Abstract

Dota2 è un videogioco per PC rilasciato nel 2012 dalla Valve Corporation come sequel di una modalità di gioco personalizzata del celeberrimo Warcraft 3, chiamata appunto DotA(Defense of the Ancients).
Questo free-to-play di casa Valve ha avuto da subito un successo imponente, arrivando recentemente a contare più di dieci milioni di giocatori sparsi per il mondo e competizioni a livello nazionale ed internazionale con montepremi esorbitanti: basti pensare ai 10 931 105$ del torneo del mondo 2014. In molte nazioni, specialmente orientali, come Cina, Corea etc questo videogame è diventato uno sport a tutti gli effetti, addirittura con delle federazioni sportive dedicate.
Dota2 fa parte della grande famiglia dei MOBA(Multiplayer Online Battle Arena) in cui due squadre da cinque giocatori ciascuna si scontrano in una battaglia strategica di circa un'ora di durata, il cui scopo è abbattere le forze nemiche e farsi largo fino all'opposta base, per poter colpire e distruggere la più grande e maestosa delle strutture presenti nella mappa: l'Antico, o Ancient.
Ogni giocatore deve scegliere inizialmente uno tra i 110 eroi disponibili, ognuno dei quali è dotato di un certo insieme di abilità, o spell, che lo rendono unico e che gli conferiscono un certo ruolo nell'economia della partita. Questa scelta deve essere effettuata in maniera da comporre un team bilanciato e con una buona sinergia tra i vari eroi.
Le combinazioni sono molte e la complessità è aumentata ancor più dal fatto che, durante la partita, ogni player è chiamato a scegliere, tra i 137 disponibili, quali oggetti comprare per il proprio personaggio: la scelta è cruciale, ogni oggetto conferisce bonus diversi: un'oculata scelta di composizione del team e degli items può fare la differenza tra la vittoria e la sconfitta.
Inoltre non solo c'è un limite di 6 agli oggetti che un eroe può avere con sè, ma essi devono anche essere acquistati con l'oro, risorsa difficile da ottenere, il che rende ancor più ardua la scelta.
Come è facile pensare, le combinazioni eroi-oggetti possibili sono un numero esorbitate, e prenderle in considerazione tutte ha una complessità poco umana. DotaDB dovrebbe aiutare in questo, permette tra le altre cose di consultare le informazioni specifiche di ogni eroe e di ogni oggetto, di provare varie composizioni oggetti su uno specifico eroe etc.

Descrizione dei requisiti e delle operazioni tipiche

Si vuole realizzare una base di dati e la relativa applicazione web che contenga e gestisca le informazioni relative al gioco Online Dota2.
In particolare si vogliono conoscere dati relativi agli Eroi, agli Oggetti ed alle possibili composizioni di questi ultimi create dall'utente.
Di taluni aspetti, in questa base di dati, si è preferito considerare una versione semplificata per non aumentarne inutilmente la complessità senza alcun miglioramento dal punto di vista didattico.

Ogni Eroe quindi è caratterizzato da:
- un nome, che lo identifica univocamente
- l'attributo primario, che ne descrive, molto genericamente, l'attitudine ed il ruolo (robusto, agile o magico)
- una biografia, che contiene generalmente informazioni riguardo l'ambientazione immaginaria di WarcraftIII
- valori di partenza di forza, agilità, intelletto e attacco (*base stats*)
- quanta forza, agilità ed intelletto sono guadagnati al passaggio di livello (*stats gain*)
- bonus di forza, agilità, intelletto ed attacco conferiti dagli oggetti posseduti (*bonus stats*)
- il livello

Ogni Eroe è in grado, di norma, di lanciare da 4 a 6 magie. Che possono essere *attive* o *passive*.
Ogni Spell è caratterizzata da:
- un nome, che la identifica univocamente
- un tasto che le è assegnato come *shortcut*
- una descrizione
- il tipo di abilità, ovvero la maniera in cui essa interagisce con il gioco
- l'*affection*, ovvero quali unità può colpire (solo alleati, solo nemici, entrambi, neutrali....)
- può o meno essere l'ultimate

Solo le Attive hanno anche:
- il tipo di danno (fisico, magico, puro)
- possono essere o meno capaci di andare oltre l'immunità alla magia
- hanno 3 o 4 livelli, ognuno dei quali con tempo di recupero, costo di Mana e danno diversi

Le passive invece sono caratterizzate semplicemente da una descrizione dell'effetto passivo.

L'altra grande famiglia di interesse è composta dagli *Items*, ovvero gli oggetti che ogni giocatore può comprare per il proprio eroe durate la partita.
Ogni Oggetto è caratterizzato da:
- un nome, che lo identifica univocamente
- un costo in oro
- una descrizione
- il tipo (passivo o attivo)
- i bonus di forza, agilità, intelletto ed attacco che conferisce

Inoltre può avere degli effetti attivi, caratterizzati da nome e descrizione.
Nella mappa sono presenti tre diverse tipologie di negozio e non tutte hanno a disposizione tutti gli oggetti, quindi bisogna tenere conto anche di questo aspetto.

