Basi di Dati A.A. 2019/2020

Progetto Database

Steam

Alberto Ursino, Fabio Marangoni, Marco Mariotto

11/03/2020

Sommario

Introduzione

Descrizione del progetto

Si vuole realizzare una base di dati simile a quella utilizzata da Steam, una piattaforma che si occupa di distribuzione di contenuti digitali (videogiochi e relative espansioni). Per ogni utente che si è registrato a Steam, si vogliono rappresentare i giochi posseduti, le relative informazioni di gioco e le attività. In Steam un utente può interagire anche con altri utenti, ad esempio mandando richieste di amicizia, vendendo oggetti, ecc.

Requisiti strutturati

Per ogni utente, identificato univocamente da un codice Steam (fornito al momento della registrazione e chiamato *steam_id*), rappresentiamo i dati account obbligatori: username, email e password (per ragioni di sicurezza e privacy verrà salvato solo il relativo *hash*) e i facoltativi: nazione, numero di telefono e biografia.
Altri dati che verranno salvati sono: la data di iscrizione, il livello corrente e lo stato di rete (‘online’, ‘offline’).
Ad un utente è associato un portafoglio con relativo saldo Steam (non utilizzabile al di fuori del negozio di Steam per altri acquisti) e una lista di carte di credito/debito collegate all’account.
Ogni utente possiede una libreria di giochi. Per ogni gioco posseduto vogliamo tener traccia delle attività dell’utente come le ore trascorse a giocare, la data di ultimo avvio, il numero di ore giocate nelle ultime due settimane e la data di acquisto.
Per ogni gioco posseduto l’utente può anche acquisire degli achievement e può recensirlo una sola volta con un commento e un voto; si vuole salvare la data di recensione.
Ciascun utente possiede una lista dei desideri a cui aggiunge giochi che ancora non possiede e che desidera avere; si vuole salvare la data in cui il gioco viene aggiunto.

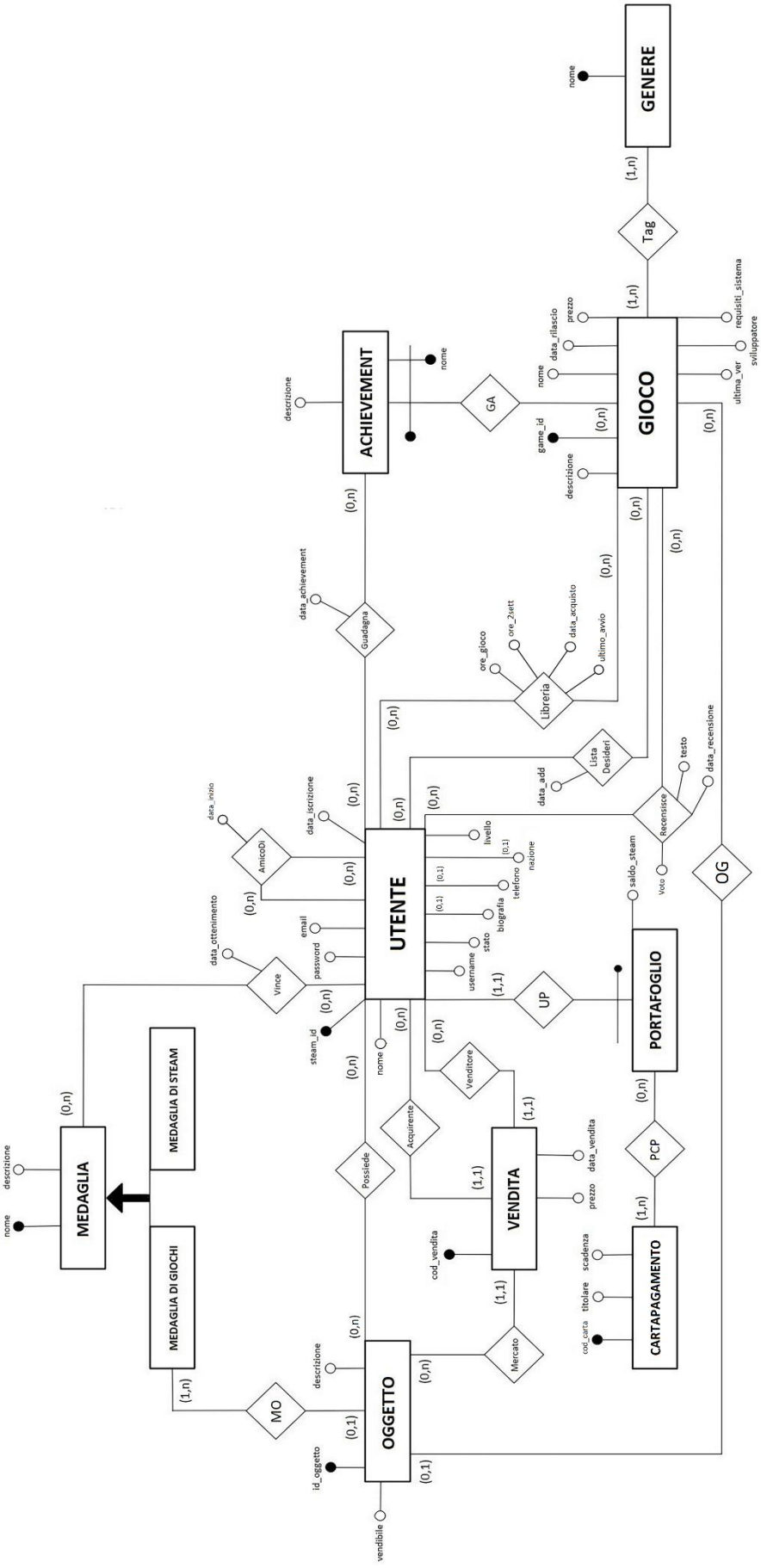
Ogni utente ha un inventario, che può contenere diversi oggetti, per la maggior parte appartenenti a giochi specifici e che si ottengono giocando; alcuni oggetti possono essere venduti nel mercato di Steam ad un prezzo deciso dall'utente. Viene tenuta traccia di ciascuna vendita, e per ciascuna di essa del venditore, dell'acquirente (che sono entrambi utenti), della data di vendita e del relativo prezzo di acquisto.
Un utente può vincere delle medaglie. Per ogni medaglia, identificata dal suo nome, abbiamo una descrizione di come viene ottenuta, in genere collezionando determinati oggetti. Inoltre, una medaglia può appartenere a due categorie: di Steam o di uno specifico gioco. Le medaglie di un gioco si ottengono esclusivamente collezionando oggetti di quel relativo gioco.
Un utente può avere amici; viene tenuta traccia della data di inizio amicizia.

