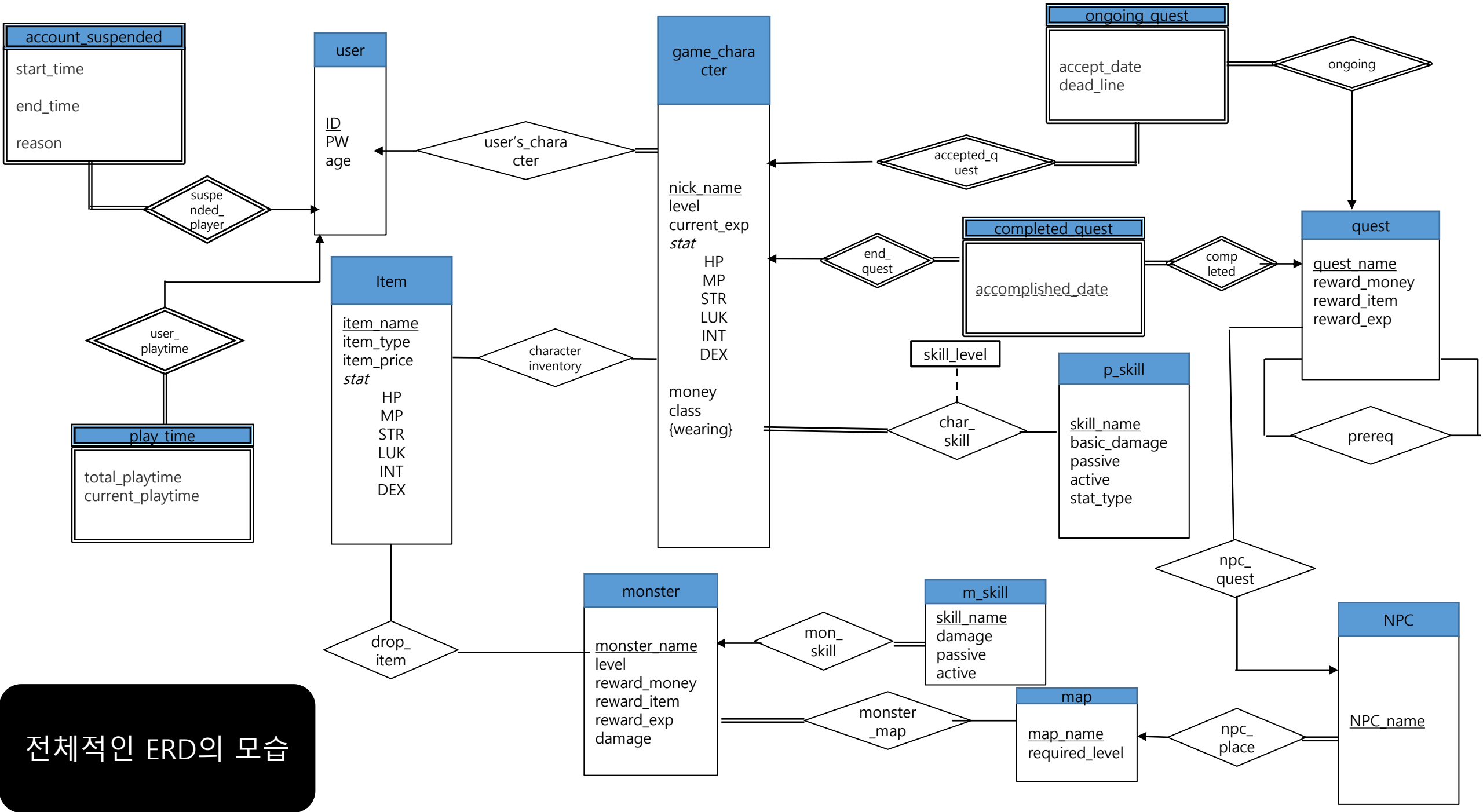
DB 레포트 VI

20182345 이건호

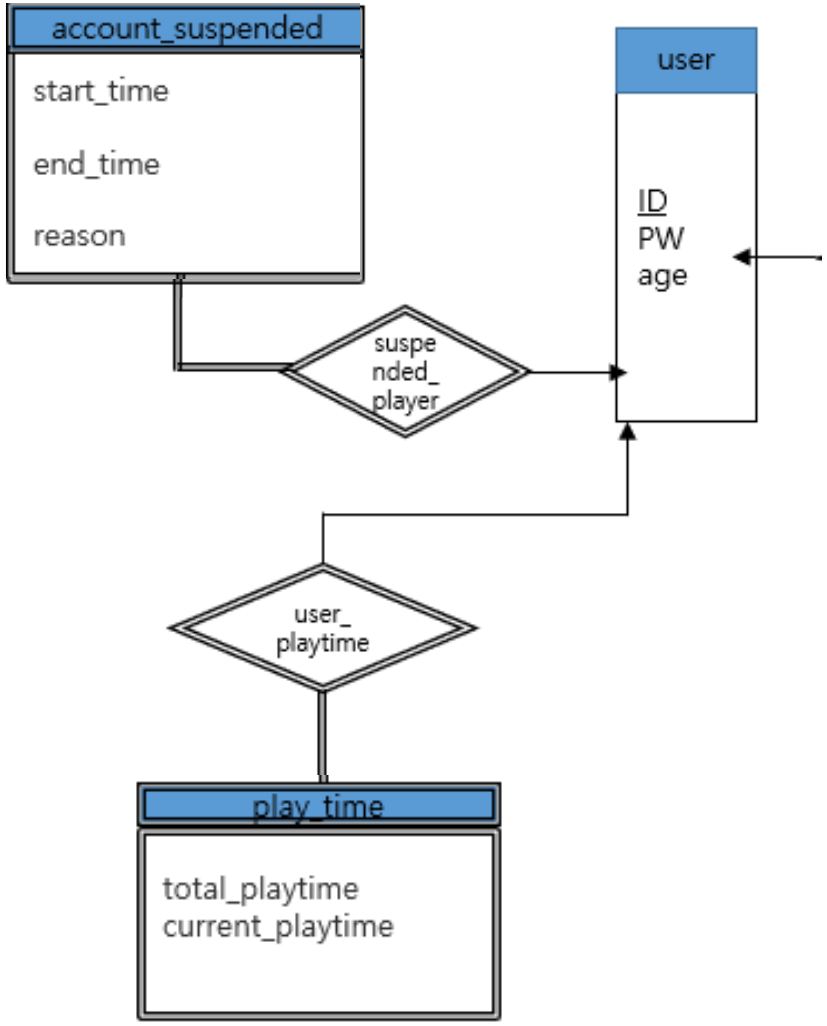
(1)응용분야 제목: RPG 게임 DB

- 선정 이유: RPG게임류를 많이 접해봤기에 해당 DB에 필요할 것 같은 대략적인 요소들을 유추할 수 있을 것 같고, 추후 게임을 개발할 생각이 있기에 연습삼아 전공 설계과제를 통해 원하는 DB를 직접 설계해보고자 하기 때문

(2)ERD 및 작성에 대한 설명


account_suspended
start_time
end_time
reason
suspe
nded_
player
user
ID
PW
age
user's_chara
cter
game_chara
cter
nick_name
level
current_exp
stat
HP
MP
STR
LUK
INT
DEX
money
class
{wearing}
ongoing_quest
accept_date
dead_line
ongoing
accepted_q
uest
completed_quest
accomplished_date
end_
quest
comp
leted
quest
quest_name
reward_money
reward_item
reward_exp
prereq
user_
playtime
play_time
total_playtime
current_playtime
Item
item_name
item_type
item_price
stat
HP
MP
STR
LUK
INT
DEX
character
inventory
skill_level
char_
skill
p_skill
skill_name
basic_damage
passive
active
stat_type
npc_
quest
drop_
item
monster
monster_name
level
reward_money
reward_item
reward_exp
damage
mon_
skill
m_skill
skill_name
damage
passive
active
NPC
NPC_name
monster
_map
map
map_name
required_level
npc_
place

