PROGETTO BASI DI DATI

DATABASE PER LA GESTIONE DI PARTITE DEL GIOCO DA TAVOLA “RisiKo!”

Andrea Grossi

PARTE 1: MySQL

1. Introduzione

RisiKo! è la versione italiana del celebre gioco da tavola strategico *Risk*. La prima edizione risale al 1968 e da allora ha sempre avuto un largo successo, introducendo nuove versioni nel corso degli anni.

Lo scopo del gioco è cercare di raggiungere il proprio obiettivo (indicato su una carta distribuita ad ogni giocatore ad inizio partita) controllando dei territori con dei "carri armati", rinforzandosi e attaccando territori degli altri giocatori confinanti con i propri. È possibile trovare il regolamento completo al link http://www.risiko.it/rol/risiko_classico.pdf.

L'idea è la realizzazione di un database in grado di memorizzare informazioni riguardo diverse partite di *RisiKo!* per una versione digitale ed online del gioco per permettere a giocatori provenienti da tutto il mondo di sfidarsi in appassionanti partite online.

Le partite sulla piattaforma seguiranno le regole "Torneo" nelle quali la condizione di vittoria per ogni giocatore è la conquista dei territori indicati sulla propria carta obiettivo.

2. Analisi dei requisiti

Il database deve essere in grado di archiviare informazioni riguardo agli utenti che utilizzano la piattaforma (dati di accesso, nazionalità, lingua, amici), sul gioco (territori che compongono la plancia di gioco con relativi continenti e confini) e sulle partite giocate (territori occupati con relative truppe, carte possedute, obiettivi, ecc…).

Nello specifico, devono essere memorizzate le seguenti informazioni riguardo il gioco:

1. **continenti** che compongono la plancia, con il numero di armate bonus associato;
2. **territori**, con il *continente di appartenenza*, i *territori confinanti* e il relativo punteggio*;*
3. **carte**, con il relativo tipo (Fante, Cannone, Cavaliere, Jolly);
4. **carte obiettivo**, con la *lista dei territori da conquistare*.

Per ogni **utente** che si iscrive alla piattaforma devono essere memorizzate le seguenti informazioni:

1. username;
2. password;
3. data di nascita;
4. *nazionalità*;
5. *lingua*;
6. altri giocatori iscritti alla piattaforma nella sua *lista amici*.

Alle **nazioni** *deve essere associata la lista delle* ***lingue parlate*** in una data nazione.

Devono poi essere memorizzate le **partite** giocate con le seguenti informazioni associate:

1. ***giocatori*** *che partecipano alla partita*;
2. *territori occupati* da ogni giocatore, con il relativo numero di armate occupanti;
3. *carte possedute* da ogni giocatore;
4. *obiettivo di ogni giocatore*.

Il database inoltre deve garantire l'integrità dei dati e il rispetto delle seguenti regole di gioco:

V1	Ogni territorio in una partita è occupato da un solo giocatore
V2	Ogni territorio è occupato da un numero di armate compreso tra 1 e 10
V3	Ogni carta può essere posseduta da un solo giocatore alla volta durante una partita
V4	In una partita ogni giocatore può possedere contemporaneamente al massimo 5 carte

2.1 Entità

"Territorio", "Continente", "Carta" e "Carta obiettivo" (da adesso solamente "Obiettivo") sono entità appartenenti al dominio del gioco.

L'entità "Territorio" serve a rappresentare i territori presenti sulla plancia di gioco. Gli attributi di questa entità sono il nome del territorio, che la identifica univocamente, e un numero intero che rappresenta i punti che vengono assegnati al giocatore che possiede il territorio per stimare il suo punteggio nel corso della partita.

L'entità "Continente" serve a rappresentare i continenti ai quali appartengono i territori presenti sulla plancia di gioco. Gli attributi di questa entità sono il nome del continente, che la identifica univocamente, e il numero di truppe bonus che spettano al giocatore che all'inizio del suo turno possiede tutti i territori che compongono il continente.

L'entità "Carta" serve a rappresentare le carte che vengono usate per ottenere delle truppe bonus durante il gioco. Gli attributi di questa entità sono un id numerico, che la identifica univocamente, e il simbolo riportato sulla carta che può essere un fante, un cavaliere, un cannone o un jolly.

L'entità "Obiettivo" serve a rappresentare le carte obiettivo, viene assegnata una carta obiettivo ad ogni giocatore all'inizio della partita e sulla carta sono riportate le condizioni di vittoria. L'attributo di questa entità è un id numerico, che la identifica univocamente.

"Utente" è l'entità predisposta a rappresentare i giocatori che si iscrivono alla piattaforma, "Nazione" permette di individuare la nazionalità di appartenenza degli utenti e "Lingua" rappresenta le lingue in accordo con lo standard ISO 639-1.

L'entità "Utente" ha come attributi lo username dell'utente, che la identifica univocamente, la password che viene utilizzata nel momento dell'accesso alla piattaforma e la data di nascita.

L'entità "Nazione" ha come attributo il nome della nazione, che la identifica univocamente.

L'entità "Lingua" ha come attributi il codice della lingua secondo lo standard ISO 639-1, che la identifica univocamente, e il nome della lingua.

"Partita" è l'entità che individua le partite che si svolgono sulla piattaforma. Ha come attributo un id numerico che la identifica univocamente.

L'entità "Giocatore"[1] individua i partecipanti ad una partita, ha come attributi l'username dell'"Utente" e l'id numerico della partita ed è identificato univocamente da questa coppia di attributi.

[1] Appendice 1: Giocatore, reificazione della relazione Utente-Partita (*reificazione di attributo di relazione*, pag. 246)

2.2 Relazioni

Le relazioni “Appartiene”, “Confina” e “Comprende” appartengono al dominio del gioco.

“Appartiene” mette in relazione un territorio al continente al quale appartiene. Ogni territorio appartiene ad un solo continente mentre ogni continente contiene più territori *(es. l’Oceania contiene 4 territori)*.

“Confina” è una relazione ricorsiva e simmetrica che mette in relazione due territori. Ogni territorio confina con almeno un altro territorio.

“Comprende” mette in relazione un obiettivo con la lista dei territori che lo compongono. Ogni obiettivo è composto da diversi territori e tutti i territori compaiono in almeno un obiettivo.

Le relazioni “Amico”, “Nazionalità” e “Parla” forniscono informazioni riguardo gli utenti.

“Amico” è una relazione ricorsiva e simmetrica che mette in relazione due utenti. Un utente può avere un qualsiasi numero di amici.

“Nazionalità” mette in relazione un utente con la nazione di residenza. Ogni utente ha associata una sola nazione mentre ogni nazione può essere la nazionalità di un qualsiasi numero di utenti.

“Parla” mette in relazione un utente con la lingua che ha indicato. Ad ogni utente è associata una sola lingua (se l’utente non sceglie esplicitamente una lingua viene scelta la lingua principale della nazione di residenza), ogni lingua può essere parlata da un qualsiasi numero di utenti.

“Lingue” mette in relazione una nazione con una lingua. Ad ogni nazione possono essere associate più lingue *(es. in Svizzera si parla Italiano, Tedesco, Francese)* e ad ogni lingua possono essere legate più nazioni *(es. Italiano parlato sia in Italia sia in Svizzera)*. La relazione ha un attributo booleano che indica se la lingua è quella più usata nella nazione.

Le relazioni “U-G” e “G-P” sono introdotte per la reificazione del concetto di “Giocatore”. Un giocatore è individuato univocamente dalla coppia username dell’utente e id della partita.