Grazie alla memorizzazione delle date, il database è capace di mostrare le attività di un utente, ad esempio l'acquisto di un gioco in data x, la data in cui stringe amicizia con un dato utente, quando ha guadagnato un certo achievement, ecc.

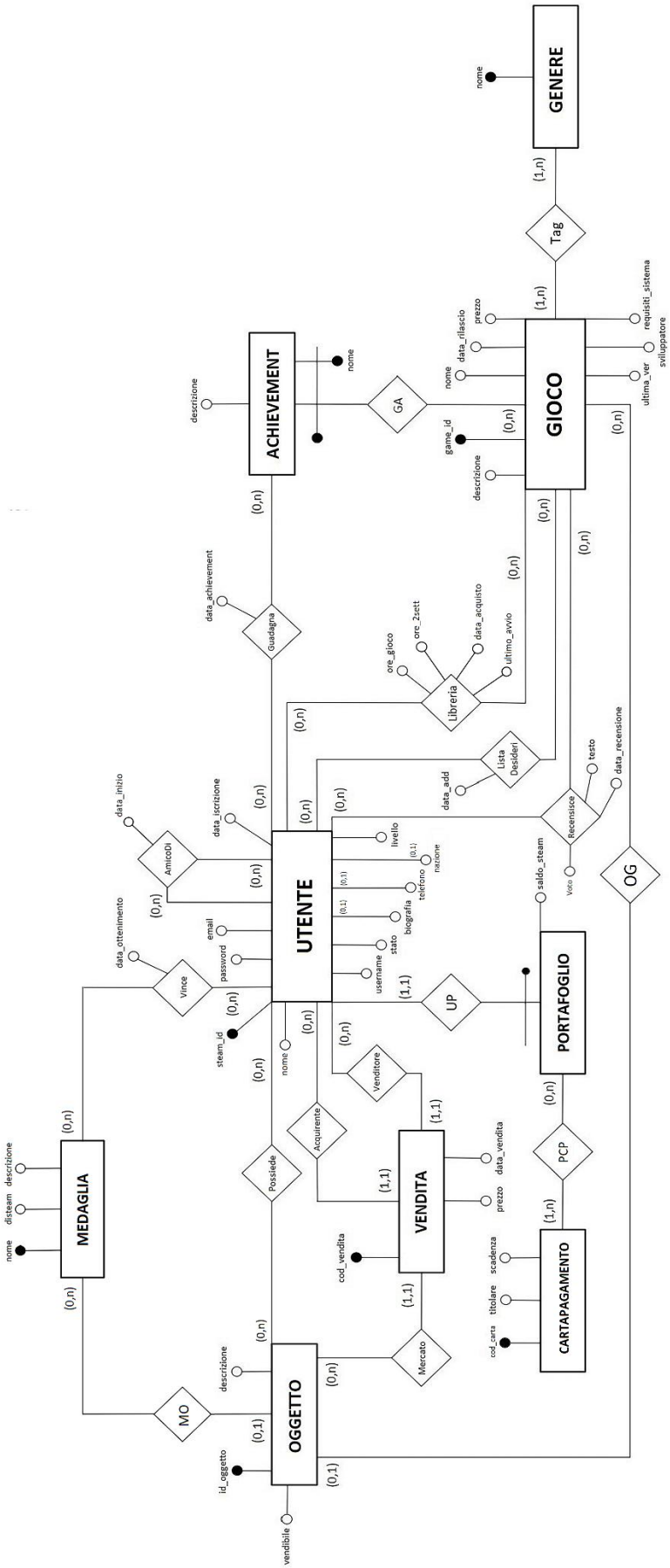
Per ogni gioco, identificato univocamente da un codice (*game_id*), rappresentiamo il nome, una descrizione, i generi a cui appartiene, la data di rilascio, il prezzo corrente, il nome dello sviluppatore, il numero dell'ultima versione e i requisiti di sistema. Ogni gioco può avere degli achievement, che sono ottenibili dopo aver completato in genere delle sfide all'interno del gioco. Per ogni gioco di ogni utente si sa quanti e quali achievement ha ottenuto con la rispettiva data di ottenimento.