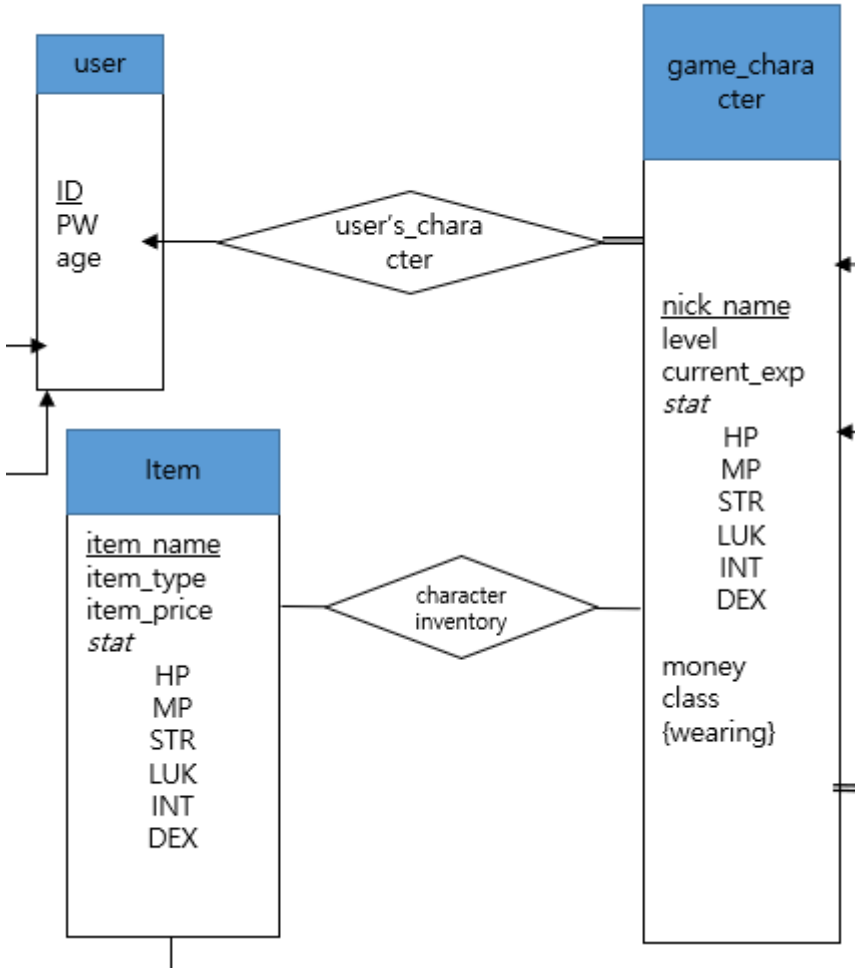
전체적인 ERD의 모습



user 엔티티 셋은 ID와 PW와 age정보를 갖는다. 이는 추후 로그인시 ID와 PW를 확인할때 사용된다.

account_suspended 엔티티 셋은 게임에서 버그나 부정행위로 제재를 당한 현황을 담는 weak entity set이다. user엔티티셋의 PK인 ID와 함께여야 identifyin이 가능해지기 때문에 weak entit로 설정하였다.
start_time,end_time은 정지를 당한 날짜와 끝날 날짜이다. reaso은 정지 사유가 적혀있다. end_time에 해당하는 날짜에 로그인을 하면 체크를 하여 나중에 이 목록에서 해당사용자 튜플을 지워줌으로서 정지현황을 관리할 예정이다.

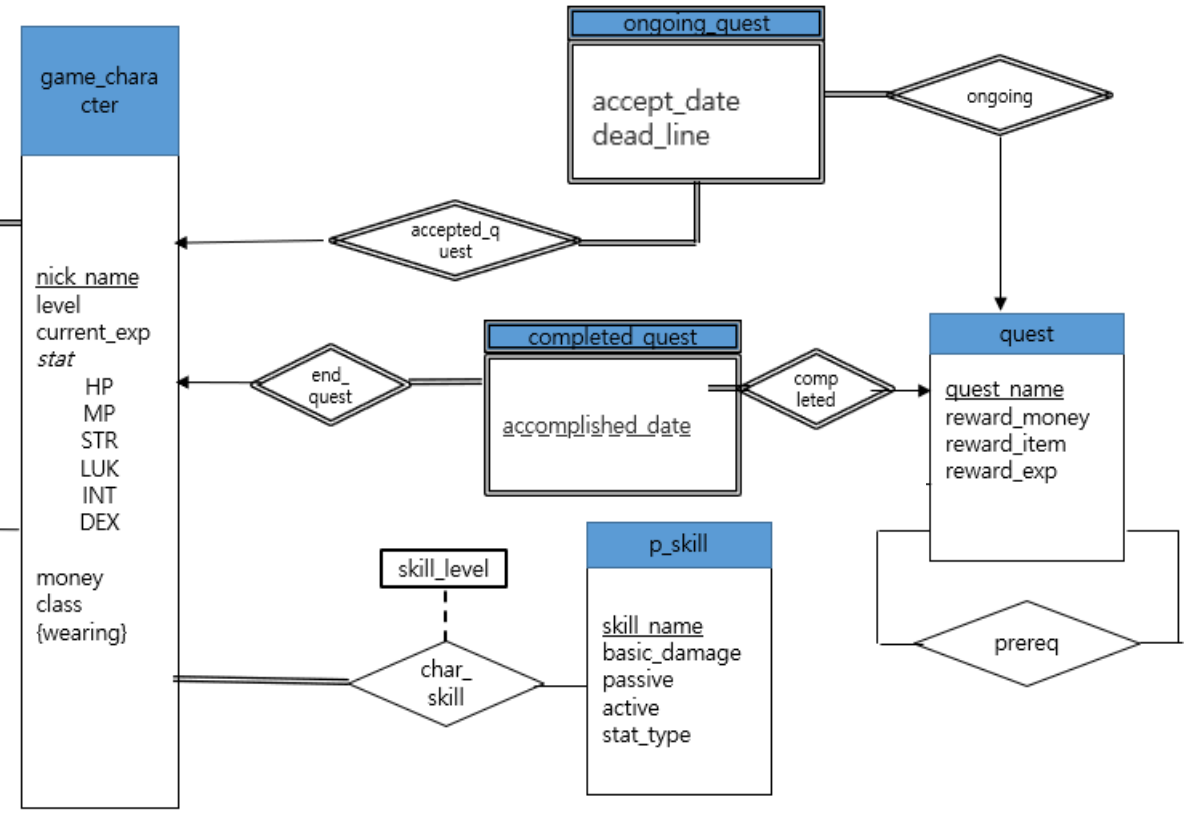
play_time 엔티티 셋은 이용자의 총 게임이용시간과 현재 접속시간(게임을 켜서 끄기전까지의 시간)이 저장된다. 현재 접속시간을 응용에서 계산할수도 있는데 굳이 ERD에 추가한 이유는 혹시라도 플레이어가 게임에서 팅겼을때 빠른시간 내에 접속하면 기록해둔 current_playtime을 불러와서 이어서 할 수 있게 하고싶기 때문이다. 나중에 이를 이용하여 현재 접속시간을 측정하여 그에 따른 효과를 부여(스탯을 올려준다던지 몬스터에게 나오는 돈을 많게 한다던지...) 하고, 총 누적시간을 측정하여 특정 시간 달성시 인벤토리에 아이템을 추가해주는 식으로 응용할 예정이다.
사실 user 엔티티에 그대로 추가해줘도 괜찮겠지만 플레이시간은 따로 기록하는게 좋을것 같아서 이렇게 표현했다.



user 엔티티 셋은 앞전 슬라이드에서 설명했기 때문에 넘기겠다.

game_character 엔티티 셋은 user엔티티 셋과 어떻게 다르냐면 user엔티티 셋은 게임을 이용하는 사용자 그 사람 자체를 표현하지만 이것은 그 사람이 생성한 캐릭터를 표현해준다. 즉 한 사람이 여러 개의 캐릭터를 가질수도 있다. 구성요소로 닉네임,레벨,현재 경험치량(current_exp),각종 스탯요소, 보유 머니,직업(class), 마지막으로 multivalued attributes로 wearing을 갖고있다. wearing은 현재 캐릭터가 장착하고 있는 아이템들을 나타내는 것이다.

item 엔티티 셋은 게임에서 나타나는 아이템들을 표현하기 위한 것이다. 아이템의 이름과, 아이템타입(소비아이템,장비아이템,기타 아이템 ...등등),아이템 가격,아이템이 지닌 스탯 을 속성으로 지니고있다. game_character엔티티와는 해당 캐릭터의 인벤토리를 나타내기 위해 릴레이션관계를 이루고있다.

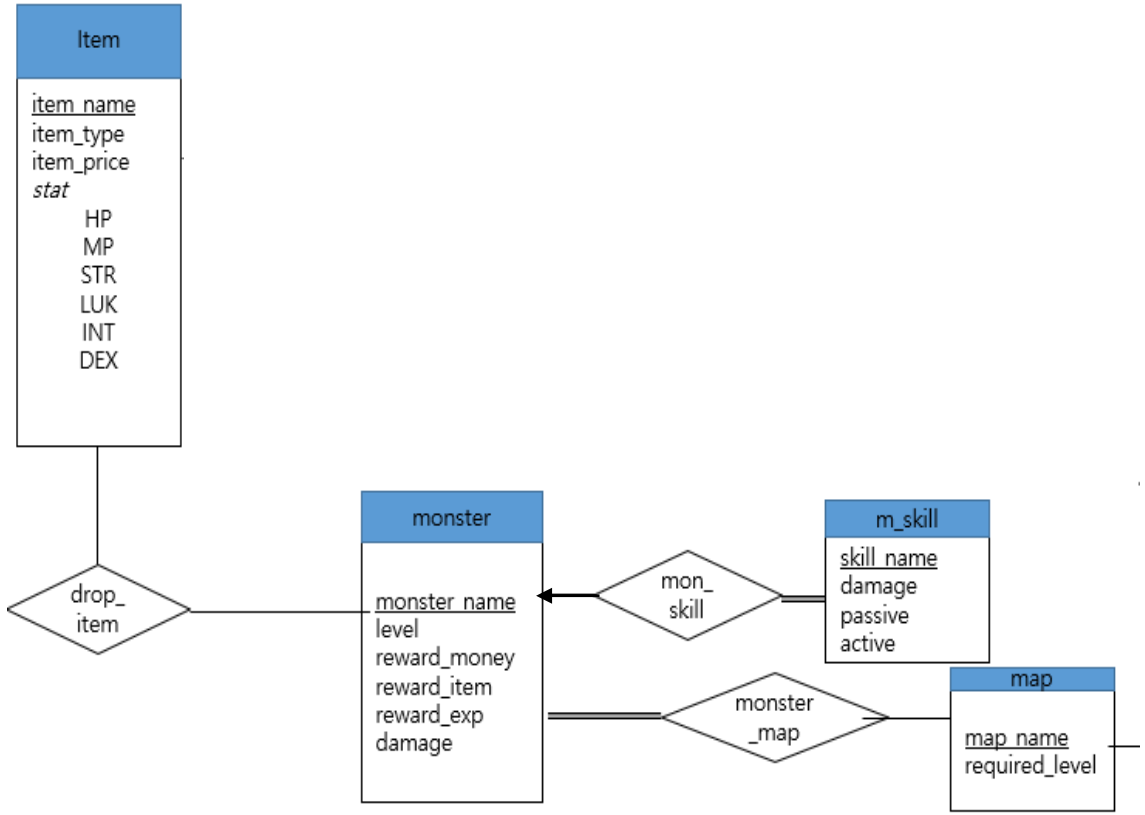


game_character 엔티티는 앞에서 설명했기에 생략.