La relazione “U-G” mette in relazione un giocatore con un utente (un giocatore è un utente che gioca una data partita). Ogni giocatore ha un solo utente al quale si riferisce mentre un utente può essere collegato ad un numero qualsiasi di giocatori (a seconda di quante partite sta giocando/ha giocato).

La relazione “G-P” mette in relazione un giocatore con la partita che sta giocando. Ogni giocatore è associato ad una sola partita mentre ogni partita è associata ad un numero di giocatori che va da 3 a 6 (come indicato dalle regole del gioco).

Le relazioni “Ha”, “Occupa” e “Possiede” forniscono informazioni riguardo lo “stato” di un giocatore in una partita.

La relazione “Ha” mette in relazione un giocatore con il suo obiettivo. Ogni giocatore ha solo un obiettivo nella sua partita mentre ogni obiettivo può essere legato ad un numero qualsiasi di giocatore (purché non siano giocatori della stessa partita).

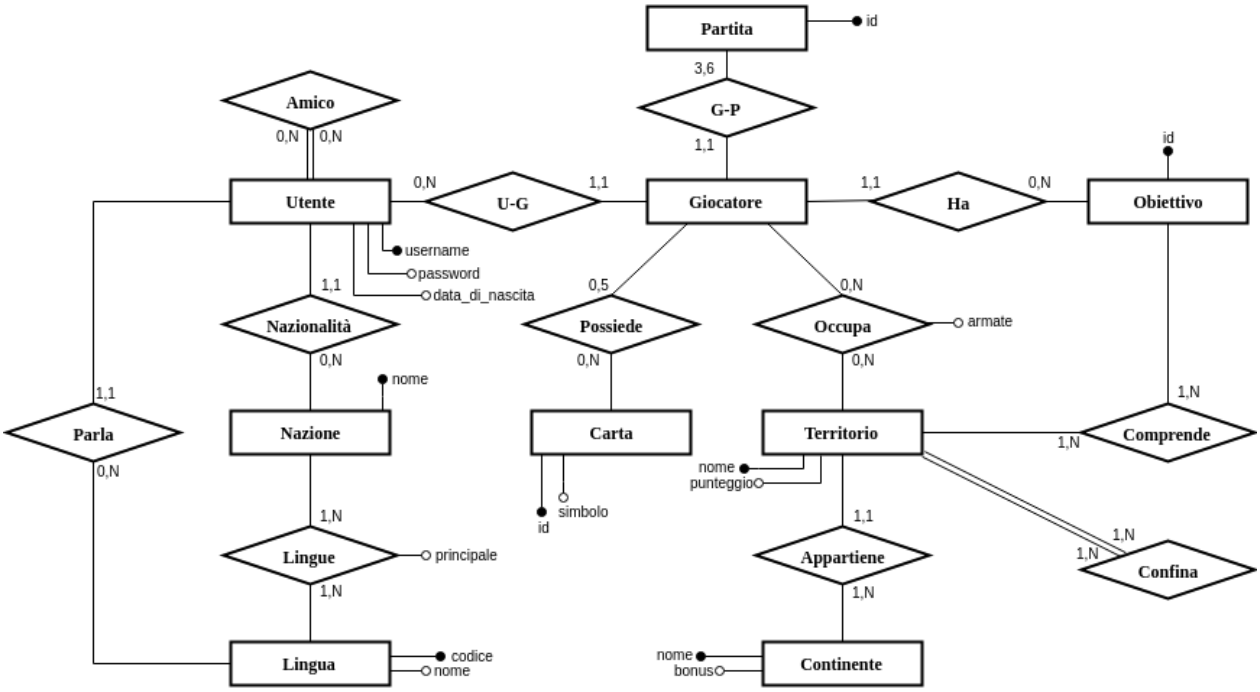
La relazione “Occupa” mette in relazione un giocatore con i territori che occupa. La relazione ha come attributo un intero uguale o maggiore di uno che indica il numero di armate

con cui il giocatore occupa un territorio. Un giocatore può occupare un numero qualsiasi di territori, un territorio può essere occupato da un numero qualsiasi di giocatori (purché non siano giocatori della stessa partita).

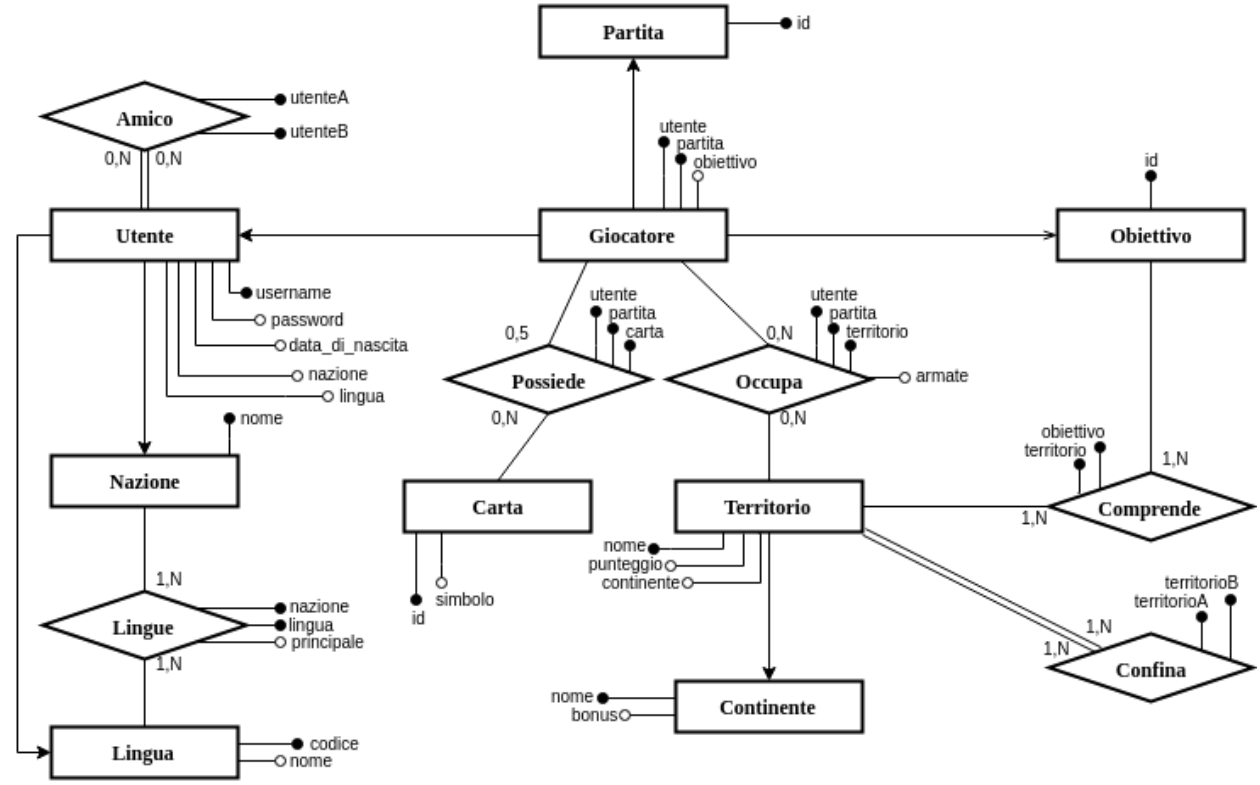
La relazione “Possiede” mette in relazione un giocatore con le carte che possiede. Un giocatore può possedere da nessuna fino ad un massimo di 5 carte (in quel caso è costretto ha giocare la combinazione che ha in mano), una carta può essere posseduta da più giocatori contemporaneamente (purché non siano giocatori della stessa partita).