L'utente di questo applicativo potrà comporre come preferisce oggetti ed eroi, modificare i livelli e, giunto alla composizione desiderata, controllare le statistiche aggiornate.
Inoltre ovviamente sarà possibile controllare e modificare/aggiornare le caratteristiche ed i dati di ogni personaggio e di ogni *Item*.

Progettazione Concettuale:

Entità ed Attributi

Eroi (chiave primaria: nome)
- nome: string
- attributo primario: string
- biografia: text
- base_str: float
- base_agl: float
- base_int: float
- base_atk: float
- gain_str: float
- gain_agl: float
- gain_int: float
- bonus_str: float
- bonus_agl: float
- bonus_int: float
- bonus_atk: float
- livello: int

Spell (chiave primaria: nome)
- nome: string
- tasto: char
- descrizione: text
- tipo di abilità: string
- affection: string
- ultimate: bool

Spell_Attive:
- tipo di danno: string
- piercing: bool

Spell_Passive:
- effetto: text

DettagliSpell (chiave primaria: livello + Spell(nome))
- livello: int
- CD: float
- costo di mana: int
- danno: float

Relazioni

Eroi – Spell: Possesso
- ogni eroe ha più spell
- ogni spell è posseduta da uno ed un solo eroe

Spell_Attive – DettagliSpell: Specifiche
- ogni spell attiva ha più di un livello e quindi più specifiche differenti a seconda del livello
- ogni dettaglio è ovviamente di una ed una sola spell

Entità ed Attributi

Oggetti (chiave primaria: nome)
- nome: string
- costo: int
- descrizione: text
- tipo: string
- bonus_str: float
- bonus_agl: float
- bonus_int: float
- bonus_atk: float

Negozio (chiave primaria: nome)
- nome: string

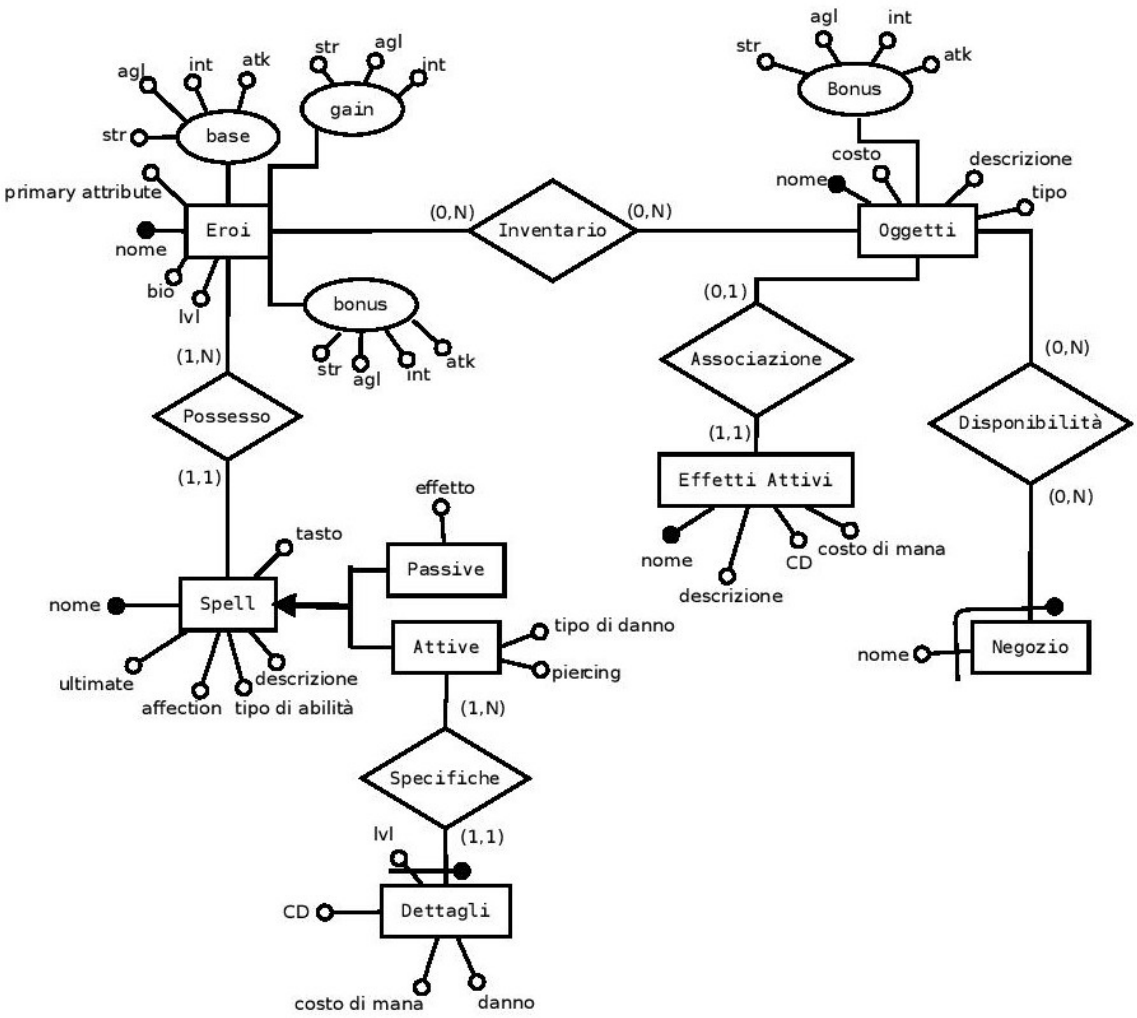
EffettiAttivi (chiave primaria: nome)
- nome: string
- descrizione: text
- CD: float
- costo di mana: int