p_skill 엔티티 셋은 캐릭터가 사용할 수 있는 스킬들을 표현하기 위한 엔티티 셋이다. p_를 붙인 이유는 후에 monster엔티티도 등장하는데 이때 m_skill(몬스터 스킬)과 구분해주기 위해서이다. 스킬 이름,기본 데미지,패시브여부,액티브여부,사용하는 스탯을 구성요소로 포함한다.
그리고 relation으로 char_skill에는 skill_level 이라는 속성이 있는데 이는 추후에 캐릭터가 지닌 스킬과,스킬 레벨,사용 스탯,그리고 기본 데미지를 이용하여 해당 캐릭터가 스킬을 썼을때 발동되는 대미지를 응용에서 다르게 만들어주기 위해서이다.

quest엔티티는 게임에서 구현될수있는 퀘스트들을 표현한 것인데 퀘스트 이름,보상 머니,보상 아이템,보상 경험치를 속성으로 갖는다. 이때 prereq 라는 unary relationship을 갖는데 이는 특정 퀘스트의 경우 선행 퀘스트가 요구되기때문에 표현해 준 것이다.

ongoing_quest엔티티 셋과 **completed_quest**엔티티 셋은 둘다 weak entity로 각각 캐릭터가 현재 진행중인 퀘스트와 이미 완료한 퀘스트를 나타낸다. weak하기 때문에 둘다 game_characte와 quest엔티티에 의존한다. ongoing_quest는 퀘스트를 수락한 날짜와 그 퀘스트의 데드라인을 속성으로 지니고 completed_quest는 퀘스트를 성공한 날짜를 속성으로 지닌다. 이때 ongoing_quest엔티티의 partial key를 지정하지 않은것은 game_character엔티티와 quest엔티티로부터 가져온 pk의 구성만으로 합성키를 이룰 수 있기 때문이다.(진행중인 퀘스트를 중복해서 진행 불가능) 반면에 completed_quest엔티티는 partial key를 갖는데 이유는 동일한 퀘스트를 추후에 중복해서 완료할수도 있기 때문이다.

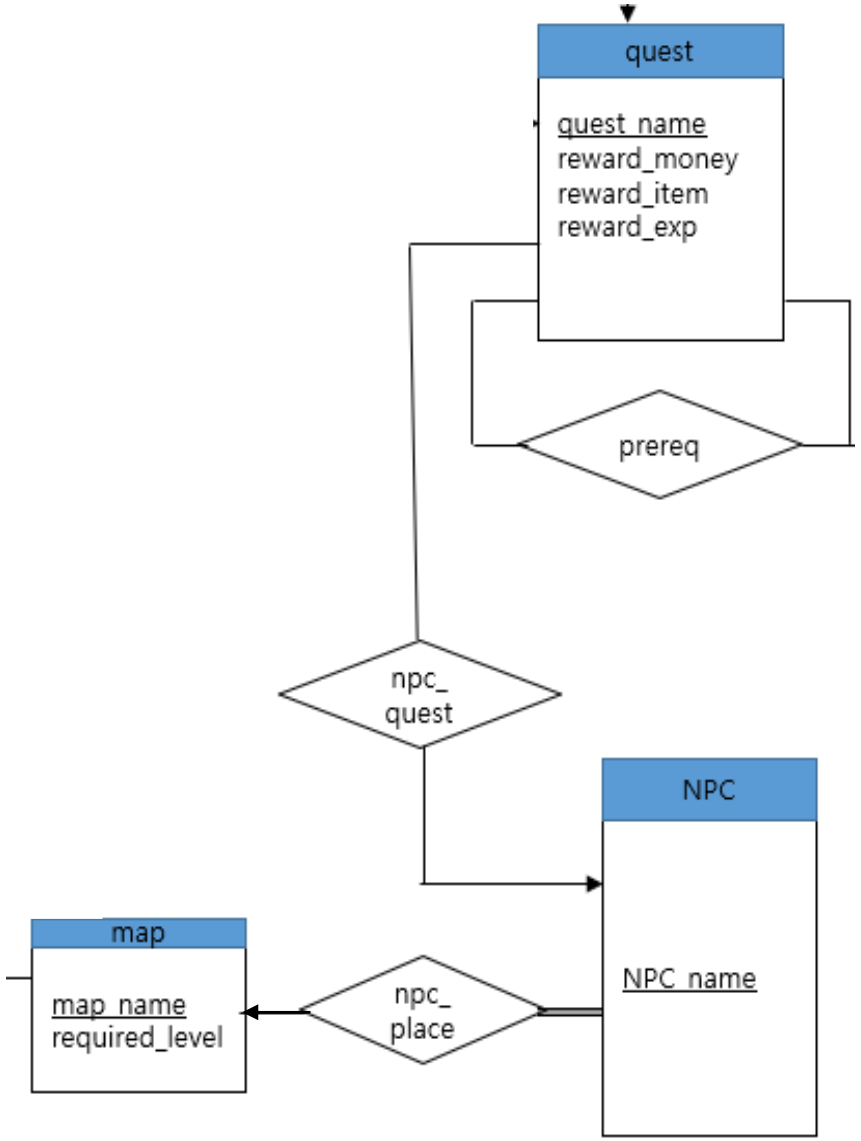


Item 엔티티는 앞에 설명했기에 생략.

monster 엔티티 셋은 게임에 등장하는 적들을 표현한 것이다. 이름,레벨, 잡았을 시 주는 돈,아이템,경험치, 그리고 몬스터가 가하는 기본 대미지가 포함된다.

m_skill은 몬스터의 스킬을 표현한 것이다. 하나의 몬스터가 여러 개의 스킬을 쓸 수도 있기때문에 별도의 엔티티로 표현하였다. 스킬 이름,대미지,패시브,액티브 여부가 표시된다.

map 엔티티는 말 그대로 게임의 맵 정보를 담고있다. 간단하게 맵 이름과 입장하기 위한 최소 레벨을 속성으로 갖고 있다.