3. Schema ER

3.1 Schema ER logico



3.2 Schema ER fisico



4. Tabelle

4.1 Partita

Creazione tabella

```
CREATE TABLE Partita (
        id INTEGER NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (id)
);
```

Inserimenti

```
INSERT INTO Partita VALUES (0);
...
```

Esempio

```
SELECT * FROM Partita ORDER BY RAND() LIMIT 5;
+-----+
| id  |
+-----+
| 100 |
|  67 |
|  78 |
|  99 |
|  30 |
+-----+
```

4.2 Obiettivo

Creazione tabella

```
CREATE TABLE Obiettivo (
        id INTEGER NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (id)
);
```

Inserimenti

```
INSERT INTO Obiettivo VALUES (0);
…
```

Esempio

```
SELECT * FROM Obiettivo ORDER BY RAND() LIMIT 5;
+----+
| id |
+----+
| 13 |
| 15 |
|  4 |
| 11 |
|  3 |
+----+
```

4.3 Carta

Creazione tabella

```
CREATE TABLE Carta (
        id INTEGER NOT NULL AUTO_INCREMENT,
        simbolo ENUM('fante','cannone','cavaliere','jolly') NOT NULL,
        PRIMARY KEY (id)
);
```

Inserimento

```
INSERT INTO Carta VALUES (0,'fante');
INSERT INTO Carta VALUES (0,'cannone');
INSERT INTO Carta VALUES (0,'cavaliere');
INSERT INTO Carta VALUES (0,'jolly');
…
```

Esempio

```
SELECT * FROM Carta ORDER BY RAND() LIMIT 5;
+----+-----------+
| id | simbolo   |
+----+-----------+
| 33 | cannone   |
| 12 | fante     |
|  1 | fante     |
|  9 | fante     |
| 25 | cavaliere |
+----+-----------+
```

4.4 Continente

Creazione tabella

```
CREATE TABLE Continente (
        nome VARCHAR(20) NOT NULL,
        bonus INTEGER NOT NULL,
        PRIMARY KEY (nome)
);
```

Inserimento

```
INSERT INTO Continente VALUES ('Africa', 3);
INSERT INTO Continente VALUES ('America del Nord', 5);
INSERT INTO Continente VALUES ('America del Sud', 2);
INSERT INTO Continente VALUES ('Asia', 7);
INSERT INTO Continente VALUES ('Europa', 5);
INSERT INTO Continente VALUES ('Oceania', 2);
```

Esempio

SELECT * FROM Continente ORDER BY RAND() LIMIT 5;

```
+-----------------+-------+
| nome            | bonus |
+-----------------+-------+
| Oceania         |     2 |
| Africa          |     3 |
| America del Sud |     2 |
| Asia            |     7 |
| Europa          |     5 |
+-----------------+-------+
```

4.5 Territorio

Creazione tabella

CREATE TABLE **Territorio** (
    nome VARCHAR(25) NOT NULL,
    punteggio INTEGER NOT NULL,
    continente VARCHAR(20) NOT NULL,
    PRIMARY KEY (nome),
    FOREIGN KEY (continente)
        REFERENCES Continente (nome)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);

Inserimento

INSERT INTO Territorio VALUES ('Medio Oriente',6,'Asia');
INSERT INTO Territorio VALUES ('Siberia',5,'Asia');
INSERT INTO Territorio VALUES ('Africa del Sud',3,'Africa');
INSERT INTO Territorio VALUES ('Brasile',4,'America del Sud');
INSERT INTO Territorio VALUES ('Stati Uniti Orientali',4,'America del Nord');
...

Esempio

SELECT * FROM Territorio ORDER BY RAND() LIMIT 5;

```
+---------------+-----------+------------------+
| nome          | punteggio | continente       |
+---------------+-----------+------------------+
| Medio Oriente |         6 | Asia             |
| Madagascar    |         2 | Africa           |
| Alberta       |         4 | America del Nord |
| Kamchatka     |         5 | Asia             |
| Gran Bretagna |         4 | Europa           |
+---------------+-----------+------------------+
```

4.6 Lingua

Creazione tabella

```
CREATE TABLE Lingua (
        codice CHAR(2) NOT NULL,
        nome VARCHAR(25) NOT NULL,
        PRIMARY KEY (codice)
);
```

Inserimento

```
INSERT INTO Lingua VALUES('en','English');
INSERT INTO Lingua VALUES('aa','Afar');
INSERT INTO Lingua VALUES('ab','Abkhazian');
INSERT INTO Lingua VALUES('af','Afrikaans');
INSERT INTO Lingua VALUES('am','Amharic');
...
```

Esempio

```
SELECT * FROM Lingua ORDER BY RAND() LIMIT 5;
+--------+-----------+
| codice | nome      |
+--------+-----------+
| sg     | Sangro    |
| yo     | Yoruba    |
| mt     | Maltese   |
| fa     | Persian   |
| km     | Cambodian |
+--------+-----------+
```

4.7 Nazione

Creazione tabella

```
CREATE TABLE Nazione (
        nome VARCHAR(50) NOT NULL,
        PRIMARY KEY (nome)
);
```

Inserimento

```
INSERT INTO Nazione VALUES ('Afghanistan');
INSERT INTO Nazione VALUES ('Albania');
INSERT INTO Nazione VALUES ('Algeria');
INSERT INTO Nazione VALUES ('American Samoa');
INSERT INTO Nazione VALUES ('Andorra');
```

Esempio

SELECT * FROM Nazione ORDER BY RAND() LIMIT 5;

```
+----------------------+
| nome                 |
+----------------------+
| Bolivia              |
| Armenia              |
| Jamaica              |
| Taiwan               |
| United Arab Emirates |
+----------------------+
```

4.8 Utente

Creazione tabella

```
CREATE TABLE Utente (
        username VARCHAR(25) NOT NULL,
        password VARCHAR(25) NOT NULL,
        data_di_nascita DATE,
        nazione VARCHAR(50) NOT NULL,
        lingua CHAR(2) NOT NULL DEFAULT 'en',
        PRIMARY KEY (username),
        FOREIGN KEY (nazione)
                REFERENCES Nazione (nome)
                ON UPDATE CASCADE,
        FOREIGN KEY (lingua)
                REFERENCES Lingua (codice)
                ON UPDATE CASCADE
);
```

Inserimento

Inserimento di 1707 record da programma Java.

```
INSERT INTO Utente VALUES ('alefenix75','2v913jnlf8','1975-11-13','Italy','it');
INSERT INTO Utente VALUES ('cern71','yjy4dbe29enjzts36','1971-12-13','Cape Verde','en');
INSERT INTO Utente VALUES ('mathilde50','ia9r3lzwwlptk5zkr9h11','1950-08-14','Egypt', 'ar');
INSERT INTO Utente VALUES ('antonia47','hfqonuih4fnz','1947-09-21','Italy','it');
INSERT INTO Utente VALUES ('tom64','m3c3nghc85ph','1964-01-06','Hong Kong','en');
```

Esempio

SELECT * FROM Utente ORDER BY RAND() LIMIT 5;

```
+-----------+--------------------------+-----------------+---------+--------+
| username  | password                 | data_di_nascita | nazione | lingua |
+-----------+--------------------------+-----------------+---------+--------+
| fiorito96 | ltdexcfqjkemnh4b9b9h4p   | 1996-05-09      | Italy   | en     |
| grayfox83 | j9ynzpe                  | 1983-05-25      | Italy   | it     |
| mike60    | x64x1uy23xwlfrrskc5fbzgk | 1960-05-03      | Iceland | is     |
```

```
| tom96     | tulaykxbz071eyzwjl       | 1996-11-01      | France  | fr     |
| atzeni98  | b8fkvmsapv0              | 1998-10-18      | Italy   | it     |
+-----------+--------------------------+-----------------+---------+--------+
```