Relazioni

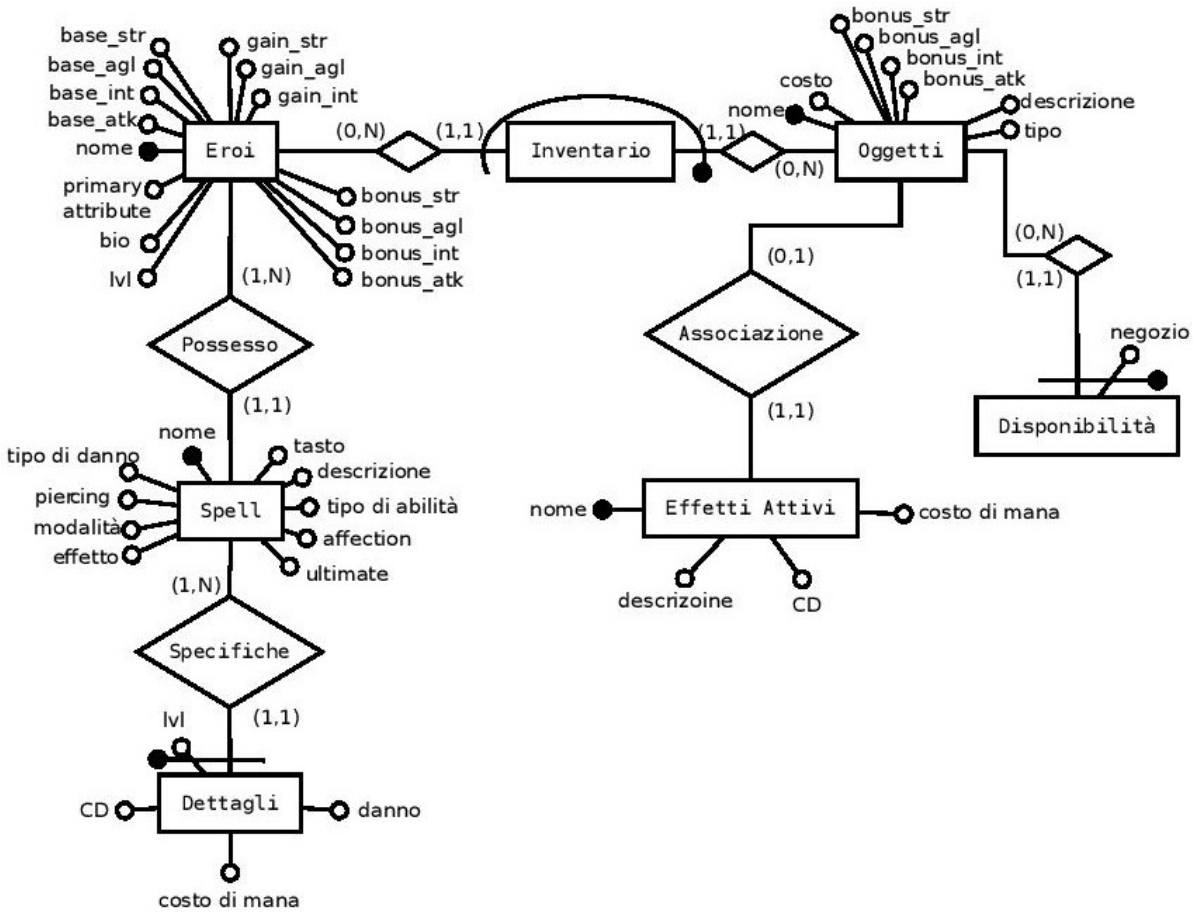
Oggetti – Negozio: Disponibilità
- ogni oggetto è disponibile in qualche negozio
- ogni negozio ha disponibilità di qualche oggetto

Oggetti – EffettiAttivi: Associazione
- qualche oggetto è associato ad un solo effetto attivo
- ogni effetto attivo è associato ad un solo oggetto