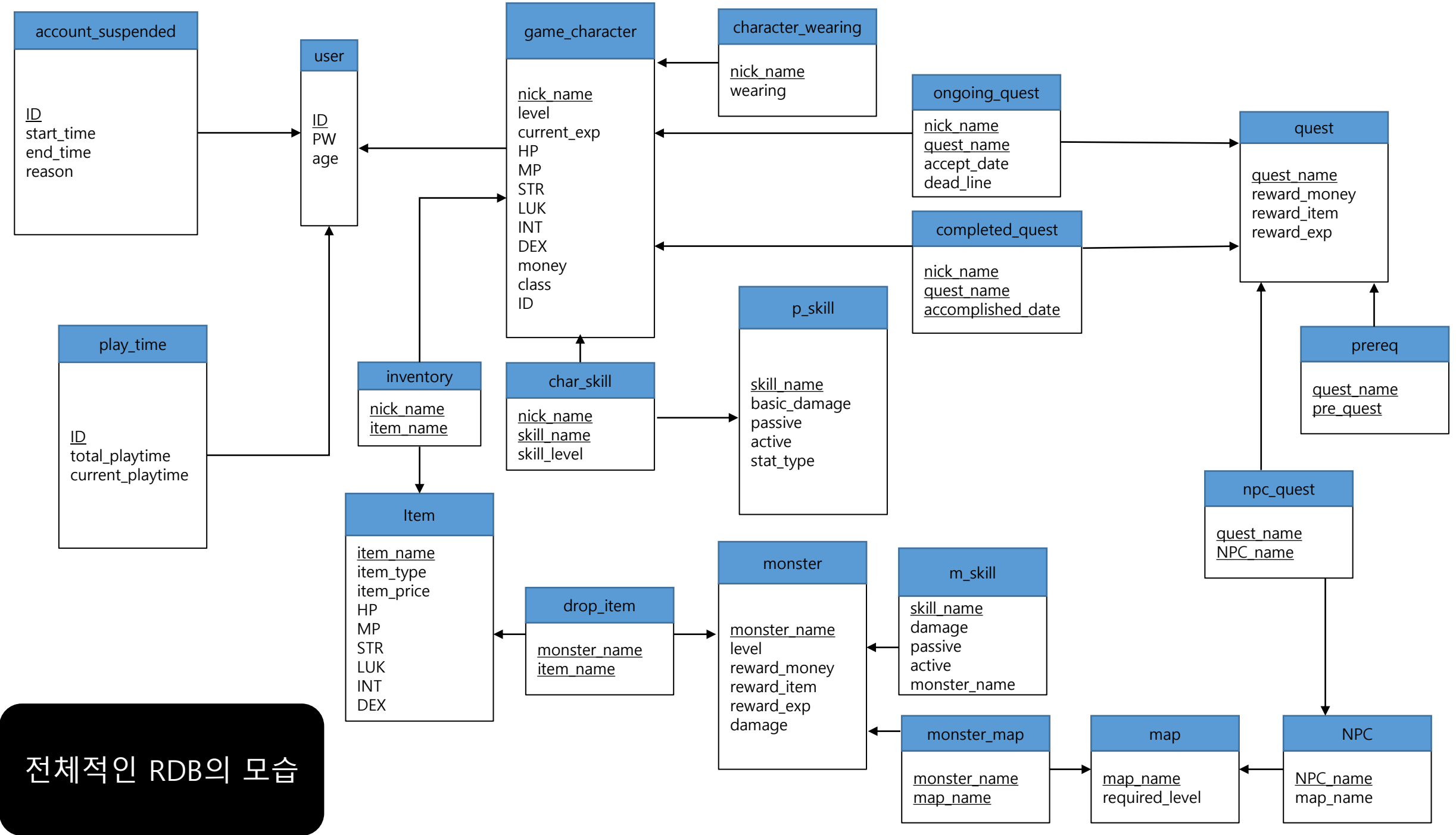


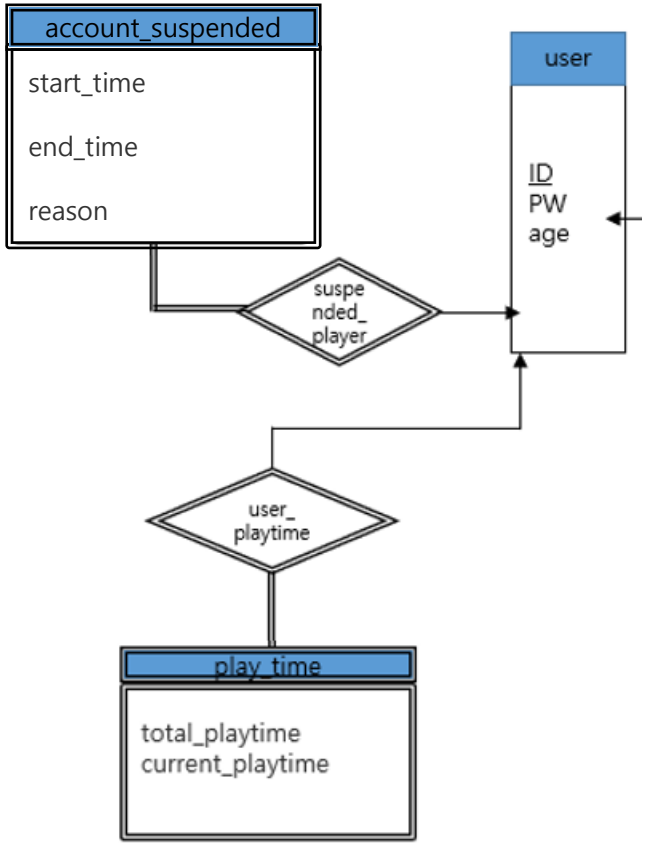
map 엔티티는 앞에서 설명했기에 생략.

NPC 엔티티는 현재 생각으로는 그저 퀘스트를 부여해주는 역할을 가지고 있다고 계획했기에 이름만을 속성으로 갖게 하였다. 그리고 NPC는 반드시 등장하는 map이 있고, 중복 등장이 안된다라고 제약했기에 전체참여로 map엔티티와 relation을 이루고있고 quest는 npc를 통해서 전달되어야 한다고 제약을 했기에 quest와는 다:1의 관계로 맺어져있다.

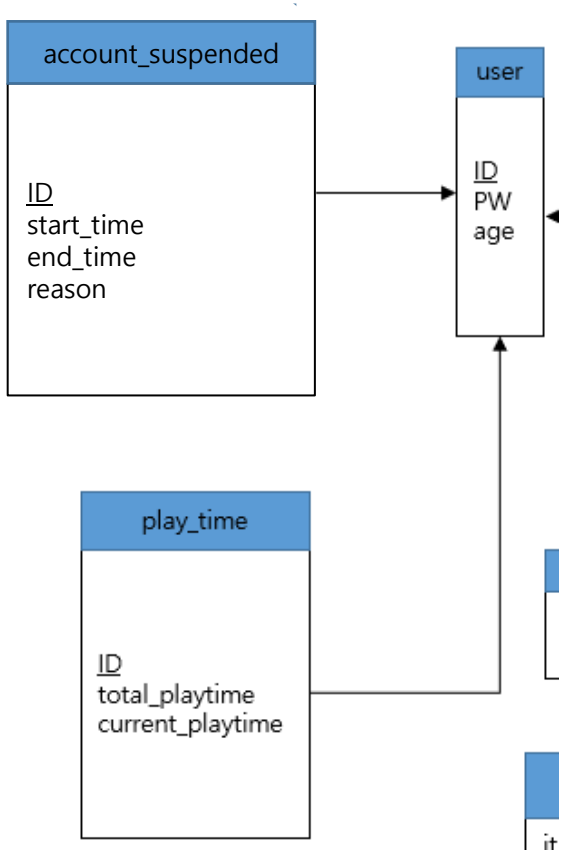
quest 엔티티는 앞에서 설명했기에 생략.

(3)RDB 테이블 스키마


account_suspended
ID
start_time
end_time
reason
user
ID
PW
age
game_character
nick_name
level
current_exp
HP
MP
STR
LUK
INT
DEX
money
class
ID
character_wearing
nick_name
wearing
ongoing_quest
nick_name
quest_name
accept_date
dead_line
quest
quest_name
reward_money
reward_item
reward_exp
completed_quest
nick_name
quest_name
accomplished_date
prereq
quest_name
pre_quest
play_time
ID
total_playtime
current_playtime
inventory
nick_name
item_name
char_skill
nick_name
skill_name
skill_level
p_skill
skill_name
basic_damage
passive
active
stat_type
npc_quest
quest_name
NPC_name
Item
item_name
item_type
item_price
HP
MP
STR
LUK
INT
DEX
drop_item
monster_name
item_name
monster
monster_name
level
reward_money
reward_item
reward_exp
damage
m_skill
skill_name
damage
passive
active
monster_name
monster_map
monster_name
map_name
map
map_name
required_level
NPC
NPC_name
map_name
전체적인 RDB의 모습

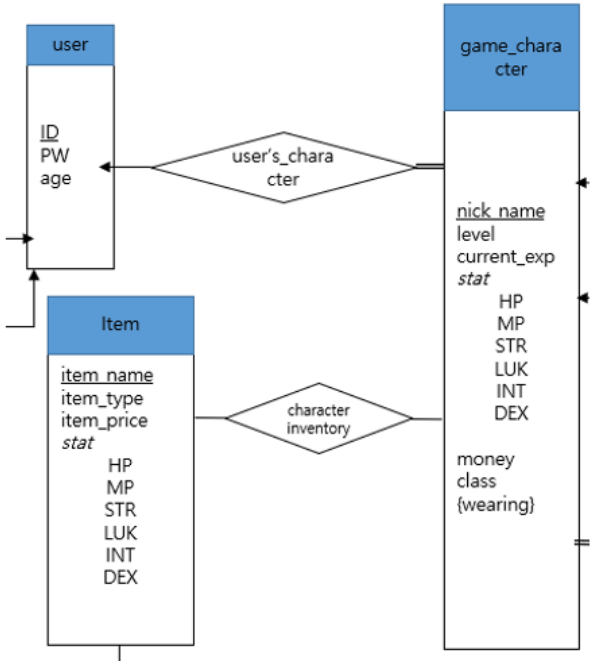


ERD

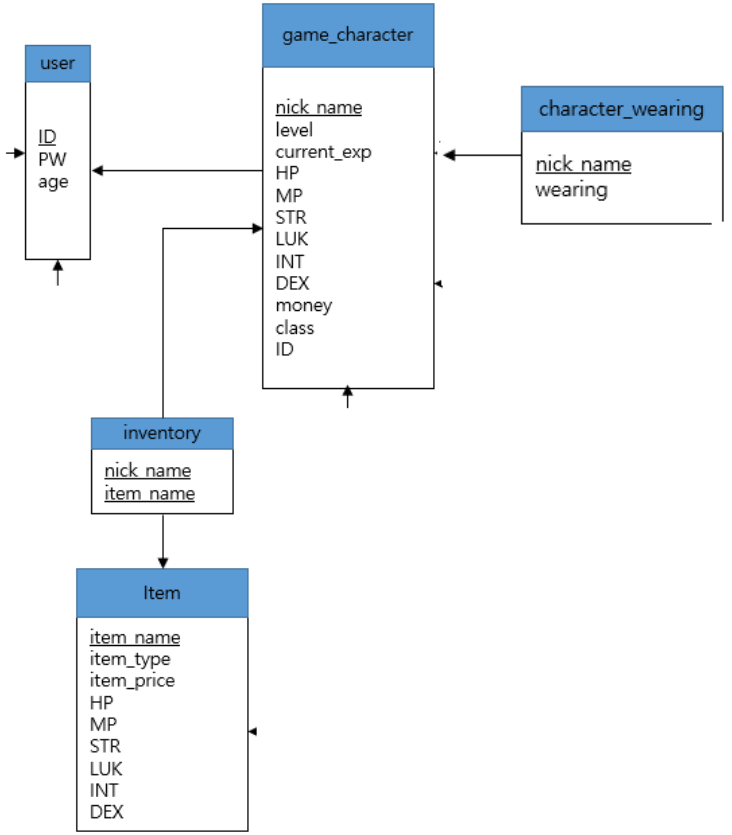


RDB

account_suspende는 weak entity이므로 의존하는 strong entit인 user의 pk를 가져와서 PK를 구성한다. user엔티티는 그대로 테이블로 만들어 주면 되고 play_time엔티티는 마찬가지로 weak entity이므로 user의 pk인 ID를 가져와서 PK로 적용하여 테이블을 만들어준다. 이때 play_time,account_suspended는 partial key가 없기때문에 테이블변환과정에서 ID단독으로 PK를 구성한다.