4.9 Giocatore

Creazione tabella

```
CREATE TABLE Giocatore (
        utente VARCHAR(25) NOT NULL,
        partita INTEGER NOT NULL,
        obiettivo INTEGER NOT NULL,
        PRIMARY KEY (utente, partita),
        FOREIGN KEY (utente)
                REFERENCES Utente (username)
                ON UPDATE CASCADE,
        FOREIGN KEY (partita)
                REFERENCES Partita (id)
                ON DELETE CASCADE
);
```

Inserimento

Inserimento di 44978 record da programma Java.

```
INSERT INTO Giocatore VALUES ('tom82',8384,3);
INSERT INTO Giocatore VALUES ('isaacs61',2066,1);
INSERT INTO Giocatore VALUES ('lebron47',2498,13);
INSERT INTO Giocatore VALUES ('liviusse64',7924,2);
INSERT INTO Giocatore VALUES ('alice68',2591,16);
```

Esempio

```
SELECT * FROM Giocatore ORDER BY RAND() LIMIT 5;
+----------+---------+-----------+
| utente   | partita | obiettivo |
+----------+---------+-----------+
| gigi46   |    8884 |        11 |
| simone97 |    8208 |        14 |
| chiara66 |    6311 |        13 |
| isaacs77 |    1115 |         8 |
| steph45  |    8516 |        16 |
+----------+---------+-----------+
5 rows in set (0,06 sec)
```

4.10 Amico

Creazione tabella

```
CREATE TABLE Amico (
        amicoa VARCHAR(25) NOT NULL,
        amicob VARCHAR(25) NOT NULL,
        PRIMARY KEY (amicoa, amicob),
        FOREIGN KEY (amicoa)
                REFERENCES Utente (username)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (amicob)
                REFERENCES Utente (username)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Inserimento

Inserimento di 2000 record da programma Java.
Per ogni coppia (‘amicoa’,’amicob’) ne esiste una (‘amicob’,’amicoa’)

```
INSERT INTO Amico VALUES('alfredo79','peter78');
INSERT INTO Amico VALUES('antonia83','fabrizio49');
INSERT INTO Amico VALUES('devjava0','camilla62');
INSERT INTO Amico VALUES('isaacs61','uranus72');
INSERT INTO Amico VALUES('alfredo0','fabio77');
```

Esempio

```
SELECT * FROM Amico ORDER BY RAND() LIMIT 5;
+-------------+------------+
| amicoa      | amicob     |
+-------------+------------+
| steph58     | steph60    |
| blanche63   | munchkin91 |
| carter90    | natalia65  |
| gribaudo94  | alefenix70 |
| bonifacio74 | munchkin52 |
+-------------+------------+
5 rows in set (0,00 sec)
```

4.11 Lingue

Creazione tabella

```
CREATE TABLE Lingue (
        nazione VARCHAR(50) NOT NULL,
        lingua CHAR(2) NOT NULL,
        principale BOOLEAN NOT NULL,
        PRIMARY KEY (nazione, lingua),
        FOREIGN KEY (nazione)
                REFERENCES Nazione (nome)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (lingua)
                REFERENCES Lingua (codice)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Inserimento

```
INSERT INTO Lingue VALUES ('Afghanistan','ps',1);
INSERT INTO Lingue VALUES ('Albania','sq',1);
INSERT INTO Lingue VALUES ('Albania','el',0);
INSERT INTO Lingue VALUES ('Algeria','ar',1);
INSERT INTO Lingue VALUES ('Algeria','fr',0);
```

Esempio

```
SELECT * FROM Lingue ORDER BY RAND() LIMIT 5;
```

```
+------------+--------+------------+
| nazione    | lingua | principale |
+------------+--------+------------+
| Cape Verde | pt     |          1 |
| Canada     | en     |          1 |
| Malta      | en     |          0 |
| Bhutan     | dz     |          1 |
| Gibraltar  | es     |          1 |
+------------+--------+------------+
```

4.12 Comprende

Creazione tabella

```
CREATE TABLE Comprende (
        obiettivo INTEGER NOT NULL,
        territorio VARCHAR(25) NOT NULL,
        PRIMARY KEY (obiettivo, territorio),
        FOREIGN KEY (obiettivo)
                REFERENCES Obiettivo (id)
```

```
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (territorio)
        REFERENCES Territorio (nome)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Inserimento

```
INSERT INTO Comprende VALUES (1,'Medio Oriente');
INSERT INTO Comprende VALUES (2,'Medio Oriente');
INSERT INTO Comprende VALUES (3,'Medio Oriente');
INSERT INTO Comprende VALUES (5,'Medio Oriente');
INSERT INTO Comprende VALUES (6,'Medio Oriente');
```

Esempio

```
SELECT * FROM Comprende ORDER BY RAND() LIMIT 5;
+-----------+----------------------+
| obiettivo | territorio           |
+-----------+----------------------+
|        14 | Australia Orientale  |
|         4 | Islanda              |
|         6 | Africa del Nord      |
|        16 | Perù                 |
|         5 | Australia Occidentale |
+-----------+----------------------+
```

4.13 Confina

Creazione tabella

```
CREATE TABLE Confina (
    territorioa VARCHAR(25) NOT NULL,
    territoriob VARCHAR(25) NOT NULL,
    PRIMARY KEY (territorioa, territoriob),
    FOREIGN KEY (territorioa)
        REFERENCES Territorio (nome)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (territoriob)
        REFERENCES Territorio (nome)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Inserimento

Per ogni coppia (‘territorioa’,’territoriob’) ne esiste una (‘territoriob’,’territorioa’)

```
INSERT INTO Confina VALUES('Argentina','Perù');
INSERT INTO Confina VALUES('Argentina','Brasile');
INSERT INTO Confina VALUES('Perù','Brasile');
INSERT INTO Confina VALUES('Perù','Venezuela');
INSERT INTO Confina VALUES('Brasile','Venezuela');
```

Esempio

```
SELECT * FROM Confina ORDER BY RAND() LIMIT 5;
+----------------------+-----------------+
| territorioa          | territoriob     |
+----------------------+-----------------+
| Jacuzia              | Siberia         |
| Congo                | Africa del Nord |
| Medio Oriente        | Cina            |
| Cina                 | Siam            |
| Europa Settentrionale | Gran Bretagna   |
+----------------------+-----------------+
```

4.14 Possiede

Creazione tabella

```
CREATE TABLE Possiede (
        utente VARCHAR(25) NOT NULL,
        partita INTEGER NOT NULL,
        carta INTEGER NOT NULL,
        PRIMARY KEY (utente, partita, carta),
        FOREIGN KEY (utente, partita)
                REFERENCES Giocatore (utente, partita)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (carta)
                REFERENCES Carta (id)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Inserimento

Inserimento di 112868 record da programma Java.