Per ogni achievement, identificato dal nome e dal gioco a cui appartiene, si conosce una semplice descrizione di come è ottenuto.

Schema ER


GENERE
nome
Tag
(1,n)
GIOCO
nome
data_rilascio
prezzo
requisiti_sistema
sviluppatore
ultima_ver
game_id
descrizione
(0,n)
ACHIEVEMENT
descrizione
nome
GA
Guadagna
data_achievement
Libreria
ore_gioco
ore_2sett
data_acquisto
ultimo_avvio
Lista Desideri
data_add
Recensisce
testo
data_recensione
voto
OG
UTENTE
livello
nazione
telefono
biografia
stato
username
email
password
data_iscrizione
data_inizio
AmicoDi
Vince
data_ottenimento
steam_id
nome
UP
saldo_steam
PORTAFOGLIO
Venditore
Acquirente
Possiede
VENDITA
data_vendita
prezzo
cod_vendita
PCP
CARTAPAGAMENTO
titolare
scadenza
cod_carta
Mercato
OGGETTO
descrizione
id_oggetto
vendibile
MO
MEDAGLIA
descrizione
nome
MEDAGLIA DI STEAM
MEDAGLIA DI GIOCHI

Schema ER – Ristrutturato



Regole di vincolo schema concettuale

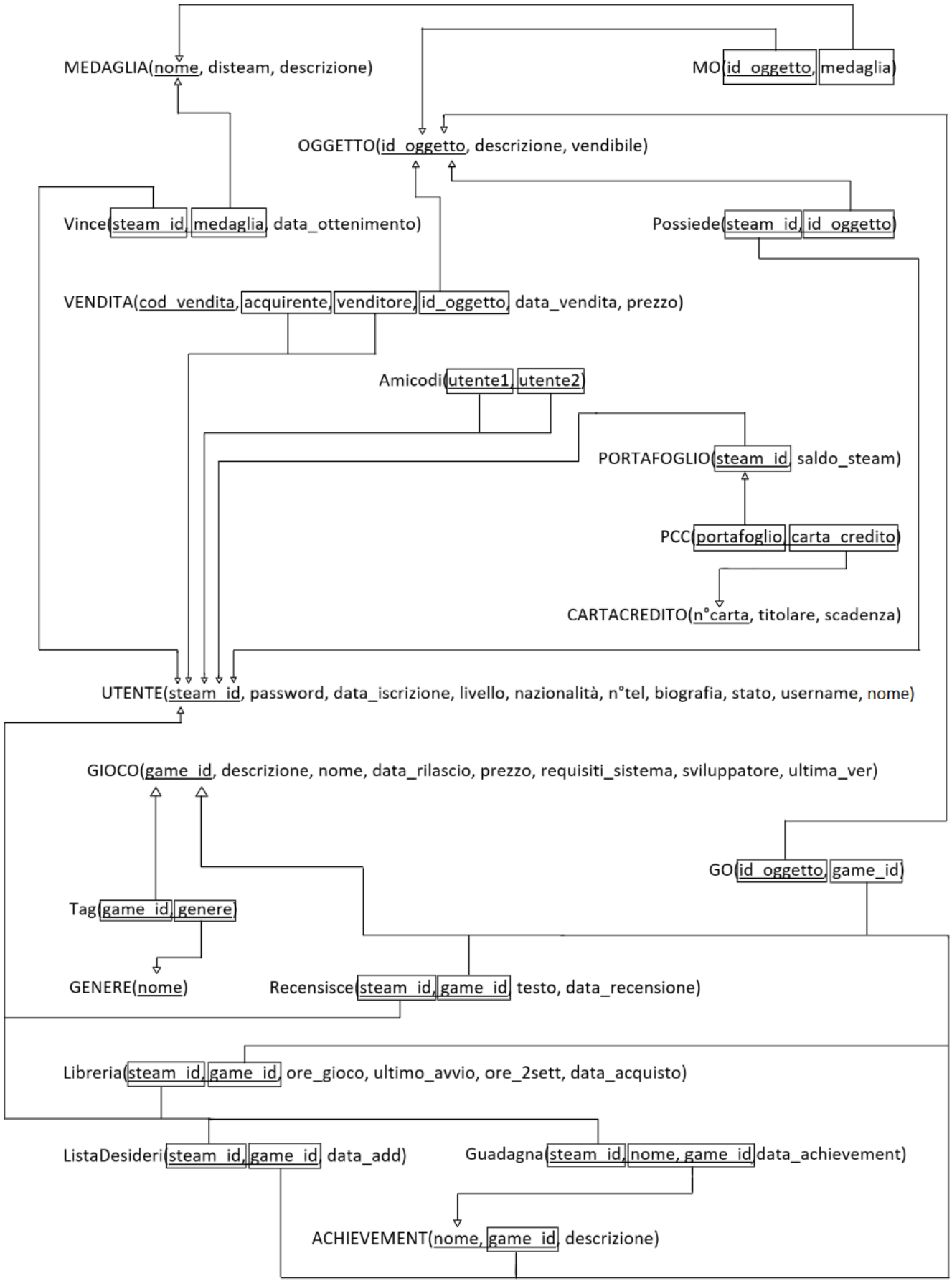
1. Un oggetto non vendibile non può partecipare alla relazione “Mercato”;
2. Un utente non può partecipare alla relazione “Guadagna” se non possiede il gioco relativo a quell’achievement (ovvero per il gioco X che fa parte dell’identificazione dell’achievement, deve esistere un collegamento tra l’utente e X in “Libreria”);
3. Un gioco posseduto da un utente (ovvero che compare nella relazione “Libreria”) non deve comparire nella lista dei desideri, ovvero in “ListaDesideri”;
4. Un utente non può recensire un gioco che non possiede. Ovvero per un dato utente, un gioco può partecipare alla relazione “Recensisce” solo se partecipa alla relazione “Libreria”;