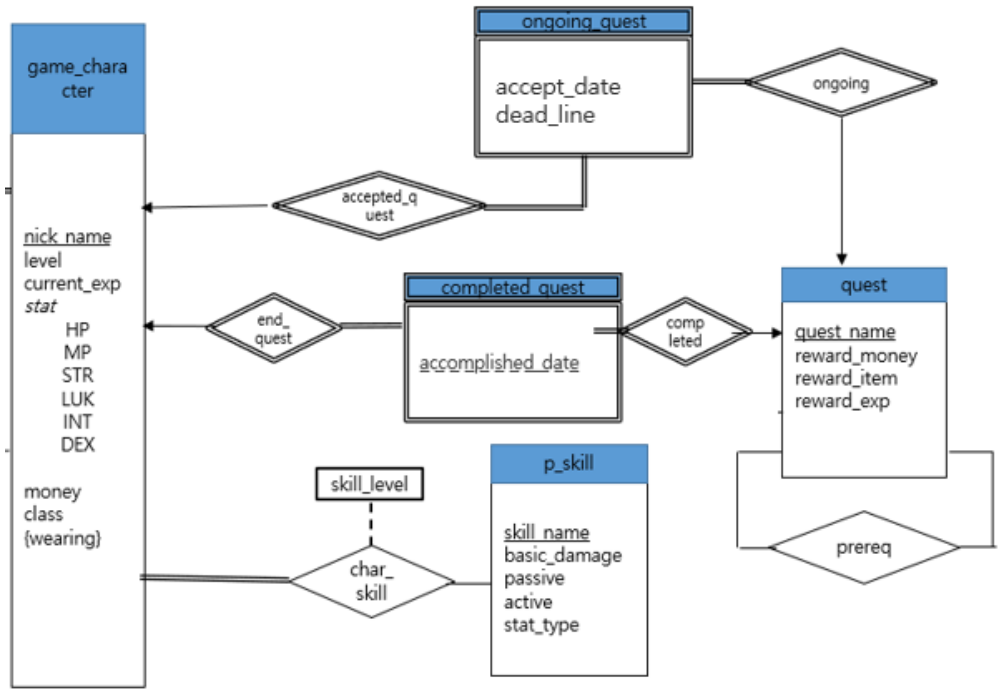


ERD

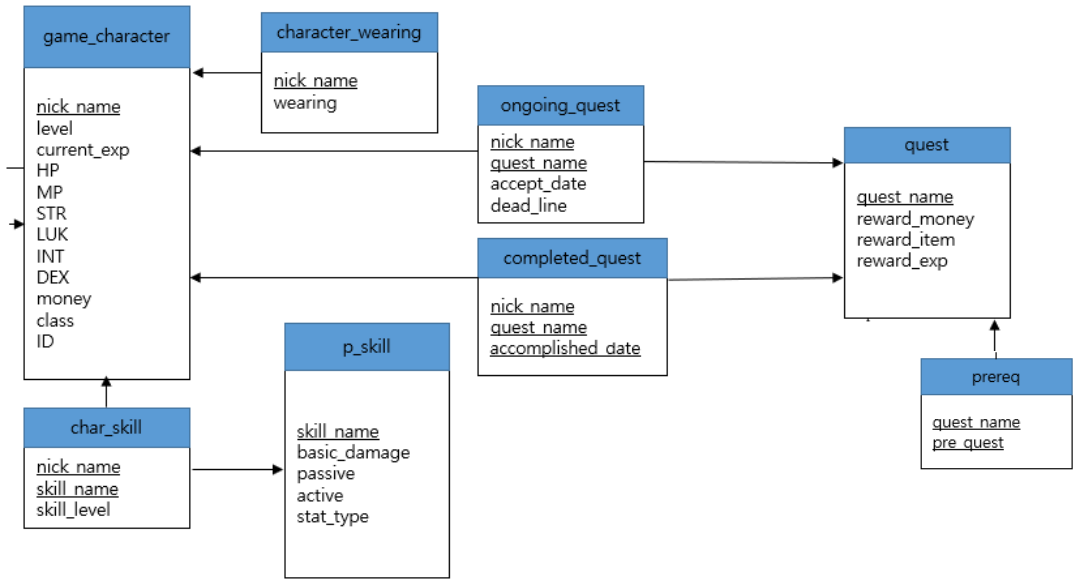


RDB

game_character 엔티티의 composite attribute인 stat은 flatten해져서 HP부터 DEX까지가 각각의 컬럼으로 테이블 스키마에 표현되고 multivalued attributes인 {wearing}은 별도의 character_wearin이라는 테이블을 만들어서 game_characte의 PK인 nick_name과 같이 테이블을 따로 만들어준다. item 엔티티 또한 composite attribut인 stat이 flatten하게 표현되고 item과 game_characte과의 relation 을 inventory라는 테이블을 통해서 표현해준다. 이때 이는 각 테이블에서 PK를 가져와서 합성키로 구성한다.

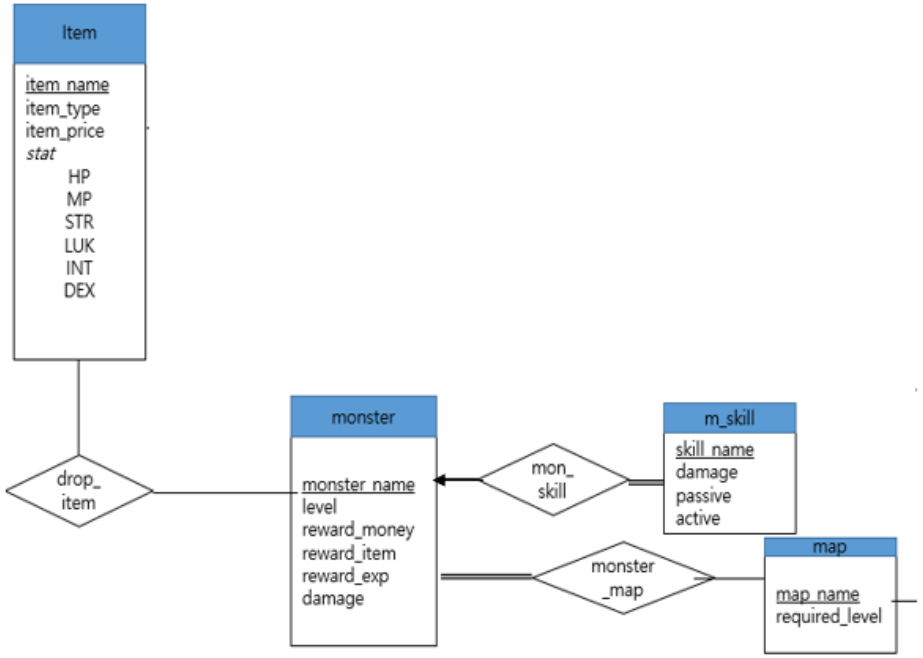


ERD

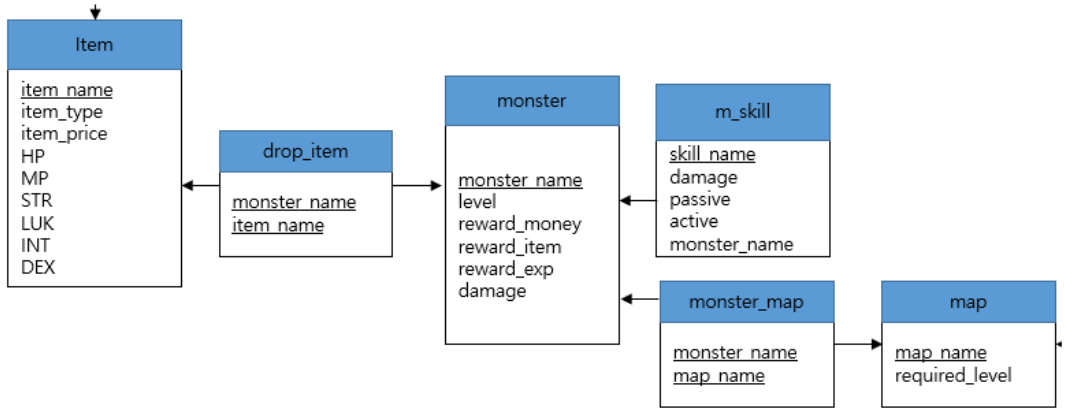


RDB

ongoing_quest와 completed_quest는 game_character와 quest에 의존하는 weak entity이기 때문에 의존하는 테이블에서 PK를 가져와서 테이블을 구성해준다. 이때 ongoing_quest는 partial key가 없기때문에 strong entity로부터 가져온 PK만으로 PK가 지정되고 completed_quest는 partial key가 있기때문에 그것을 포함해서 테이블 스키마의 PK가 된다. 그리고 p_skill 엔티티는 그대로 테이블스키마로 생성되고 char_skill릴레이션은 릴레이션이 포함하는 속성인 skill_level을 포함하고 연결된 엔티티에서 PK를 가져와 합성키로 설정하여 테이블스키마를 생성한다.
quest엔티티는 그대로 테이블 스키마로 생성되고 unary relationship을 가지기 때문에 prereq라는 테이블스키마를 따로 만들어서 quest의 pk를 가져와서 이름을 하나만 바꿔서 두번써준다.