```
INSERT INTO Possiede VALUES ('celestino66',416,26);
INSERT INTO Possiede VALUES ('gribaudo65',7046,16);
INSERT INTO Possiede VALUES ('vilma85',8669,27);
INSERT INTO Possiede VALUES ('remo78',2874,33);
INSERT INTO Possiede VALUES ('bonaventura56',9382,3);
```

Esempio

SELECT * FROM Possiede ORDER BY RAND() LIMIT 5;

```
+----------+---------+-------+
| utente   | partita | carta |
+----------+---------+-------+
| fabio55  |    1129 |    18 |
| mattiw52 |    9451 |    12 |
| alfredo0 |    9269 |     7 |
| andrea72 |    4331 |    20 |
| mattiw86 |    9447 |    27 |
+----------+---------+-------+
```

4.15 Occupa

Creazione tabella

```
CREATE TABLE Occupa (
        utente VARCHAR(25) NOT NULL,
        partita INTEGER NOT NULL,
        territorio VARCHAR(25) NOT NULL,
        armate INTEGER NOT NULL,
        PRIMARY KEY (utente, partita, territorio),
        FOREIGN KEY (utente, partita)
                REFERENCES Giocatore (utente, partita)
                ON UPDATE CASCADE
                ON DELETE CASCADE,
        FOREIGN KEY (territorio)
                REFERENCES Territorio (nome)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Inserimento

Inserimento di 420000 record da programma Java.

```
INSERT INTO Occupa VALUES('mattiw86',1400,'Madagascar',6);
INSERT INTO Occupa VALUES('edo58',9900,'Afghanistan',3);
INSERT INTO Occupa VALUES('hubble49',444,'Quebec',9);
INSERT INTO Occupa VALUES('manu76',5644,'Alberta',1);
INSERT INTO Occupa VALUES('camel57',8979,'Quebec',10);
```

Esempio

```
SELECT * FROM Occupa ORDER BY RAND() LIMIT 5;
```

```
+-------------+---------+--------------------------+--------+
| utente      | partita | territorio               | armate |
+-------------+---------+--------------------------+--------+
| eva93       |    9785 | Afghanistan              |      9 |
| celestino72 |     243 | Siberia                  |      8 |
| bassetta91  |    1609 | Congo                    |      4 |
| mars99      |    6905 | Territori del Nord Ovest |      3 |
| lebron66    |    2042 | Giappone                 |      3 |
+-------------+---------+--------------------------+--------+
```

5. Query

5.1 Query di gioco

5.1.1 Lista dei giocatori, con i relativi obiettivi, in una partita X

```
SELECT
    utente, obiettivo
FROM
    Giocatore
WHERE
    partita = @partita;
```

ESECUZIONE

(SET @partita = 8033;)

```
+-------------+-----------+
| utente      | obiettivo |
+-------------+-----------+
| davide84    |        11 |
| erik2       |        14 |
| francesco80 |         2 |
| torloni67   |        12 |
+-------------+-----------+
4 rows in set (0,01 sec)
```

5.1.2 Lista dei territori, con i giocatori che li occupano e numero armate, in una partita X

```
SELECT
    utente,territorio,armate
FROM
    Occupa
WHERE
    partita =  @partita
ORDER BY
    territorio;
```

ESECUZIONE

(SET @partita = 8033;)

```
+----------+--------------------------+--------+
| utente   | territorio               | armate |
+----------+--------------------------+--------+
| davide84 | Afghanistan              |      9 |
| davide84 | Africa del Nord          |     10 |
| davide84 | Africa del Sud           |      8 |
| davide84 | Africa Orientale         |      7 |
| erik2    | Alaska                   |      9 |
...
| davide84 | Stati Uniti Orientali    |      6 |
| davide84 | Territori del Nord Ovest |      8 |
| davide84 | Ucraina                  |      3 |
| davide84 | Urali                    |      3 |
| davide84 | Venezuela                |      9 |
+----------+--------------------------+--------+
42 rows in set (0,24 sec)
```

5.1.3 Carte possedute da ogni giocatore in una partita X

SELECT
    Giocatore.utente, Carta.simbolo, **COUNT**(Carta.simbolo)
FROM
    Giocatore, Possiede, Carta
WHERE
    Giocatore.utente = Possiede.utente **AND**
    Giocatore.partita = Possiede.partita **AND**
    Possiede.carta = Carta.id **AND**
    Giocatore.partita = *@partita*
GROUP BY
    Giocatore.utente, Carta.simbolo;

ESECUZIONE

(SET @partita = 8033;)

```
+-------------+-----------+----------------------+
| utente      | simbolo   | COUNT(Carta.simbolo) |
+-------------+-----------+----------------------+
| davide84    | fante     |                    1 |
| davide84    | cannone   |                    2 |
| erik2       | fante     |                    2 |
| erik2       | cannone   |                    1 |
| erik2       | cavaliere |                    1 |
| erik2       | jolly     |                    1 |
| francesco80 | fante     |                    1 |
| francesco80 | cannone   |                    2 |
```

```
| francesco80 | cavaliere |                    2 |
| torloni67   | cannone   |                    2 |
| torloni67   | cavaliere |                    1 |
+-------------+-----------+----------------------+
11 rows in set (0,00 sec)
```

5.1.4 Giocatore vincitore (se esiste) di una partita X

```
SELECT
    Giocatore.utente, Giocatore.partita
FROM
    Giocatore
WHERE
    (Giocatore.utente, Giocatore.partita) NOT IN
        (SELECT
            Giocatore.utente, Giocatore.partita
        FROM
            Giocatore LEFT JOIN Comprende ON
                Giocatore.obiettivo = Comprende.obiettivo AND
                Giocatore.partita = @partita
            LEFT JOIN Occupa ON
                Comprende.territorio = Occupa.territorio AND
                Occupa.utente = Giocatore.utente AND
                Occupa.partita = Giocatore.partita
        WHERE
            Occupa.territorio IS NULL);
```

ESECUZIONE

(SET @partita = 8033;)

```
+----------+---------+
| utente   | partita |
+----------+---------+
| davide84 |    8033 |
+----------+---------+
1 row in set (0,81 sec)
```

5.1.5 Classifica punti giocatore in una partita X

```
SELECT
    Giocatore.utente, SUM(Territorio.punteggio) AS punti
FROM
    Giocatore LEFT JOIN Occupa ON
```

```
        Giocatore.utente = Occupa.utente AND
        Giocatore.partita = Occupa.partita
    LEFT JOIN Territorio ON
        Occupa.territorio = Territorio.nome
WHERE
    Giocatore.partita = @partita
GROUP BY
    Giocatore.utente
ORDER BY punti DESC;
```

ESECUZIONE

(SET @partita = 8033;)

```
+-------------+-------+
| utente      | punti |
+-------------+-------+
| davide84    |   151 |
| erik2       |    13 |
| francesco80 |  NULL |
| torloni67   |  NULL |
+-------------+-------+
4 rows in set (0,00 sec)
```

5.1.6 Numero di armate di rinforzo spettanti ad ogni giocatore in una partita X

```
SELECT
    Giocatore.utente, COUNT(Occupa.territorio) DIV 3
FROM
    Giocatore LEFT JOIN Occupa
ON
    Giocatore.utente = Occupa.utente AND
    Giocatore.Partita = Occupa.Partita
WHERE
    Giocatore.partita = @partita
GROUP BY
    Giocatore.Utente;
```

ESECUZIONE

(SET @partita = 8033;)

```
+-------------+--------------------------------+
| utente      | COUNT(Occupa.territorio) DIV 3 |
+-------------+--------------------------------+
| davide84    |                             12 |
| erik2       |                              1 |
| francesco80 |                              0 |
```

```
| torloni67   |                                0 |
+-------------+----------------------------------+
4 rows in set (0,00 sec)
```

5.1.7 Giocatori che occupano tutti i territori di un continente con relativo bonus di truppe in una partita X

```
SELECT
    G.utente, C.continente, C.bonus
FROM
    (SELECT
        Giocatore.utente AS utente, Continente.nome AS continente, COUNT(*) AS territoriposseduti
    FROM
        Giocatore, Occupa, Territorio, Continente
    WHERE
        Giocatore.utente = Occupa.utente AND
        Giocatore.partita = Occupa.partita AND
        Occupa.territorio = Territorio.nome AND
        Territorio.continente = Continente.nome AND
        Giocatore.partita = @partita
    GROUP BY
        Continente.nome, Giocatore.utente) AS G
    JOIN (SELECT
        Continente.nome AS continente, Continente.bonus AS bonus, COUNT(*) AS territoricontinente
    FROM
        Territorio, Continente
    WHERE
        Territorio.continente = Continente.nome
    GROUP BY
        Continente.nome) AS C
    ON
        G.territoriposseduti = C.territoricontinente AND
        G.continente = C.continente;
```