5. Una medaglia di un gioco (e non di Steam) deve essere ottenuta esclusivamente collezionando oggetti appartenenti a quel relativo gioco;
6. Ogni attività di ciascun utente deve essere successiva alla sua data di iscrizione su Steam.

Dizionario dei dati

Entità	Descrizione	Attributi	Identificatore
UTENTE	L'utenza della piattaforma Steam.	steam_id, nome, username, password, email, data_iscrizione, livello, stato, telefono (0,1), biografia (0,1), nazione (0,1)	steam_id
GIOCO	Prodotto software distribuito agli utenti mediante Steam.	game_id, nome, data_rilascio, descrizione, prezzo, requisiti_sistema, sviluppatore, ultima_ver	game_id
ACHIEVEMENT	Obiettivo relativo al progresso in un gioco	nome, descrizione	nome, GIOCO
GENERE	Categoria in cui si distinguono i giochi per tipologia	nome	nome
VENDITA	Record di uno scambio tra due utenti di un oggetto per denaro	cod_vendita, prezzo, data_vendita	cod_vendita
OGGETTO	Elemento dell'inventario di un utente; può essere o meno relativo ad un gioco	id_oggetto, descrizione, vendibile	id_oggetto
MEDAGLIA	Riconoscimento di un traguardo raggiunto mediante semplici attività sulla piattaforma (medaglia di Steam) oppure al completamento di una collezione di oggetti di un gioco	nome, descrizione, disteam	nome
PORTAFOGLIO	Sezione privata del profilo di un utente contenente informazioni su saldo residuo e carte di pagamento collegate all'account	saldo_steam	UTENTE
CARTAPAGAMENTO	Strumento di pagamento che viene registrato nell'account dell'utente	cod_carta, titolare, scadenza	cod_carta