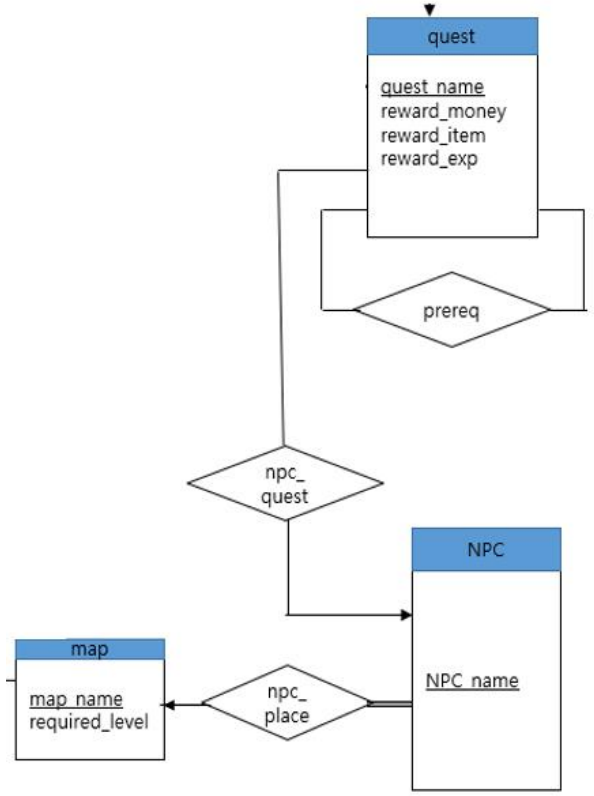


ERD

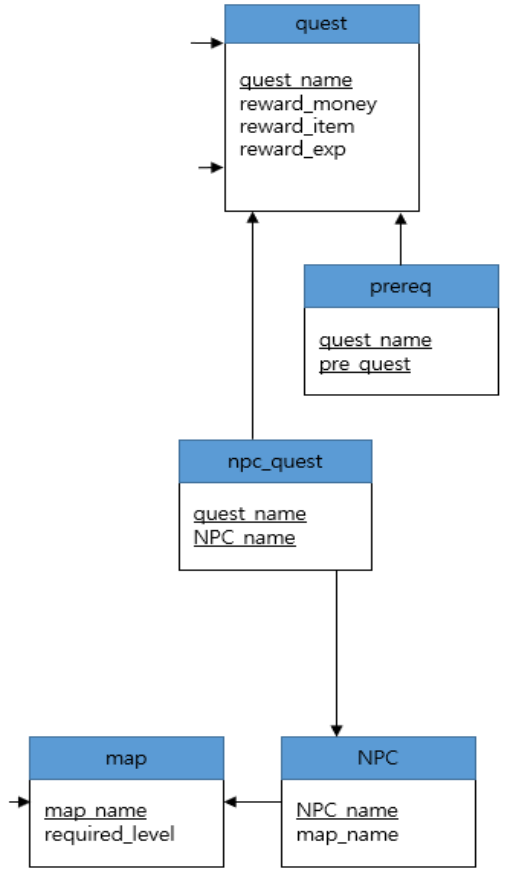


RDB

drop_item 릴레이션은 다:다 이기때문에 item 테이블과 monster 테이블에서 PK를가져와 합성키를 속성으로 갖는 테이블스키마로 만들어준다.
monster는 그대로 테이블 스키마로 만들어주면 되고 m_skill은 monster과 다:1이며 total participation이기 때문에 monster의 PK를 속성으로 추가해줌으로써 mon_skill테이블을 생략하고 m_skill테이블 하나로 만들어준다. 테이블스키마로 변환할때 이런 상황이 발생할때는 대부분 이런식으로 처리했다.
monster_map 릴레이션은 다:다 이기때문에 monster테이블과 map테이블에서 PK를 가져와 합성키를 속성으로 갖는 테이블스키마로 만들어주었다. 그리고 map은 그냥 그대로 테이블스키마로 생성한다.



ERD



RDB

앞에서도 그래왔지만 이미 설명한 테이블스키마는 생략함. (여기선 map,quest,prereq)

npc_place릴레이션은 total participation+다:1 관계이기때문에 따로테이블을 만들어주지 않고 NPC테이블에다가 map의 PK인 map_name을 추가해줌으로써 처리해준다. 그렇게 NPC테이블이 생성되고 난 후 npc_quest릴레이션은 다:다 관계이기때문에 다른 방법없이 그냥 quest와 NPC 테이블에서 PK를 각각 가져와서 합성키로 설정하여 테이블을 만들어 준다.

(4)응용의 복잡도 높은 기능을 지원하기 위한 자연어 질의 리스트 및 해당 SQL문

자연어 질의1): 입력받은 ID가 DB상에 존재하는 ID인지 체크하기 위해 필요한 질의

```
select ID from user where ? in (select ID from user)
```

이것은 복잡한 질의는 아니지만 처음 시작을 맡는 부분인 만큼 넣어봤다.

prepared statement를 사용한다.

자연어 질의2):
현재 접속시간 5시간 이상인 사람을 찾아서 해당 ID가 보유한 모든 캐릭터중에 보유머니가 가장 적은 캐릭터의 닉네임을 찾는다.

해당 SQL문:

```
select nick_name
from game_character
where (ID,money) in (select ID,min(money)
                     from  game_character
                      where ID in (select ID
                                   from play_time
                                   where current_playtime>5)
                      group by ID);
```

게임 관리자로 프로그램을 실행할시에 접속시간 5시간 이상인 사람에게 이벤트를 열기 위한 쿼리문이다. 해당하는 캐릭터에게 게임머니를 주기 위해 존재하는 쿼리문.

자연어 질의3):
현재 접속시간 5시간 이상인 사람을 찾아서 해당 ID가 보유한 모든 캐릭터중에 레벨이 가장 낮은 캐릭터의 닉네임을 찾는다.

해당 SQL문:

```
select nick_name
from game_character
where (ID,level) in (select ID,min(level)
                     from   game_character
                      where ID in (select ID
                                   from play_time
                                   where current_playtime>5)
                      group by ID);
```

게임 관리자로 프로그램을 실행할시에 접속시간 5시간 이상인 사람에게 이벤트를 열기 위한 쿼리문이다. 해당하는 캐릭터(레벨이 가장 낮은)에게 적정 장비를 이벤트로 지급해 주기 위해 존재하는 쿼리문.

자연어 질의4):
캐릭터 이름을 검색하여 해당 캐릭터의 레벨을 얻은 뒤, 해당 레벨이하의 요구조건을 가진 map중 가장 높은 요구레벨을 갖는 map의 이름을 찾는다.

해당 SQL문:

```
1
2   select map_name
3   from map
4   where required_level=
5       (select max(required_level)
6       from map
7       where required_level <all (
8                                 select level
9                                 from game_character
10                                where nick_name=?)&&? in (select nick_name
11                                                          from game_character))
```

//prepared statement이용 예정

user로 입장시 본인의 캐릭터에 맞는 추천 사냥터를 받기 위한 쿼리이다. 자신이 갈 수 있는 사냥터중 가장 요구레벨이 높은 map의 이름을 보여준다.

자연어 질의5):특정 유저 정지기간 부여함. 더 상세

해당 SQL문:

```
use game;

update account_suspended
set end_time=DATE_ADD((select end_time from(select end_time from account_suspended  where account_suspended.ID='n0etccq')T),INTERVAL 1 DAY)
where ID='n0etccq';

```

//prepared statement이용 예정. 이미 정지당한 유저의 경우 정지만료기간을 늘림. prepared statement 이용함 실제로는

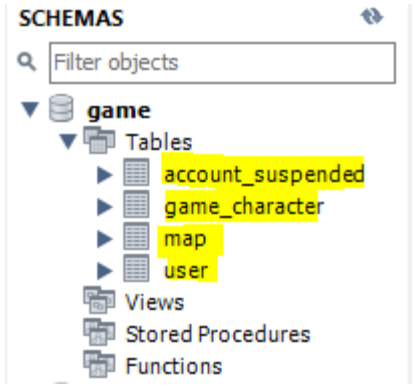
해당 SQL문:

```
insert into account_suspended
  values(?,?,?,?)
```

//prepared statement이용 예정. 정지중이 아닌 유저의 경우 새로 튜플을 삽입해줌.