Eroi – Oggetti: Inventario
- ogni eroe può avere più oggetti
- ogni oggetto oggetto può essere posseduto da più eroi


agl
int
atk
str
base
gain
str
agl
int
primary attribute
Eroi
nome
bio
lvl
(0,N)
Inventario
(0,N)
bonus
str
agl
int
atk
(1,N)
Possesso
(1,1)
agl
int
str
atk
Bonus
costo
nome
descrizione
tipo
Oggetti
(0,1)
Associazione
(1,1)
Effetti Attivi
nome
descrizione
CD
costo di mana
(0,N)
Disponibilità
(0,N)
nome
Negozio
effetto
tasto
Passive
nome
Spell
Attive
tipo di danno
piercing
ultimate
affection
tipo di abilità
descrizione
(1,N)
Specifiche
lvl
(1,1)
CD
Dettagli
costo di mana
danno

Progettazione Logica



Lo schema ER del punto precedente è stato ristrutturato:

Non erano presenti ridondanze significative, eccetto per gli attributi bonus_xxx dell'entità “Eroi” che potrebbe essere ricavata come somma degli attributi bonus_xxx di ogni singolo oggetto presente nell'inventario dell'eroe. Dato che i bonus sono immediatamente visualizzati ad ogni accesso alla pagina dell'eroe ho pensato di conservare questa ridondanza escludendo eventuali incosistenze mediante l'uso di appositi trigger.

Gli attributi composti (come *bonus*, *gain* e *base*) sono stati sodduddivisi in attributi semplici.

La relazione molti a molti “Inventario” è stata sostituita da una nuova entità (anch'essa chiamata “Inventario”) in relazione sia con “Eroi” che con “Oggetti”.

La relazione molti a molti “Disponibilità” e l'entità “Negozio” sono state unite in una unica entità “Disponibilità” in relazione uno a molti con “Oggetti”. Questo è stato possibile e non è causa di anomalie in quanto ogni negozio è caratterizzato solo dal proprio nome. **Non** è possibile aggiungere/modificare le tipologie di negozio in quanto nella mappa ce ne sono solo 3 ed esse sono note a priori (Side, Main, Secret).

La gerarchia Spell-SpellAttive-SpellPassive è stata eliminata accorpando le figlie al padre. La scelta è stata quasi obbligata in quanto la generalizzazione era totale e gli accessi al padre e le figlie è

quasi sempre contestuale. Si è quindi aggiunto l'attributo “modalità” che indica se la spell è attiva o passiva.

Il resto è rimasto invariato.

A questo punto le relazioni presenti sono le seguenti:
Eroi(nome, primary attribute, bio, lvl, base_str, base_agl, base_int, base_atk, gain_str, gain_agl, gain_int, bonus_str, bonus_agl, bonus_int, bonus_atk)

Spell(nome, eroe, modalità, tasto, descrizione, tipo di abilità, affection, ultimate, piercing, effetto)
vincolo di integrità referenziale tra Spell(eroe) e Eroi(nome)

Dettagli(lvl, spell, CD, costo di mana, danno)
vincolo di integrità referenziale tra Dettagli(spell) e Spell(nome)

Oggetti(nome, costo, descrizione, tipo, bonus_str, bonus_agl, bonus_int, bonus_atk)

Inventario(eroe, oggetto)
vincolo di integrità referenziale tra Inventario(eroe) e Eroi(nome)
vincolo di integrità referenziale tra Inventario(oggetto) e Oggetti(nome)

Diponibilità(negozio, oggetto)
vincolo di integrità referenziale tra Disponibili(oggetto) ed Oggetti(nome)

EffettiAttivi(nome, descrizione, CD, costo di mana, oggetto)
vincolo di integrità referenziale tra EffettiAttivi(oggetto) e Oggetti(nome)

Implementazione dello Schema Logico

Di seguito il codice *mysql* usato per implementare le relazioni ricavate al punto precedente.

Eroi

```
DROP TABLE IF EXISTS Eroi;
CREATE TABLE Eroi (
     nome varchar(20) primary key,
     primary_attribute varchar(3) not null,
     bio varchar(2048),

     base_str float not null,
     base_agl float not null,
     base_int float not null,
     base_atk float not null,

     gain_str float not null,
     gain_agl float not null,
     gain_int float not null,

     curr_lvl tinyint not null,

     bonus_str float not null default 0,
     bonus_agl float not null default 0,
     bonus_int float not null default 0,
     bonus_atk float not null default 0
)ENGINE=InnoDB;
```

Spell

```
DROP TABLE IF EXISTS Spell;
CREATE TABLE Spell (
     nome varchar(20) primary key,
     tasto varchar(1),
     descrizione varchar(512) not null,
     ability_type varchar(20),
     affection varchar(20),
     damage_type varchar(20),
     piercing bool,
     ultimate bool,
     modalita varchar(20),
     effetto varchar(512),

     eroe varchar(20),

     foreign key (eroe) references Eroi(nome) on delete cascade on
          update cascade
)Engine=InnoDB;
```