ESECUZIONE

(SET @partita = 8033;)

```
+----------+----------------+-------+
| utente   | continente     | bonus |
+----------+----------------+-------+
```

```
| davide84 | Africa          |     3 |
| davide84 | America del Sud |     2 |
| davide84 | Europa          |     5 |
+----------+-----------------+-------+
3 rows in set (0,00 sec)
```

5.1.8 Tris giocabili dai giocatori in una partita X

```
SELECT
    T1.giocatore, T1.simbolo, T1.simbolo, T2.simbolo
    FROM
        (SELECT
            Giocatore.utente AS giocatore, Carta.simbolo AS simbolo,
            COUNT(Carta.simbolo) AS numero
        FROM
            Giocatore, Possiede, Carta
        WHERE
            Giocatore.utente = Possiede.utente AND
            Giocatore.partita = Possiede.partita AND
            Possiede.carta = Carta.id AND
            Carta.simbolo != 'jolly' AND
            Giocatore.partita = @partita
        GROUP BY
            Giocatore.utente, Carta.simbolo
        HAVING
            COUNT(Carta.Simbolo)>1) AS T1
        JOIN (SELECT
            Giocatore.utente AS giocatore, Carta.simbolo AS simbolo
        FROM
            Giocatore, Possiede, Carta
        WHERE
            Giocatore.utente = Possiede.utente AND
            Giocatore.partita = Possiede.partita AND
            Possiede.carta = Carta.id AND
            Giocatore.partita = @partita AND
            Carta.simbolo ='jolly') AS T2
        ON
            T1.giocatore = T2.giocatore
UNION SELECT
    Giocatore.utente AS giocatore, Carta.simbolo AS simbolo, Carta.simbolo AS simbolo,
    Carta.simbolo AS simbolo
FROM
    Giocatore, Possiede, Carta
WHERE
```

```
        Giocatore.utente = Possiede.utente AND
        Giocatore.partita = Possiede.partita AND
        Possiede.carta = Carta.id AND
        Giocatore.partita = @partita
GROUP BY
        Giocatore.utente, Carta.simbolo
HAVING
        COUNT(Carta.Simbolo)>2
UNION SELECT DISTINCT
        T1.giocatore, T1.carta, T2.carta, T3.carta
FROM
        (SELECT
                Giocatore.utente AS giocatore, Carta.simbolo AS carta
        FROM
                Giocatore, Possiede, Carta
        WHERE
                Giocatore.utente = Possiede.utente AND
                Giocatore.partita = Possiede.partita AND
                Giocatore.partita = @partita AND
                Possiede.carta = Carta.id AND
                Carta.simbolo = 'fante') AS T1
        JOIN (SELECT
                Giocatore.utente AS giocatore, Carta.simbolo AS carta
        FROM
                Giocatore, Possiede, Carta
        WHERE
                Giocatore.utente = Possiede.utente AND
                Giocatore.partita = Possiede.partita AND
                Giocatore.partita = @partita AND
                Possiede.carta = Carta.id AND
                Carta.simbolo = 'cavaliere') AS T2
        ON
                T1.giocatore = T2.giocatore
        JOIN (SELECT
                Giocatore.utente AS giocatore, Carta.simbolo AS carta
        FROM
                Giocatore, Possiede, Carta
        WHERE
                Giocatore.utente = Possiede.utente AND
                Giocatore.partita = Possiede.partita AND
                Giocatore.partita = @partita AND
                Possiede.carta = Carta.id AND
                Carta.simbolo = 'cannone') AS T3
        ON
                T2.giocatore = T3.giocatore;
```

ESECUZIONE

(SET @partita = 8033;)

```
+-------------+---------+-----------+---------+
| giocatore   | simbolo | simbolo   | simbolo |
+-------------+---------+-----------+---------+
| erik2       | fante   | fante     | jolly   |
| erik2       | fante   | cavaliere | cannone |
| francesco80 | fante   | cavaliere | cannone |
+-------------+---------+-----------+---------+
3 rows in set (0,00 sec)
```

5.1.9 Attacchi possibili in una partita X

SELECT
GA.utente, Att.territorio, Att.armate, Dif.territorio, Dif.armate
FROM
Giocatore GA, Occupa Att, Occupa Dif, Confina
WHERE
GA.utente = Att.utente **AND**
GA.partita = Att.partita **AND**
Att.utente != Dif.utente **AND**
GA.partita = Dif.partita **AND**
Att.armate > 1 **AND**
Att.territorio = Confina.territorioa **AND**
Dif.territorio = Confina.territoriob **AND**
GA.partita = @*partita*;

ESECUZIONE

(SET @partita = 8033;)

```
+----------+--------------------------+--------+--------------------------+--------+
| utente   | territorio               | armate | territorio               | armate |
+----------+--------------------------+--------+--------------------------+--------+
| davide84 | Alberta                  |      7 | Alaska                   |      9 |
| davide84 | Australia Occidentale    |      6 | Indonesia                |      9 |
| davide84 | Kamchatka                |     10 | Alaska                   |      9 |
| davide84 | Kamchatka                |     10 | Cita                     |     10 |
| davide84 | Kamchatka                |     10 | Jacuzia                  |      4 |
...
| erik2    | Indonesia                |      9 | Australia Occidentale    |      6 |
| erik2    | Indonesia                |      9 | Nuova Guinea             |      8 |
| erik2    | Indonesia                |      9 | Siam                     |      9 |
| erik2    | Jacuzia                  |      4 | Kamchatka                |     10 |
| erik2    | Jacuzia                  |      4 | Siberia                  |      9 |
+----------+--------------------------+--------+--------------------------+--------+
22 rows in set (0,43 sec)
```

5.1.10 Territori di un giocatore che confinano solo con altri territori dello stesso giocatore nella partita X

```
SELECT
    T1.utente, T1.territorio, Territorio.punteggio
FROM
    Occupa T1, Confina, Occupa T2, Territorio
WHERE
    T1.territorio = Confina.territorioa AND
    T2.territorio = Confina.territoriob AND
    T1.partita = T2.partita AND
    T1.utente = T2.utente AND
    T1.territorio = Territorio.nome AND
    T1.partita = @partita
GROUP BY
    T1.territorio, T1.utente
HAVING
    COUNT(*) = Territorio.punteggio;
```

ESECUZIONE

(SET @partita = 8033;)

```
+----------+-------------------------+-----------+
| utente   | territorio              | punteggio |
+----------+-------------------------+-----------+
| davide84 | Afghanistan             |         4 |
| davide84 | Africa del Nord         |         6 |
| davide84 | Africa del Sud          |         3 |
| davide84 | Africa Orientale        |         5 |
| davide84 | America Centrale        |         3 |
...
| davide84 | Stati Uniti Occidentali |         4 |
| davide84 | Stati Uniti Orientali   |         4 |
| davide84 | Ucraina                 |         6 |
| davide84 | Urali                   |         4 |
| davide84 | Venezuela               |         3 |
+----------+-------------------------+-----------+
30 rows in set (0,76 sec)
```

5.1.11 Numero totale di armate sulla plancia per ogni giocatore in una partita X

```
SELECT
    Giocatore.utente, SUM(Occupa.armate)
FROM
    Giocatore, Occupa
WHERE
```

```
        Giocatore.utente = Occupa.utente AND
        Giocatore.partita = Occupa.partita AND
        Giocatore.partita = @partita
GROUP BY
        Giocatore.utente;
```