(5) 대화식 SQL 도구로 DB 생성 및 데이터 적재

```sql
use game;

create table user
(
ID varchar(20),
PW varchar(20),
age int,
primary key(ID)
);

create table account_suspended
(
ID varchar(20),
start_time DATE,
end_time DATE,
reason varchar(30),
primary key(ID)
);

create table game_character
(
nick_name varchar(20) ,
level int,
current_exp int,
HP int,
MP int,
STR int,
LUK int,
IINT int,
DEX int,
money int,
class varchar(20),
ID varchar(20),
primary key(nick_name)
);
create table map
(
map_name varchar(20),
required_level int,
primary key(map_name)
);
```



account_suspended
game_character
map
user
총 4개의 테이블을 생성하였고 이를 이용해 기능을 만들 것이다.

```sql
use game;

insert into user (ID,PW,age)
values
('lgeonho01','lgh01',23),
('n0etccq','kfiiig',26),
('k43ncdk','jhsai1',33),
('yqe6a0j','hsi3sc',21),
('7qy69up','hd1e4x',18),
('p1tpgg0cbzg','fozphv',28),
('d6hxn2','n97xt0',23),
('wxtk8g','dauslx',29),
('f19xjy','lbk0m9',34),
('wy6v1v','hg42tp',38),
('4wrimz','fr061h',32);

insert into map (map_name,required_level)
values
('forest',0),
('snowfield',5),
('beach',10),
('city',15),
('heaven',20),
('hell',25),
('mountain',30),
('desert',35),
('ocean',40),
('volcano',45),
('space',50);


```

```sql

insert into game_character
(nick_name,level,current_exp,HP,MP,STR,LUK,IINT,DEX,money,class,ID)
values
('Druan',4,100,100,100,100,10,20,30,3000,'warrior','lgeonho01'),
('Hunus',8,100,50,50,10,100,30,100,2000,'magician','lgeonho01'),
('Jerri',12,200,60,60,20,150,20,40,4000,'archer','lgeonho01'),
('Millin',16,250,60,60,20,150,20,40,5000,'archer','n0etccq'),
('Donan',20,350,630,60,20,150,20,40,6000,'warrior','k43ncdk'),
('Horgeren',24,330,330,620,20,150,20,40,3000,'assassin','yqe6a0j'),
('Zargais',28,230,130,670,720,170,720,40,13000,'assassin','7qy69up'),
('Rorix',32,650,230,670,720,150,320,40,15500,'magician','p1tpgg0cbzg'),
('Chuman',36,250,1130,670,720,150,320,40,134600,'magician','d6hxn2'),
('Jonelle',40,250,1230,330,1420,150,320,40,132400,'warrior','wxtk8g'),
('Silin',44,2520,1232,430,220,150,320,40,1222400,'warrior','f19xjy'),
('Milia',48,2222,1242,430,220,150,320,40,1222400,'archer','wy6v1v'),
('Cales',52,2222,1242,430,220,150,320,40,1222400,'archer','4wrimz');

insert into account_suspended
(ID,start_time,end_time,reason)
values
('n0etccq','2021-12-01','2021-12-12','swear word'),
('k43ncdk','2021-11-01','2022-01-22','cheating program');
```

account_suspended 테이블을 제외하고 최소 10개 이상씩의 tuple을 만들었다.

account_suspended 테이블을 2개의 튜플만 삽입한 이유는 정지당한 목록이기에 너무 많을 경우 기능 실현에서 문제가 있을것 같아서이다.

(6) JDBC/MySQL 프로그램

구현할 기능은 다음과 같다.
먼저 프로그램을 실행하면 **1.user로 입장 / 2. 게임 관리자로 입장** 선택창이 나온다.
1을 선택할 경우 로그인 기능을 수행한 다음 정지된 계정이 아닐 경우 캐릭터 이름을 검색하여 추천 사냥터 결과를 받는 입출력문이 실행된다.

2를 선택할 경우 특정 유저 정지기간 부여하는 메뉴가 실행된다.

3을 누르면 종료된다.

ID	PW	age
7qy69up	hd1e4x	18
d6hxn2	n97xt0	23
f19xjy	lbk0m9	34
k43ncdk	jhsai1	33
lgeonho01	lgh01	23
n0etccq	kfiiig	26
p1tpgg0cbzg	fozphv	28
wxtk8g	dauslx	29
wy6v1v	hg42tp	38

user 인스턴스

nick_name	level	current_exp	HP	MP	STR	LUK	IINT	DEX	money	class	ID
Cales	52	2222	1242	430	220	150	320	40	1222400	archer	4wrimz
Chuman	36	250	1130	670	720	150	320	40	134600	magician	d6hxn2
Donan	20	350	630	60	20	150	20	40	6000	warrior	k43ncdk
Druan	4	100	100	100	100	10	20	30	3000	warrior	lgeonho01
Horgeren	24	330	330	620	20	150	20	40	3000	assassin	yqe6a0j
Hunus	8	100	50	50	10	100	30	100	2000	magician	lgeonho01
Jerri	12	200	60	60	20	150	20	40	4000	archer	lgeonho01
Jonelle	40	250	1230	330	1420	150	320	40	132400	warrior	wxtk8g
Milia	48	2222	1242	430	220	150	320	40	1222400	archer	wy6v1v
Millin	16	250	60	60	20	150	20	40	5000	archer	n0etccq
Rorix	32	650	230	670	720	150	320	40	15500	magician	p1tpgg0c...
Silin	44	2520	1232	430	220	150	320	40	1222400	warrior	f19xjy
Zargais	28	230	130	670	720	170	720	40	13000	assassin	7qy69up

game_character 인스턴스

map_name	required_level
beach	10
city	15
desert	35
forest	0
heaven	20
hell	25
mountain	30
ocean	40
snowfield	5
space	50
volcano	45

map 인스턴스

ID	start_time	end_time	reason
k43ncdk	2021-11-01	2022-01-22	cheating program
n0etccq	2021-12-01	2021-12-16	swear word

account_suspended 인스턴스

```
1.USER로 입장 2.게임 관리자로 입장 3.종료:
1
로그인할 ID를 입력해주세요:
lgeonho01
로그인할 PW를 입력해주세요:
lgh01
아이디와 비밀번호가 일치합니다.
적정사냥터를 추천받고 싶은 캐릭터의 이름을 입력해주세요:
Jerri
현재 해당 캐릭터에게 가장 적합한 사냥터(맵)은 : beach
프로그램이 정상적으로 끝났습니다. 종료합니다.
```

ID	PW	age
7qy69up	hd1e4x	18
d6hxn2	n97xt0	23
f19xjy	lbk0m9	34
k43ncdk	jhsai1	33
lgeonho01	lgh01	23
n0etccq	kfiiig	26
p1tpgg0cbzg	fozphv	28
wxtk8g	dauslx	29
wy6v1v	hg42tp	38

user 인스턴스

map_name	required_level
beach	10
city	15
desert	35
forest	0
heaven	20
hell	25
mountain	30
ocean	40
snowfield	5
space	50
volcano	45
NULL	NULL

map 인스턴스

첫번째 구현한 것은 user로 들어가는 것이다. 1을 눌러서 user모드로 들어가면 로그인할 ID 와 PW를 입력받는다. 입력받은 것을 토대로 DB속 user인스턴스를 검색하고 존재한다면 정상 실행하고 없다면 아이디와 비밀번호가 틀렸다하고 종료한다.
정상적으로 id,pw를 입력하고 다음단계로 진입하면 생성한 캐릭터의 닉네임을 검색하면 해당 캐릭터에게 가장 적합한 사냥터를 DB속 map인스턴스를 참조하여 계산해서 보여준다. 원리는 캐릭터 닉네임을 검색하면

nick_name	level	current_exp	HP	MP	STR	LUK	IINT	DEX	money	class	ID
Cales	52	2222	1242	430	220	150	320	40	1222400	archer	4wrimz
Chuman	36	250	1130	670	720	150	320	40	134600	magician	d6hxn2
Donan	20	350	630	60	20	150	20	40	6000	warrior	k43ncdk
Druan	4	100	100	100	100	10	20	30	3000	warrior	lgeonho01
Horgeren	24	330	330	620	20	150	20	40	3000	assassin	yqe6a0j
Hunus	8	100	50	50	10	100	30	100	2000	magician	lgeonho01
Jerri	12	200	60	60	20	150	20	40	4000	archer	lgeonho01
Jonelle	40	250	1230	330	1420	150	320	40	132400	warrior	wxtk8g
Milia	48	2222	1242	430	220	150	320	40	1222400	archer	wy6v1v
Millin	16	250	60	60	20	150	20	40	5000	archer	n0etccq
Rorix	32	650	230	670	720	150	320	40	15500	magician	p1tpgg0c...
Silin	44	2520	1232	430	220	150	320	40	1222400	warrior	f19xjy
Zargais	28	230	130	670	720	170	720	40	13000	assassin	7qy69up
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