Dettagli

```
DROP TABLE IF EXISTS SpellDetail;
CREATE TABLE SpellDetail (
     spell varchar(20),
     lvl tinyint,
     CD float,
     mana_cost smallint,
     damage float,

     primary key(spell,lvl),
     foreign key (spell) references Spell(nome) on delete cascade
          on update cascade
)Engine=InnoDB;
```

Oggetti

```
DROP TABLE IF EXISTS Oggetti;
CREATE TABLE Oggetti (
     nome varchar(20) primary key,
     costo  smallint not null,
     descrizione varchar(512),
     tipo varchar(20),

     bonus_str float default 0,
     bonus_agl float default 0,
     bonus_int float default 0,
     bonus_atk float default 0
)ENGINE=InnoDB;
```

Inventario

```
DROP TABLE IF EXISTS Inventario;
CREATE TABLE Inventario (
     eroe varchar(20),
     oggetto varchar(20),

     primary key(eroe,oggetto),
     foreign key (eroe) references Eroi(nome) on delete cascade on
          update cascade,
     foreign key (oggetto) references Oggetti(nome) on delete
          cascade on update cascade
)ENGINE=InnoDB;
```

Disponibilità

```
DROP TABLE IF EXISTS Disponibilita;
CREATE TABLE Disponibilita (
     oggetto varchar(20),
     negozio varchar(20),

     primary key(oggetto,negozio),
     foreign key (oggetto) references Oggetti(nome) on delete
          cascade on update cascade
)ENGINE=InnoDB;
```

EffettiAttivi

```
DROP TABLE IF EXISTS EffettiAttivi;
CREATE TABLE EffettiAttivi (
     nome varchar(20) primary key,
     descrizione varchar(512),
     CD float,
     mana_cost smallint,

     oggetto varchar(20),

     foreign key (oggetto) references Oggetti(nome) on delete
cascade on update cascade
)ENGINE=InnoDB;
```

Query

1) Mostra per l'eroe selezionato il numero di oggetti posseduti ed il loro costo totale

```
select e.nome, count(*) as 'N. Oggetti', sum(o.costo) as 'Net
Worth'
from Eroi e left join Inventario i on e.nome=i.eroe
          left join Oggetti o on i.oggetto=o.nome
where e.nome='$nome_eroe'
group by e.nome;
```

2) Tutti gli oggetti (mostrando nome, costo e descrizione) che hanno effetto attivo, il cui costo di mana sia superiore a ad un certo tot e CD più lungo di tot

```
select o.nome, o.costo, o.descrizione, e.mana_cost, e.CD
from Oggetti o join EffettiAttivi e on o.nome=e.oggetto
where o.nome in (
     select oggetto
     from EffettiAttivi
     where mana_cost>$mana and CD>$calldown
);
```

3) Costo totale di mana di "tutta la rotation di spell" (=tutte le magie una immediatamente dopo l'altra) di ogni eroe che ha solo spell attive, considerando le spell al massimo livello

```
select e.nome as Eroe, sum(d.mana_cost) as 'Rotation Total Cost'
from Eroi e join Spell s on e.nome=s.eroe
          join SpellDetail d on s.nome=d.spell
where ((s.ultimate=0 and d.lvl=4) or (s.ultimate=1 and d.lvl=3))
and e.nome not in (
     select nome
     from Eroi
     where nome in (
          select eroe
          from Spell
          where modalita="Passiva"
     )
)
group by e.nome;
```

4) Eroi la cui spell con CD più alto (a qualsiasi livello) ha CD maggiore dell'oggetto più lento a ricaricarsi posseduto dallo stesso eroe

```
select e.nome as Eroe, max(d.CD) as 'CallDown massimo'
from Eroi e join Spell s on e.nome=s.eroe
          join SpellDetail d on s.nome=d.spell
group by e.nome
having max(d.CD) >ALL(
     select max(CD)
     from EffettiAttivi
     where oggetto in (
          select oggetto
          from Inventario
          where eroe=e.nome
     )
);
```

5) Il negozio che ha venduto ad eroi il cui nome compicia per X il maggior numero di oggetti aventi effetti attivi

```
select d.negozio as Negozio, count(*) as 'Numero Oggetti Venduti'
from Disponibilita d join Oggetti o on d.oggetto=o.nome
                    join Inventario i on o.nome=i.oggetto
where i.eroe like '$x%' and d.oggetto in (
     select oggetto
     from EffettiAttivi
)
group by d.negozio
having count(*)= (
     select max(counted) from (
          select count(*) as counted
               from Disponibilita d2 join Oggetti o2 on
d2.oggetto=o2.nome
                                    join Inventario i2 on
o2.nome=i2.oggetto
               where i2.eroe like '$x%' and d2.oggetto in (
                    select oggetto
                    from EffettiAttivi
               )
               group by d2.negozio
     ) as counts
);
```