Associazione	Descrizione	Attributi	Entità partecipi
AmicoDi	Relazione di amicizia tra gli utenti	data_inizio	UTENTE(0,n), UTENTE(0,n)
Libreria	Registro dei giochi posseduti dall'utente e del loro tempo di utilizzo	ore_gioco, ultimo_avvio, ore_2sett, data_acquisto	UTENTE(0,n), GIOCO(0,n)
Recensisce	Recensione rilasciata dall'utente a un gioco	testo, data_recensione, voto	UTENTE(0,n), GIOCO(0,n)
ListaDesideri	Registro dei giochi desiderati dall'utente e non ancora posseduti	data_add	UTENTE(0,n), GIOCO(0,n)
Guadagna	Sbloccamento di un achievement da parte dell'utente	data_achievement	UTENTE(0,n), ACHIEVEMENT(0,n)
GA	Relazione tra un achievement ed il gioco a cui riferisce		GIOCO(0,n), ACHIEVEMENT(1,1)
Tag	Etichettatura del gioco secondo i generi di appartenenza		GIOCO(1,n), GENERE(1,n)
UP	Collegamento tra l'utente e il suo portafoglio		UTENTE(1,1), PORTAFOGLIO(1,1)
PCP	Associazione delle carte di pagamento al portafoglio per cui sono state usate		PORTAFOGLIO(0,n), CARTACREDITO(1,n)
Possiede	Relazione rappresentante l'inventario degli oggetti di un utente		UTENTE(0,n), OGGETTO(0,n)
Acquirente	Ottenimento dell'oggetto di scambio nella vendita		UTENTE(0,n), VENDITA(1,1)
Venditore	Cessione dell'oggetto di scambio nella vendita		UTENTE(0,n), VENDITA(1,1)
Mercato	Riferimento dell'oggetto che viene sottoposto alla vendita		VENDITA(1,1), OGGETTO(0,n)
OG	Afferenza di un oggetto al gioco di dominio		GIOCO(0,n), OGGETTO(0,1)
MO	Collegamento di una medaglia agli eventuali oggetti la cui collezione ne permette l'ottenimento		OGGETTO(0,1), MEDAGLIA(0,n)
Vince	Ottenimento della medaglia da parte dell'utente	data_ottenimento	UTENTE(0,n), MEDAGLIA(0,n)

Schema Logico



Regole di vincolo schema logico

1. Un utente deve partecipare ad una e solo un'istanza di UP;
2. Un portafoglio deve partecipare ad una e solo un'istanza di UP;
3. Una carta di pagamento deve partecipare ad almeno un'istanza di PCP;
4. Un oggetto può partecipare al massimo ad un'istanza di MO;
5. Una vendita deve partecipare ad una e solo un'istanza di Acquirente;
6. Una vendita deve partecipare ad una e solo un'istanza di Venditore;
7. Una vendita deve partecipare ad una e solo un'istanza di Mercato;
8. Un gioco deve partecipare ad almeno un'istanza di Tag;
9. Un oggetto può partecipare al massimo ad un'istanza di OG;
10. Un achievement deve partecipare ad una e solo un'istanza di GA;
11. In UTENTE gli attributi *username*, *nome*, *password*, *email*, *data_iscrizione*, *livello* e *stato* non devono essere nulli;
12. In Libreria gli attributi *ore_gioco*, *ore_2sett*, *data_acquisto* non devono essere nulli;
13. In Recensisce gli attributi *testo*, *data_recensione e voto* non devono essere nulli;
14. In ListaDesideri l'attributo *data_add* non deve essere nullo;
15. In GIOCO gli attributi *nome*, *descrizione*, *data_rilascio*, *prezzo*, *requisiti_sistema*, *sviluppatore* e *ultima_ver* non devono essere nulli;
16. In ACHIEVEMENT l'attributo *descrizione* non deve essere nullo;
17. In Guadagna l'attributo *data_achievement* non deve essere nullo;
18. In PORTAFOGLIO l'attributo *saldo_steam* non deve essere nullo;
19. In CARTAPAGAMENTO gli attributi *titolare* e *scadenza* non devono essere nulli;
20. In VENDITA gli attributi *prezzo* e *data_vendita* non devono essere nulli;
21. In OGGETTO gli attributi *descrizione* e *vendibile* non devono essere nulli;
22. In MEDAGLIA gli attributi *descrizione* e *disteam* non devono essere nulli;

23. In Vince l'attributo *data_ottenimento* non deve essere nullo;
24. In AmicoDi l'attributo *data_inizio* non deve essere nullo.