ESECUZIONE

(SET @partita = 8033;)

```
+----------+--------------------+
| utente   | SUM(Occupa.armate) |
+----------+--------------------+
| davide84 |                251 |
| erik2    |                 32 |
+----------+--------------------+
2 rows in set (0,00 sec)
```

5.2 Query statistiche

5.2.1 Utenti che hanno scelto una lingua non parlata nella propria nazione di residenza

```
SELECT
    Utente.username, Utente.nazione, Utente.lingua
FROM
    Utente LEFT JOIN Lingue ON
        Utente.nazione = Lingue.nazione AND
        Utente.lingua = Lingue.lingua
WHERE
    Lingue.nazione IS NULL;
```

ESECUZIONE

```
+-----------------+----------------------+--------+
| username        | nazione              | lingua |
+-----------------+----------------------+--------+
| alefenix47      | Italy                | en     |
| alefenix61      | Italy                | en     |
| andrea54        | Italy                | en     |
| andrea96        | Italy                | en     |
| android94       | Sudan                | en     |
...
| void99          | Cameroon             | it     |
| winston56       | Congo                | en     |
| winston70       | Greenland            | en     |
| winter66        | Djibouti             | en     |
| winter77        | Puerto Rico          | en     |
+-----------------+----------------------+--------+
153 rows in set (0,02 sec)
```

5.2.2 Numero medio di giocatori per partita

```
SELECT
    AVG(P.giocatori)
FROM
    (SELECT
        COUNT(*) AS giocatori
    FROM
        Giocatore
    GROUP BY
        partita)
    AS P;
```

ESECUZIONE

```
+------------------+
| AVG(P.giocatori) |
+------------------+
|           4.4978 |
+------------------+
1 row in set (0,05 sec)
```

5.2.3 Giocatori che hanno vinto la propria partita

```
SELECT
    Giocatore.utente, Giocatore.partita
FROM
    Giocatore
WHERE
    (Giocatore.utente,Giocatore.partita) NOT IN
        (SELECT
            Giocatore.utente, Giocatore.partita
        FROM
            Giocatore LEFT JOIN Comprende ON
                Giocatore.obiettivo = Comprende.obiettivo
            LEFT JOIN Occupa ON
                Comprende.territorio = Occupa.territorio AND
                Occupa.utente = Giocatore.utente AND
                Occupa.partita = Giocatore.partita
        WHERE
            Occupa.territorio IS NULL);
```

ESECUZIONE

```
+----------+---------+
| utente   | partita |
+----------+---------+
| vilma77  |    7648 |
| davide84 |    8033 |
+----------+---------+
2 rows in set (1,78 sec)
```

5.2.4 Primi 10 giocatori con più punti obiettivo

```
SELECT
    Giocatore.utente, Giocatore.partita, SUM(Territorio.punteggio) AS punti
FROM
    Giocatore, Comprende, Occupa, Territorio
WHERE
    Giocatore.obiettivo = Comprende.obiettivo AND
    Giocatore.utente = Occupa.utente AND
    Giocatore.partita = Occupa.partita AND
    Occupa.Territorio = Comprende.Territorio AND
    Territorio.nome = Occupa.territorio
GROUP BY
    Giocatore.utente, Giocatore.partita
ORDER BY punti DESC LIMIT 10;
```

ESECUZIONE

```
+-----------+---------+-------+
| utente    | partita | punti |
+-----------+---------+-------+
| vilma77   |    7648 |    86 |
| davide84  |    8033 |    86 |
| tim69     |    8802 |    83 |
| patty71   |    2207 |    83 |
| tom55     |    4089 |    83 |
| camilla45 |    8883 |    83 |
| jopaxxx83 |    6276 |    82 |
| lebron95  |    9774 |    82 |
| valerio80 |    9228 |    82 |
| tim47     |    8417 |    82 |
+-----------+---------+-------+
10 rows in set (2,49 sec)
```

5.2.5 Tutti i possibili attacchi in tutte le partite dove l'attaccante ha almeno 2 truppe in più del difensore

```
SELECT
    GA.partita, GA.utente, Att.territorio, Att.armate, GD.utente, Dif.territorio, Dif.armate
FROM
    Giocatore GA, Giocatore GD, Occupa Att, Occupa Dif, Confina
WHERE
    GA.utente = Att.utente AND
    GA.partita = Att.partita AND
    GD.utente = Dif.utente AND
    GD.partita = Dif.partita AND
    GA.utente != GD.utente AND
```

GA.partita = GD.partita **AND**
Att.armate > Dif.armate +1 **AND**
Att.territorio = Confina.territorioa **AND**
Dif.territorio = Confina.territoriob;

ESECUZIONE

```
+---------+-------------+-----------------------+--------+------------+--------------+--------+
| partita | utente      | territorio            | armate | utente     | territorio   | armate |
+---------+-------------+-----------------------+--------+------------+--------------+--------+
|    1934 | manfredi48  | Medio Oriente         |     10 | alefenix45 | Egitto       |      5 |
|    1934 | nessuno58   | Mongolia              |      8 | alefenix45 | Giappone     |      5 |
|    1934 | matteo77    | Jacuzia               |      6 | alefenix45 | Kamchatka    |      2 |
|    1934 | nessuno58   | Mongolia              |      8 | alefenix45 | Kamchatka    |      2 |
|    1934 | manfredi48  | Australia Occidentale |      5 | alefenix45 | Nuova Guinea |      2 |
...
|    8432 | alefenix45  | Mongolia              |      7 | winter84   | Giappone     |      4 |
|    8432 | alefenix45  | Cita                  |      6 | winter84   | Jacuzia      |      2 |
|    8432 | francesco45 | Groenlandia           |      5 | winter84   | Terr...Ovest |      3 |
|    8432 | alefenix45  | Ontario               |      6 | winter84   | Terr...Ovest |      3 |
|    8432 | alefenix45  | Ucraina               |      6 | winter84   | Urali        |      4 |
+---------+-------------+-----------------------+--------+------------+--------------+--------+
306564 rows in set (32,04 sec)
```

5.2.6 Lista delle nazioni con numero di utenti iscritti di ogni nazione

SELECT
    Utente.nazione, **COUNT**(*) **AS** utenti
FROM
    Utente
GROUP BY
    Utente.nazione
ORDER BY utenti **DESC**;

ESECUZIONE

```
+--------------------------------------+--------+
| nazione                              | utenti |
+--------------------------------------+--------+
| Italy                                |    827 |
| Egypt                                |      9 |
| Pakistan                             |      9 |
| Brunei Darussalam                    |      8 |
| Uganda                               |      8 |
...
| France, Metropolitan                 |      1 |
| Montserrat                           |      1 |
| Saint Kitts and Nevis                |      1 |
| St. Helena                           |      1 |
| Vietnam                              |      1 |
+--------------------------------------+--------+
235 rows in set (0,00 sec)
```

5.2.7 Primi 10 territori occupati mediamente da più armate

```
SELECT
    Occupa.territorio, AVG(Occupa.armate) AS armate
FROM
    Occupa
GROUP BY
    Occupa.territorio
ORDER BY
    armate DESC LIMIT 10;
```

ESECUZIONE

```
+--------------------------+--------+
| territorio               | armate |
+--------------------------+--------+
| Stati Uniti Orientali    | 5.5509 |
| Europa Meridionale       | 5.5470 |
| Mongolia                 | 5.5420 |
| Nuova Guinea             | 5.5387 |
| Brasile                  | 5.5313 |
| Stati Uniti Occidentali  | 5.5274 |
| Australia Orientale      | 5.5212 |
| India                    | 5.5211 |
| Urali                    | 5.5195 |
| Territori del Nord Ovest | 5.5188 |
+--------------------------+--------+
10 rows in set (1,18 sec)
```