6) Eroi, media dei tempi d'attesa delle spell a livello masssimo, solo se i la media di tempi di attesa a livello 1 è più lunga delle spell maxate

```
select e.nome as Eroe, e.primary_attribute as 'Attributo
Primario', avg(d.CD) as 'Media CallDown spell al massimo livello'
from Eroi e join Spell s on e.nome=s.eroe
          join SpellDetail d on s.nome=d.spell
where (s.ultimate=0 and d.lvl=4) or (s.ultimate=1 and d.lvl=3)
group by e.nome, e.primary_attribute
having avg(d.CD)<ALL (
     select avg(d2.CD)
     from Eroi e2 join Spell s2 on e2.nome=s2.eroe
               join SpellDetail d2 on s2.nome=d2.spell
     where e2.nome=e.nome and d2.lvl=1
     group by e2.nome
);
```

Altre query

```
$riep_eroi="
select nome as Nome, primary_attribute as 'Attributo Primario', bio as Storia
from Eroi;
";

function dett_eroi_spell($nome_eroe) {
      $dett_eroi_spell="
      select nome as Nome, tasto as Tasto, modalita as 'Attiva/Passiva',
damage_type as 'Tipo di Danno', descrizione as Descrizione
      from Spell
      where eroe=\"$nome_eroe\";
      ";
      return $dett_eroi_spell;
}


function stat_eroe($nome_eroe) {
      $qr="
      select base_str, base_agl, base_int, base_atk,
            gain_str, gain_agl, gain_int
      from Eroi
      where nome=\"$nome_eroe\";
      ";
      return $qr;
}

function items_eroe($nome_eroe) {
      $qr="
      select oggetto
      from Inventario
      where eroe=\"$nome_eroe\";
      ";
      return $qr;
}

function bonus_stats_eroe($nome_eroe) {
      $qr="
      select bonus_str, bonus_agl,bonus_int,bonus_atk
      from Eroi
      where nome=\"$nome_eroe\";
      ";
      return $qr;
}

function net_worth_eroe($nome_eroe) {
      $qr="
      select count(*) as 'N. Oggetti', sum(o.costo) as 'Net Worth'
      from Eroi e left join Inventario i on e.nome=i.eroe
                        left join Oggetti o on i.oggetto=o.nome
      where e.nome=\"$nome_eroe\"
      group by e.nome;
      ";
      return $qr;
}

function ins_inventario($nome_eroe,$nome_oggetto) {
      $qr="
      insert into Inventario
      values (\"$nome_eroe\", \"$nome_oggetto\");";
      return $qr;
}


function det_spell1($nome_spell) {
            $qr="
            select nome as Nome, tasto as Tasto, descrizione as Descrizione
            from Spell
            where nome=\"$nome_spell\";
            ";
```

```
            return $qr;
      }

      function det_spell2($nome_spell) {
            $qr="
            select ability_type as Ability,affection as Affects,damage_type as
Damage
            from Spell
            where nome=\"$nome_spell\";
            ";
            return $qr;
      }

      function det_spell3($nome_spell) {
            $qr="
            select piercing,ultimate,modalita
            from Spell
            where nome=\"$nome_spell\";
            ";
            return $qr;
      }


      function det_lvl_spell($nome_spell) {
            $qr="
            select sd.lvl as Livello, sd.CD, sd.mana_cost as 'Mana Cost',
sd.damage as Danno
            from Spell s join SpellDetail sd on s.nome=sd.spell
            where s.nome=\"$nome_spell\"
            order by sd.lvl;
            ";
            return $qr;
      }



$riep_oggetti="
select nome as Nome, costo as Costo, descrizione as Descrizione
from Oggetti
";


function det_oggetto1($nome_oggetto) {
      $qr="
      select nome as Nome, costo as Costo, descrizione as Descrizione
      from Oggetti
      where nome=\"$nome_oggetto\";
      ";
      return $qr;
}


function det_oggetto2($nome_oggetto) {
      $qr="
      select bonus_str,bonus_agl,bonus_int,bonus_atk,tipo
      from Oggetti
      where nome=\"$nome_oggetto\";
      ";
      return $qr;
}


function disp_oggetto($nome_oggetto) {
      $qr="
      select negozio
      from Disponibilita
      where oggetto=\"$nome_oggetto\";
      ";
      return $qr;
}
```

```
function hero_has($nome_oggetto) {
	$qr="
	select eroe
	from Inventario
	where oggetto=\"$nome_oggetto\";
	";
	return $qr;
}



function ins_item($n,$c,$d,$t,$bs,$bag,$bi,$bat) {
	$qr="
	insert into Oggetti
	values (\"$n\",\"$c\",\"$d\",\"$t\",\"$bs\",\"$bag\",\"$bi\",\"$bat\");
	";
	return $qr;
}



function ins_hero($n,$p,$b,$bs,$bag,$bi,$bat,$gs,$gag,$gi) {
	$qr="
	insert into Eroi
	values
(\"$n\",\"$p\",\"$b\",\"$bs\",\"$bag\",\"$bi\",\"$bat\",\"$gs\",\"$gag\",\"$gi\"
,\"1\",\"0\",\"0\",\"0\",\"0\");
	";
	return $qr;
}

function ins_spell($n,$t,$d,$at,$af,$dt,$p,$u,$m,$e) {
	$qr="
	insert into Spell
	values
(\"$n\",\"$t\",\"$d\",\"$at\",\"$af\",\"$dt\",\"$p\",\"$u\",\"$m\",\"$e\")
	";
	return $qr;
}

function ins_dettagli_spell($s,$l,$CD,$mc,$d) {
	$qr="
	insert into SpellDetail
	values (\"$s\",\"$l\",\"$CD\",\"$mc\",\"$d\")
	";
	return $qr;
}
```