game_character 인스턴스

game_character 인스턴스를 참조하는데 이때 해당 캐릭의 레벨이 나와있다. 이를 이용하여 그 캐릭이 갈 수 있는 사냥터중 가장 높은 레벨을 요구하는 사냥터의 이름을 보여주는 것이다.
결과를 보면 알다시피 Jerri의 레벨은 12이기에 갈 수 있는 가장 높은 사냥터는 beach(레벨제한10) 이다.

```
else if(input==1) //user
{
    System.out.println("로그인할 ID를 입력해주세요:");
    String id=scanner.next();
    System.out.println("로그인할 PW를 입력해주세요:");
    String pw=scanner.next();

    ResultSet rset=stmt.executeQuery("select ID from account_suspended"); //정지당한 아이디면 메세지 출력후 종료
    while(rset.next()) //계정이 정지 당했나 여부확인.
    {
        String temp=rset.getString("ID");
        if(id.equals(temp))
        {
            System.out.println("계정이 정지당한 상태입니다. 프로그램을 종료합니다.");
            System.exit(0); //정상종료
        }
    }

    rset=stmt.executeQuery("select * from user");
    while(rset.next())
    {
        String tem_id=rset.getString("ID");
        String tem_pw=rset.getString("PW");
        if(id.equals(tem_id)&&pw.equals(tem_pw))//DB상에 있는 id,비번일때만 정상실행
        {
            System.out.println("아이디와 비밀번호가 일치합니다. \n적정사냥터를 추천받고 싶은 캐릭터의 이름을 입력해주세요:");
            String nickn=scanner.next();
            PreparedStatement pStmt=conn.prepareStatement("select map_name from map where required_level=(select max(required_level)from map where required_level<all(select level from game_character where r
            pStmt.setString(1,nickn);
            pStmt.setString(2,nickn);
            ResultSet rs;
            rs=pStmt.executeQuery();
            if(rs.next()==false)
            {
                System.out.println("입력한 닉네임은 존재하지 않습니다. 프로그램을 종료합니다.");
                System.exit(0); //정상종료
            }
            System.out.println("현재 해당 캐릭터에게 가장 적합한 사냥터(맵)은 : "+rs.getString("map_name"));
            System.out.println("프로그램이 정상적으로 끝났습니다. 종료합니다.");
            rs.close();
            System.exit(0); //정상종료
        }
    }
    System.out.println("아이디와 비밀번호가 틀렸습니다. 종료합니다.");
    System.exit(0); //정상종료
```

앞에서 설명한 내용을 실행하는 코드이다. preparedStatement를 이용하여 사람에게 입력받은 정보를 가지고 DB쿼리를 그때그때 다르게 수행할 수 있었다. 물론 당연히 캐릭터의 이름을 잘못 입력했을때 경고를 띄우는 기능과 아이디와 비밀번호가 틀렸을때 종료되는 기능도 들어가 있다.

```
1.USER로 입장 2.게임 관리자로 입장 3.종료:
2
정지를 주고자 하는 유저의 ID:
k43ncdk
정지 시작 날짜(ex.2021-10-11):
2021-12-03
정지 만료 날짜(ex.2021-10-11):
2022-02-02
정지 사유:
hacking
계정이 정지당한 상태입니다. 정지 만료기간을 연장합니다.

ID:k43ncdk    start_time:2021-11-01    end_time:2022-01-22    reason:cheating program
몇일을 연장시키겠습니까?(일 단위 입력):

30
변경 후 상태는->ID:k43ncdk    start_time:2021-11-01    end_time:2022-02-21    reason:cheating program

종료합니다.
```

	ID	start_time	end_time	reason
▶	k43ncdk	2021-11-01	2022-01-22	cheating program
	n0etccq	2021-12-01	2021-12-16	swear word
*	NULL	NULL	NULL	NULL

account_suspended 인스턴스

두번째로 구현한 것은 게임관리자 모드로 들어갔을때 특정 유저를 날짜를 지정하여 정지 시키는 기능이다. 현재 구현한 것은 이미 account_suspended 디비에 있는, 즉 이미 정지중인 유저에게 추가적인 정지를 가하는 기능이다. 정지를 주고자 하는 유저의 ID를 검색하면 먼저 account_suspended인스턴스를 검색하는데 이때 여기서 발견이 되면 다음단계로 넘어간다. 그러면 현재 정지상태의 정보를 띄워줌과 동시에 정지를 얼마나 연장시킬 것인가를 묻는다. 이때 단위는 '일'단위이며 위 예와같이 정지만료일이 22년1월22일 에서 30을 연장시키겠다고 입력하면 자동으로 date를 계산하여 단순 string이 아닌 날짜계산방식을 통해 저장되는 것을 알 수 있다.

```
1.USER로 입장 2.게임 관리자로 입장 3.종료:
1
로그인할 ID를 입력해주세요:
k43ncdk
로그인할 PW를 입력해주세요:
sadfvc3
계정이 정지당한 상태입니다. 프로그램을 종료합니다.
```

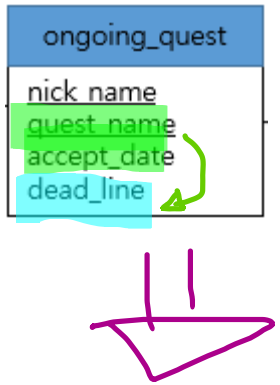
정지당해서 account_suspended 디비에 추가된 계정은 user로서 접속 이 왼쪽 사진과 같이 제한된다.

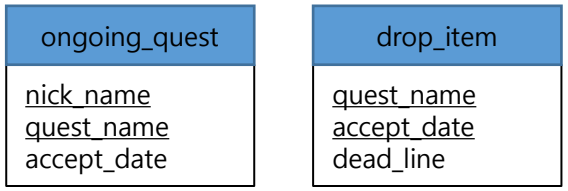
```
else if(input==2) //게임 관리자
{
    ResultSet rset=stmt.executeQuery("select ID from account_suspended");

    ResultSet check;
    PreparedStatement pStmt=conn.prepareStatement("select ID from user where ? in (select ID from user)");

    System.out.println("정지를 주고자 하는 유저의 ID:");
    String id=scanner.next();
    pStmt.setString(1, id);
    check=pStmt.executeQuery();
    if(check.next()==false)
    {
        System.out.println("해당 ID를 가진 user는 존재하지 않습니다. 프로그램을 종료합니다.");
        System.exit(0); //정상종료
    }


    System.out.println("정지 시작 날짜(ex.2021-10-11):");
    String start=scanner.next();
    System.out.println("정지 만료 날짜(ex.2021-10-11):");
    String end=scanner.next();
    System.out.println("정지 사유:");
    String reason=scanner.next();

    while(rset.next()) //계정이 정지 당했나 여부확인.
    {
        String temp=rset.getString("ID");
        if(id.equals(temp)) //조건을 만족하면 이미 정지당한계정. 따라서 새로운 튜플을 추가하는 것이 아닌 정지 만료기간을 연장함
        {
            PreparedStatement ps;
            pStmt=conn.prepareStatement("select * from account_suspended where ID=?");
            pStmt.setString(1,id);
            ResultSet rs;
            rs=pStmt.executeQuery();
            rs.next();
            System.out.println("계정이 정지당한 상태입니다. 정지 만료기간을 연장합니다.\n");
            System.out.println("ID:" + rs.getString("ID")+"    start_time:"+rs.getDate("start_time")+"    end_time:"+rs.getDate("end_time")+"    reason:"+rs.getString("reason"));
            System.out.println("몇일을 연장시키겠습니까?(일 단위 입력):\n");
            int extend=scanner.nextInt();
            ps=conn.prepareStatement("update account_suspended set end_time=DATE_ADD((select end_time from(select end_time from account_suspended where account_suspended.ID=?)T),INTERVAL ? DAY) wher
            ps.setString(1, id);
            ps.setInt(2, extend);
            ps.setString(3, id);
            ps.executeUpdate();
            rs=pStmt.executeQuery();
            rs.next();

            ;
            System.out.println("변경 후 상태는->ID:" + rs.getString(1)+"      start_time:"+rs.getDate(2)+"      end_time:"+rs.getDate(3)+"      reason:"+rs.getString(4));
            System.out.println("\n종료합니다.");
```

해당 부분 소스코드이다.

(6) BCNF 정규화 및 스키마 정제

BCNF가 아닌 스키마를 가져와서 함수종속을 열거/설명 후 BCNF로 정규화를 수행해 보도록 하겠다.





BCNF에 위배되는 테이블스키마를 딱 1개 찾을 수 있었다.

{quest_name,accep_date} ->dead_line 의 함수종속 (functional dependency)를 지니고 있다. 퀘스트가 무엇인지,그리고 언제 수락했는지를 알면 dead_line이 자동으로 정해진다고 가정하고 설계한 스키마이기 때문이다.

{quest_name,accep_date}는 해당 테이블에서 고유식별자가 되지 않는다. 왜냐하면 같은 퀘스트를 같은날짜에 받았더라도 여러 다른캐릭들이 있기때문에 그 날짜에 그 퀘스트를 받은것은 여러 번 중복될 수 있기때문이다. 따라서 이를 BCNF로 분해하기 위해서는 알파= {quest_name,accep_date}, 베타=dead_line이므로 다음과 같이 두 개의 테이블로 나눌 수 있을것이다. 그러나 고작 dead_line하나를 위해서 두 테이블로 분리하는 것은 추후에 탐색과정에 있어서도 매우 불편할 것으로 예상되기때문에 최종적으로는 분해를 하지 않고 쓰는것이 낫다고 결정했다.

이상으로 마치겠습니다.