Tabelle SQL

```
CREATE TABLE OGGETTO (

    id_oggetto      INTEGER PRIMARY KEY CHECK (id_oggetto >= 0),
    descrizione     TEXT NOT NULL,
    vendibile       BOOLEAN NOT NULL
);

CREATE TABLE GIOCO (
    game_id               INTEGER PRIMARY KEY CHECK (game_id >= 0),
    nome                  VARCHAR(50) NOT NULL,
    descrizione           TEXT NOT NULL,
    data_rilascio         DATE NOT NULL,
    prezzo                DECIMAL(5,2) CHECK (prezzo >= 0),
    requisiti_sistema     TEXT NOT NULL,
    sviluppatore          VARCHAR(50) NOT NULL,
    ultima_ver            VARCHAR(20) NOT NULL
);

CREATE TABLE UTENTE (
    steam_id            INTEGER PRIMARY KEY CHECK (steam_id >= 0),
    nome                VARCHAR(40) NOT NULL,
    username            VARCHAR(40) NOT NULL,
    email               VARCHAR(50) NOT NULL,
    password            CHAR(64) NOT NULL, /*hex string composed of 64 chars for
                                        a 256-bit hash value*/
    biografia           TEXT,
    stato               VARCHAR(7) CHECK (stato = 'online' OR stato =
'offline'),
    telefono            VARCHAR(15),
    nazione             VARCHAR(30),
    data_iscrizione     DATE NOT NULL,
    livello             SMALLINT DEFAULT 0 CHECK (livello >= 0)
);

CREATE TABLE MEDAGLIA (
    nome            VARCHAR(50) PRIMARY KEY,
    disteam         BOOLEAN NOT NULL,
    descrizione     TEXT NOT NULL
);

CREATE TABLE GENERE (
    nome     VARCHAR(50) PRIMARY KEY
);

CREATE TABLE CARTAPAGAMENTO (
    cod_carta     VARCHAR(16) PRIMARY KEY, /*VISA and MASTERCARD have 16
digits*/
    titolare      VARCHAR(50) NOT NULL,
    scadenza      DATE NOT NULL
);

CREATE TABLE MO (
    id_oggetto     INTEGER PRIMARY KEY,
    medaglia       VARCHAR(50) NOT NULL,
    FOREIGN KEY (id_oggetto) REFERENCES OGGETTO (id_oggetto)
        ON UPDATE CASCADE,
    FOREIGN KEY (medaglia) REFERENCES MEDAGLIA (nome)
        ON UPDATE CASCADE
);
```

```sql
CREATE TABLE Vince (
    steam_id            INTEGER,
    medaglia            VARCHAR(50),
    data_ottenimento    DATE NOT NULL,
    PRIMARY KEY (steam_id, medaglia),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (medaglia) REFERENCES MEDAGLIA (nome)
        ON UPDATE CASCADE
);

CREATE TABLE Possiede (
    steam_id       INTEGER,
    id_oggetto     INTEGER,
    PRIMARY KEY (steam_id, id_oggetto),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (id_oggetto) REFERENCES OGGETTO (id_oggetto)
        ON UPDATE CASCADE
);

CREATE TABLE Tag (
    game_id     INTEGER,
    genere      VARCHAR(50),
    PRIMARY KEY (game_id, genere),
    FOREIGN KEY (game_id) REFERENCES GIOCO (game_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (genere) REFERENCES GENERE (nome)
        ON UPDATE CASCADE
);

CREATE TABLE AmicoDi (
    utente1         INTEGER,
    utente2         INTEGER,
    data_inizio     DATE NOT NULL,
    PRIMARY KEY (utente1, utente2),
    FOREIGN KEY (utente1) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (utente2) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE
    CHECK utente1 < utente2 /* Per evitare tuple doppie */