Trigger

1) Controlla che nella tabella eroi l'attributo primario sia uno tra str, agl e int, sia in update che in insert, in caso contrario genera un errore.
I due trigger chiamano una procedura comune dato che hanno lo stesso compito.

```
drop procedure if exists cpa;
create procedure cpa (pa varchar(3))
BEGIN
     IF not (pa='str' or pa='agl' or pa='int') THEN
          insert into Eroi
          select *
          from Eroi
          limit 1;
     END IF;
END|

drop trigger if exists check_primary_attribute_insert;
create trigger check_primary_attribute_insert
before insert on Eroi
for each row
BEGIN
     call cpa(new.primary_attribute);
END|

drop trigger if exists check_primary_attribute_update;
create trigger check_primary_attribute_update
before update on Eroi
for each row
BEGIN
     call cpa(new.primary_attribute);
END|
```

2) Ad ogni inserimento o rimozione di un oggetto dall'inventario di un determinato eroe questo trigger si occupa di aggiornare eventualmente le statistiche bonus dell'eroe stesso.
Come in qualche patch fa si assume che una volta comprato l'oggetto "*Black King Bar*" esso non sia più vendibile.

```
#INSERIMENTO
drop trigger if exists nuovo_oggetto|
create trigger nuovo_oggetto
after insert on Inventario
for each row
BEGIN
     declare attacco float;
     declare forza float;
     declare agilita float;
     declare intelligenza float;

     select o.bonus_atk into attacco
     from Oggetti o
     where o.nome=new.oggetto;

     select o.bonus_str into forza
     from Oggetti o
     where o.nome=new.oggetto;
```

```
     select o.bonus_agl into agilita
     from Oggetti o
     where o.nome=new.oggetto;
     select o.bonus_int into intelligenza
     from Oggetti o
     where o.nome=new.oggetto;

     update Eroi
     set bonus_atk=bonus_atk+attacco,
          bonus_str=bonus_str+forza,
          bonus_agl=bonus_agl+agilita,
          bonus_int=bonus_int+intelligenza
     where nome=new.eroe;
END|



#RIMOZIONE
drop trigger if exists cancellazione_oggetto|
create trigger cancellazione_oggetto
before delete on Inventario
for each row
BEGIN
     IF old.oggetto="Black King Bar" THEN
          insert into Eroi
          select *
          from Eroi
          limit 1;
     ELSE begin
          declare attacco float;
          declare forza float;
          declare agilita float;
          declare intelligenza float;

          select o.bonus_atk into attacco
          from Oggetti o
          where o.nome=old.oggetto;
          select o.bonus_str into forza
          from Oggetti o
          where o.nome=old.oggetto;
          select o.bonus_agl into agilita
          from Oggetti o
          where o.nome=old.oggetto;
          select o.bonus_int into intelligenza
          from Oggetti o
          where o.nome=old.oggetto;


          update Eroi
          set bonus_atk=bonus_atk-attacco,
               bonus_str=bonus_str-forza,
               bonus_agl=bonus_agl-agilita,
               bonus_int=bonus_int-intelligenza
          where nome=old.eroe;
     end;
     END IF;
END|
```

3) Dato che la cancellazione a cascata non attiva i trigger è stato necessario aggiungere questo controllo per ovviare ad un piccolo bug:
la dichiarazione della tabella “Inventario” contiene la seguente riga di codice:

```
foreign key (oggetto) references Oggetti(nome) on delete
     cascade on update cascade
```

quindi la cancellazione di una riga della tabella Oggetti potrebbe comportare la cancellazione “automatica” di una o più righe della tabella Inventario.
Dato che queste cancellazioni *cascaded* non attivano il trigger numero 2, creerebbero una inconsistenza nei valori bonus dalla tabella Eroi in quanto essi non verrebbero aggiornati dopo la rimozione dell'oggetto dall'inventario.

