger insert\_or\_update\_player

before insert or update on player

referencing new row as newrecord

for each row

begin

if (newrecord.lv<1)or(newrecord.lv>6)

then newrecord.lv=1

end if

end;

create trigger insert\_or\_update\_cha\_balance

before insert or update on figure

referencing new row as newrecord

for each row

begin

if (newrecord.hair>30)or(newrecord.attack>30)or(newrecord.defend>30)

then newrecord.charac='unbalanced'

end if

end;

create trigger insert\_or\_update\_player\_balance

before insert or update on player

referencing new row as newrecord

for each row

begin

if (newrecord.hair>30)or(newrecord.attack>30)or(newrecord.defend>30)

then newrecord.charac='unbalanced'

end if

end;

对数据库增加四条触发器，对figure、player表进行了限制。前两条为了维护游戏规则，根据规则，一共进行六轮战斗即能通关，所以人物和玩家等级不允许超过6或者小于1，在错误设置的情况下，将其修改为一个默认等级1级。后两条为了维护游戏平衡，对角色的发量值、攻击值和防御值做出限制，因为某项过高的数值会导致玩家过于强大或者角色太难击败。一旦修改或插入操作中某一项的值超过了30，即会将该条记录的名称先命名为“unbalanced”，由后续开发人员进行讨论后再进一步考虑如何处理这条打破游戏平衡的记录。

三．视图的建立

create view FC\_match1

as

select \*

from figure

create view FC\_match2

as

select \*

from player

create view FPcard\_match1

as

select \*

from f\_card

create view FPcard\_match2

as

select \*

from p\_card

create view viewskill

as

select \*

from skill

create view viewcha\_type

as

select \*

from cha\_type

create view viewfs

as

select \*

from fs\_match

以上视图的建立是为了配合前端的代码工作，生成的视图中仅仅提供了前端代码所需要的数据表，一方面便于前端开发者直接调用，简化操作，另一方面在之后的数据库维护和进一步开发中，可以将不同视图的权限分给不同层次的管理者或用户，保证其他数据的机密和安全，一定程度上确保最小访问权原则。

四．数据库安全性

grant all privileges

on table cha\_type,f\_card,figure,fs\_match,p\_card,player,skill,type\_relation

to manager1,manager2,manager3,manager4

with grant option;

grant all privileges

on table figure,cha\_type,player

to cha\_manager;

grant all privileges

on table f\_card,p\_card

to card\_manager;

grant all privileges

on table skill,fs\_match

to skill\_manager;

grant select

on table cha\_type,f\_card,figure,fs\_match,p\_card,player,skill,type\_relation

to gameplayer

主要通过对用户的授权来实现数据库的安全性控制。

在目前的情况下，将数据库的管理情况分为三级，即开发人员——管理者——游戏玩家。开发人员有四位，所以将所有表的所有权限给了manager1到manager4，并允许开发人员进行授权操作。管理者分为卡牌管理者、角色管理者、技能管理者，卡牌管理者负责f\_card、p\_card表，角色管理者负责figure、cha\_type、player表，技能管理者负责skill、fs\_match表，三种管理者在协商一致的情况下分别对各自的负责的表格进行操作，由于需要管理者对数据库进行更新或修改，所以授予了全部权限。对于游戏玩家，为了使其了解游戏的机制、玩法和内容，所以允许查看所有表格的数据。