);

CREATE TABLE VENDITA (
    cod_vendita      INTEGER CHECK (cod_vendita >= 0),
    acquirente       INTEGER NOT NULL,
    venditore        INTEGER NOT NULL,
    id_oggetto       INTEGER NOT NULL,
    data_vendita     DATE NOT NULL,
    prezzo           DECIMAL(6,2) CHECK (prezzo >= 0) NOT NULL,
    PRIMARY KEY (cod_vendita),
    FOREIGN KEY (acquirente) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (venditore) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (id_oggetto) REFERENCES OGGETTO (id_oggetto)
        ON UPDATE CASCADE,
    CHECK acquirente != venditore
);
```

```sql
CREATE TABLE PORTAFOGLIO (
    steam_id            INTEGER PRIMARY KEY,
    saldo_steam     DECIMAL(5,2) DEFAULT 0 CHECK (saldo_steam >= 0),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE
);

CREATE TABLE PCP (
    portafoglio     INTEGER,
    carta           VARCHAR(16),
    PRIMARY KEY (portafoglio, carta),
    FOREIGN KEY (portafoglio) REFERENCES PORTAFOGLIO (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (carta) REFERENCES CARTAPAGAMENTO (cod_carta)
        ON UPDATE CASCADE
);

CREATE TABLE OG (
    id_oggetto      INTEGER PRIMARY KEY,
    game_id         INTEGER NOT NULL,
    FOREIGN KEY (id_oggetto) REFERENCES OGGETTO (id_oggetto)
        ON UPDATE CASCADE,
    FOREIGN KEY (game_id) REFERENCES GIOCO (game_id)
        ON UPDATE CASCADE
);

CREATE TABLE Recensisce (
    steam_id            INTEGER,
    game_id             INTEGER,
    testo               TEXT NOT NULL,
    data_recensione     DATE NOT NULL,
    voto                INTEGER CHECK (voto >= 0 AND voto <= 10),
    PRIMARY KEY (steam_id, game_id),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (game_id) REFERENCES GIOCO (game_id)
        ON UPDATE CASCADE
);

CREATE TABLE ACHIEVEMENT (
    nome            VARCHAR(50),
    game_id         INTEGER,
    descrizione     TEXT NOT NULL,
    PRIMARY KEY (nome, game_id),
    FOREIGN KEY (game_id) REFERENCES GIOCO (game_id)
        ON UPDATE CASCADE
);

CREATE TABLE Guadagna (
    steam_id            INTEGER,
    nome                VARCHAR(50),
    game_id             INTEGER,
    data_achievement    DATE NOT NULL,
    PRIMARY KEY (steam_id, nome, game_id),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (nome, game_id) REFERENCES ACHIEVEMENT (nome, game_id)
        ON UPDATE CASCADE
);
```

```sql
CREATE TABLE ListaDesideri (
    steam_id     INTEGER,
    game_id      INTEGER,
    data_add     DATE NOT NULL,
    PRIMARY KEY (steam_id, game_id),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (game_id) REFERENCES GIOCO (game_id)
        ON UPDATE CASCADE
);

CREATE TABLE Libreria (
    steam_id          INTEGER,
    game_id           INTEGER,
    ore_2sett         INTEGER CHECK (ore_2sett >= 0),
    ore_gioco         INTEGER CHECK (ore_gioco >= 0),
    ultimo_avvio      DATE,
    data_acquisto     DATE NOT NULL,
    PRIMARY KEY (steam_id, game_id),
    FOREIGN KEY (steam_id) REFERENCES UTENTE (steam_id)
        ON UPDATE CASCADE,
    FOREIGN KEY (game_id) REFERENCES GIOCO (game_id)
        ON UPDATE CASCADE
);
```