Per risolvere il problema dei “bonus fantasma” è stato aggiunto questo piccolo trigger che elimina “direttamente” le righe in questione dalla tabella Inventario:

```
drop trigger if exists cascade_emulator_for_items|
create trigger cascade_emulator_for_items
before delete on Oggetti
for each row
BEGIN
     delete from Inventario
     where oggetto=old.nome;
END|
```

Funzioni

1) Tre piccole funzioni che accettano come parametro il nome di un eroe (che è chiave primaria) e una stringa di utilità.
Ognuna delle tre funzioni calcola caratteristiche avanzate dell'eroe in questione relative ad uno specifico attributo; tiene conto anche degli eventuali bonus conferiti dagli oggetti e dall'aumento delle statistiche dovuto all'aumento di livello.
La prima funzione si occupa della *strenght* la seconda dell'*agility* e la terza dell'*intellect*.

a- Accetta come stringa di utilità le stringhe 'hp', 'regen' e vuota.
Ritorna rispettivamente i punti vita totali, la rigenerazione della vita al secondo ed i punti totali di *strenght.*

```
DROP function IF EXISTS `calculate_str`;

DELIMITER $$
CREATE FUNCTION `calculate_str`(hero varchar(20),what varchar(20))
RETURNS int(11)
BEGIN
     declare base float;
     declare gain float;
     declare bonus float;
     declare lvl int;

     select base_str, gain_str, bonus_str, curr_lvl into base,
          gain, bonus, lvl
     from Eroi
     where nome=hero;

     IF what='hp' THEN
          return (base+(gain*lvl)+bonus)*19;
     ELSEIF what='regen' THEN
          return (base+(gain*lvl)+bonus)*0.03;
     ELSE
          return base+(gain*lvl)+bonus;
     END IF;

END$$

DELIMITER ;
```

b- Accetta come stringa di utilità le stringhe 'armor', 'atkspeed', e vuota.
Ritorna rispettivamente i punti armatura totali, la velocità di attacco ed i punti totali di *agility*.

```
DROP function IF EXISTS `calculate_agl`;

DELIMITER $$
CREATE FUNCTION `calculate_agl`(hero varchar(20), what
varchar(20)) RETURNS int(11)
BEGIN
     declare base float;
     declare gain float;
     declare bonus float;
     declare lvl int;
```

```
     select base_agl, gain_agl, bonus_agl, curr_lvl into base,
          gain, bonus, lvl
     from Eroi
     where nome=hero;

          IF what='armor' THEN
          return (base+(gain*lvl)+bonus)*0.14;
     ELSEIF what='atkspeed' THEN
          return (base+(gain*lvl)+bonus)*1;
     ELSE
          return base+(gain*lvl)+bonus;
     END IF;
END$$

DELIMITER ;
```

c- Accetta come stringa di utilità le stringhe 'manapool', 'regen' e vuota.
Ritorna rispettivamente la quantità massima di Mana, la rigenerazione del Mana al secondo ed i punti totali di *intellect*.

```
DROP function IF EXISTS `calculate_int`;

DELIMITER $$
CREATE FUNCTION `calculate_int`(hero varchar(20), what
varchar(20)) RETURNS int(11)
BEGIN
     declare base float;
     declare gain float;
     declare bonus float;
     declare lvl int;

     select base_int, gain_int, bonus_int, curr_lvl into base,
gain, bonus, lvl
     from Eroi
     where nome=hero;

          IF what='manapool' THEN
          return (base+(gain*lvl)+bonus)*13;
     ELSEIF what='regen' THEN
          return (base+(gain*lvl)+bonus)*0.04;
     ELSE
          return base+(gain*lvl)+bonus;
     END IF;
END$$

DELIMITER ;
```

2) Questa funzione si occupa di calcolare l'attacco totale dell'eroe. Esso è influenzato da:
- attacco base
- attacco conferito come bonus dagli oggetti
- punti totali presenti nell'attributo primario dell'eroe

Ad esempio se un eroe forza dovesse vere 100 punti di *strenght* egli guadagnerebbe subito 100 punti di attacco.
Dato che le funzioni presentate al punto 1 si occupano di calcolare, tra le varie cose, anche i punteggi totali delle varie *stats* questa funzione le richiamerà al suo interno.

```
DROP FUNCTION IF EXISTS `calculate_atk`;
DELIMITER $$
CREATE FUNCTION `calculate_atk`(hero varchar(20)) RETURNS int(11)
BEGIN
     declare pr_atr varchar(3);
     declare base float;
     declare bonus float;

     select primary_attribute,base_atk, bonus_atk into pr_atr,
base, bonus
     from Eroi
     where nome=hero;

     IF pr_atr='str' THEN
          return calculate_str(hero,'')+base+bonus;
     ELSEIF pr_atr='agl' THEN
          return calculate_agl(hero,'')+base+bonus;
     ELSE
          return calculate_int(hero,'')+base+bonus;
     END IF;

END$$

DELIMITER ;
```