Query

```
/* 1. Per un dato utente (steam_id) e un dato gioco (game_id),
      mostrare gli username degli amici dell'utente che possiedono tale gioco.
*/
SELECT steam_id, username FROM(
    SELECT (CASE
            WHEN utente1 = 17 THEN utente2
            WHEN utente2 = 17 THEN utente1
            END) as steam_id
    FROM amicodi
    WHERE utente1 = 17 or utente2 = 17) AS X
    NATURAL JOIN Libreria NATURAL JOIN UTENTE
    WHERE game_id = 10;


/* 2. Mostrare una classifica dei primi 10 giochi in base alla media dei voti.
*/
SELECT game_id, nome, round(AVG(voto),2) as mediaVoti
FROM GIOCO NATURAL JOIN Recensisce
GROUP BY game_id
ORDER BY mediaVoti DESC
LIMIT 10;


/* 3. Mostrare la classifica dei primi 10 giochi più scaricati. */
DROP VIEW IF EXISTS NUM_DOWNLOAD;

CREATE VIEW NUM_DOWNLOAD AS
    SELECT game_id, nome, count(*) AS num_download
    FROM Libreria NATURAL JOIN GIOCO
    GROUP BY game_id, nome;

SELECT game_id, nome, num_download
FROM GIOCO NATURAL JOIN NUM_DOWNLOAD
ORDER BY num_download DESC
LIMIT 10;


/* 4. Per un dato utente (steam_id), mostrare la percentuale di
      achievement ottenuti di ogni gioco posseduto. */
SELECT game_id, nome, round(ottenuti::numeric/totali, 2) * 100 AS
percentuale_completamento
FROM (( SELECT game_id, count(*) AS totali
       FROM ACHIEVEMENT NATURAL JOIN (
            SELECT game_id
            FROM Libreria
            WHERE steam_id = 1 ) AS X
       GROUP BY game_id ) AS Y NATURAL JOIN
     ( SELECT game_id, count(*) AS ottenuti
       FROM Guadagna
       WHERE steam_id = 1
       GROUP BY game_id ) AS Z) NATURAL JOIN GIOCO;
```

```sql
/* 5. Per un dato utente (steam_id), mostrare un sottoinsieme (di cardinalità 5)
      di giochi non posseduti che potrebbero interessare all'utente.
      I giochi consigliati sono scelti se hanno in comune almeno 1 genere
      con il gioco più giocato dall'utente;
      di questi si preferisce mostrare quelli che hanno più generi in comune
      e con più download. */
DROP VIEW IF EXISTS TAGS;
DROP VIEW IF EXISTS GIOCHI_POSSEDUTI;
DROP VIEW IF EXISTS NUM_DOWNLOAD;

CREATE VIEW NUM_DOWNLOAD AS
    SELECT game_id, nome, count(*) AS num_download
    FROM Libreria NATURAL JOIN GIOCO
    GROUP BY game_id, nome;

CREATE VIEW GIOCHI_POSSEDUTI AS
    SELECT game_id, ore_gioco
    FROM Libreria
    WHERE steam_id = 1;

CREATE VIEW TAGS AS
    SELECT genere
    FROM Tag NATURAL JOIN (
        SELECT game_id
        FROM GIOCHI_POSSEDUTI
        WHERE ore_gioco = ( SELECT MAX(ore_gioco) FROM GIOCHI_POSSEDUTI )) AS X;

SELECT game_id, nome, generi_in_comune, num_download
FROM ( SELECT game_id, count(*) AS generi_in_comune
       FROM Tag NATURAL JOIN (
           SELECT game_id
           FROM GIOCO EXCEPT (SELECT game_id FROM GIOCHI_POSSEDUTI)) AS X
       WHERE genere IN ( SELECT genere FROM TAGS )
       GROUP BY game_id ) AS Y
       NATURAL JOIN NUM_DOWNLOAD
ORDER BY generi_in_comune DESC, num_download DESC
LIMIT 5;


/* 6. Per un dato utente X (steam_id), mostrare i giochi contenuti nella lista
dei desideri
      di ogni suo amico che, con l'attuale saldo steam, X ha la possibilità di
regalargli. */
DROP VIEW IF EXISTS FRIENDS;

CREATE VIEW FRIENDS AS
    SELECT (CASE
            WHEN utente1 = 7 THEN utente2
            WHEN utente2 = 7 THEN utente1
            END) as steam_id
    FROM amicodi
    WHERE utente1 = 7 or utente2 = 7;

SELECT game_id, nome, prezzo
FROM (FRIENDS NATURAL JOIN listadesideri) NATURAL JOIN GIOCO
WHERE prezzo <= (SELECT saldo_steam FROM PORTAFOGLIO WHERE steam_id = 7);
```

```sql
/* 7. Dato un oggetto (id_oggetto), mostrare quante volte è stato venduto negli
ultimi X giorni. */
SELECT id_oggetto, COUNT(*) AS num_vendite
FROM VENDITA
WHERE id_oggetto = 37 AND
    (SELECT NOW()::DATE) - data_vendita <= 2
GROUP BY id_oggetto;


/* 8. Mostrare tutti gli scambi avvenuti tra due dati utenti, X e Y (steam_id),
in ordine cronologico.
      Mostrare inoltre a quale gioco appartiene ogni oggetto scambiato. */
SELECT id_oggetto, data_vendita, prezzo, game_id, nome
FROM (vendita NATURAL LEFT JOIN OG) AS X NATURAL LEFT JOIN GIOCO
WHERE (acquirente = 2 AND venditore = 7) OR (acquirente = 7 AND venditore = 2)
ORDER BY data_vendita;


/* 9. Dato un utente (steam_id), per ogni gioco posseduto,
      mostrare quali oggetti collezionabili mancano
      per vincere la medaglia del gioco. */
SELECT game_id, id_oggetto, medaglia
FROM OG NATURAL JOIN MO
WHERE game_id IN (SELECT game_id
                  FROM Libreria
                  WHERE steam_id = 2)
EXCEPT
SELECT game_id, id_oggetto, medaglia
FROM Possiede NATURAL JOIN MO NATURAL JOIN OG
WHERE steam_id = 2
ORDER BY game_id;
```