5.2.8 Partite in cui tutti i giocatori parlano la stessa lingua

```
SELECT
    Giocatore.partita, COUNT(DISTINCT Utente.lingua) AS lingue
FROM
    Giocatore, Utente
WHERE
    Giocatore.utente = Utente.username
GROUP BY
    Giocatore.partita
HAVING
    lingue = 1;
```

ESECUZIONE

```
+---------+--------+
| partita | lingue |
+---------+--------+
|      27 |      1 |
|      36 |      1 |
|      37 |      1 |
|      55 |      1 |
|      67 |      1 |
...
|    9926 |      1 |
|    9929 |      1 |
|    9944 |      1 |
|    9953 |      1 |
|    9986 |      1 |
+---------+--------+
377 rows in set (0,08 sec)
```

5.2.9 Partite in cui tutti i giocatori parlano lingue diverse

SELECT
    Giocatore.partita, **COUNT**(Giocatore.utente) **AS** giocatori, **COUNT**(**DISTINCT** Utente.lingua) **AS** lingue
FROM
    Giocatore, Utente
WHERE
    Giocatore.utente = Utente.username
GROUP BY
    Giocatore.partita
HAVING
    lingue = giocatori;

ESECUZIONE

```
+---------+-----------+--------+
| partita | giocatori | lingue |
+---------+-----------+--------+
|       1 |         3 |      3 |
|       7 |         4 |      4 |
|      13 |         3 |      3 |
|      22 |         4 |      4 |
|      38 |         3 |      3 |
...
|    9971 |         4 |      4 |
|    9972 |         4 |      4 |
|    9973 |         3 |      3 |
|    9978 |         5 |      5 |
|    9985 |         4 |      4 |
+---------+-----------+--------+
1966 rows in set (0,10 sec)
```

5.2.10 10 utenti più anziani

```
SELECT
    username, YEAR(CURRENT_DATE) - YEAR(data_di_nascita) AS anni
FROM
    Utente
ORDER BY
    anni DESC LIMIT 10;
```

ESECUZIONE

```
+---------------+------+
| username      | anni |
+---------------+------+
| alefenix45    |   71 |
| camilla45     |   71 |
| exoplanet45   |   71 |
| bonaventura45 |   71 |
| alice45       |   71 |
| bassetta45    |   71 |
| carter45      |   71 |
| eva45         |   71 |
| crab45        |   71 |
| alfredo45     |   71 |
+---------------+------+
10 rows in set (0,01 sec)
```

5.2.11 Utenti che compiono gli anni oggi

```
SELECT
    username, YEAR(CURRENT_DATE) - YEAR(data_di_nascita) AS anni
FROM
    Utente
WHERE
    MONTH(CURRENT_DATE) = MONTH(data_di_nascita) AND
    DAY(CURRENT_DATE) = DAY(data_di_nascita);
```

ESECUZIONE

```
+-------------+------+
| username    | anni |
+-------------+------+
| feynman69   |   47 |
| francesco75 |   41 |
| juno74      |   42 |
| tim46       |   70 |
| winston2    |   14 |
+-------------+------+
5 rows in set (0,00 sec)
```

5.2.12 Tutti i giocatori che in una partita occupano un intero continente

```
SELECT
    G.utente, G.partita, C.continente, C.bonus
FROM
    (SELECT
        Occupa.utente AS utente, Occupa.partita AS partita, Continente.nome AS
        continente, COUNT(*) AS territoriposseduti
    FROM
        Occupa, Territorio, Continente
    WHERE
        Occupa.territorio = Territorio.nome AND
        Territorio.continente = Continente.nome
    GROUP BY
        Continente.nome, Occupa.utente, Occupa.partita) AS G
    JOIN (SELECT
        Continente.nome AS continente, Continente.bonus AS bonus, COUNT(*) AS
        territoricontinente
    FROM
        Territorio, Continente
    WHERE
        Territorio.continente = Continente.nome
    GROUP BY
        Continente.nome) AS C
ON
    G.territoriposseduti = C.territoricontinente AND
    G.continente = C.continente;
```

ESECUZIONE

```
+-----------------+---------+------------------+-------+
| utente          | partita | continente       | bonus |
+-----------------+---------+------------------+-------+
| alefenix64      |    2550 | Africa           |     3 |
| alefenix64      |    2662 | Africa           |     3 |
| alefenix70      |    5272 | Africa           |     3 |
| alefenix75      |    6165 | Africa           |     3 |
| alefenix83      |    1420 | Africa           |     3 |
...
| winter74        |    2244 | Oceania          |     2 |
| winter74        |    5368 | Oceania          |     2 |
| winter74        |    8015 | Oceania          |     2 |
| winter84        |    7102 | Oceania          |     2 |
| winter84        |    8432 | Oceania          |     2 |
+-----------------+---------+------------------+-------+
4984 rows in set (1,62 sec)
```

5.3 Query tradotte in algebra relazionale

5.3.1 Lista dei territori, con i giocatori che li occupano e numero armate, in una partita X (Query 5.1.2)

$$\pi_{giocatore,territorio,armate}(\rho_{giocatore \to utente}(Giocatore \bowtie (\sigma_{partita=432}(Occupa))))$$

5.3.2 Attacchi possibili in una partita X (Query 5.1.9)

$$\pi_{utente,territorioa,armatea,territoriob,armateb}\ (\sigma_{utente \neq utenteb}\ (((\rho_{territorioa,armatea \to territorio,armate}\ (\sigma_{partita=432 \wedge armate>1}\ (Occupa))) \bowtie Confina \bowtie (\rho_{territoriob,armateb,utenteb \to territorio,armate,utenteb}\ (\sigma_{partita=432}\ (Occupa)))))$$

6. Ottimizzazione

6.1 Storage engine

Lo storage engine utilizzato nell'implementazione della base di dati è InnoDB.

```
mysql> SELECT ENGINE, SUPPORT, COMMENT FROM ENGINES WHERE (Engine='InnoDB');
+--------+---------+----------------------------------------------------------------+
| ENGINE | SUPPORT | COMMENT                                                        |
+--------+---------+----------------------------------------------------------------+
| InnoDB | DEFAULT | Supports transactions, row-level locking, and foreign keys |
+--------+---------+----------------------------------------------------------------+
```

InnoDB, supportando le foreign keys, fornisce sia uno strumento per preservare l'integrità dei dati sia un utile supporto all'ottimizzazione.
L'integrità referenziale è preservata grazie alle azioni referenziali associate alle varie foreign keys. "Restrict" (valore di default) non permette l'update o il delete di record referenziati in altre tabelle, "Cascade" modifica o cancella a cascata tutti i record che referenziano al record su cui viene effettuato l'update o il delete.
Le foreign keys, inoltre, permettono di avere query più performanti. InnoDB, infatti, definisce indici secondari sulle foreign keys. Questi indici, quando sono utilizzati come condizione di join o nella clausola WHERE, permettono di trovare velocemente i record che soddisfano la condizione richiesta.

6.2 Trigger

6.2.1 Un territorio può essere occupato con un numero di armate che va da 1 a 10

Soddisfa il vincolo V2

```
DELIMITER //
CREATE TRIGGER armate_insert BEFORE INSERT ON Occupa
FOR EACH ROW
BEGIN
  IF NEW.armate < 1 THEN
    SET NEW.armate = 1;
  ELSEIF NEW.armate > 10 THEN
    SET NEW.armate = 10;
  END IF;
END
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER armate_update BEFORE UPDATE ON Occupa
FOR EACH ROW
BEGIN
  IF NEW.armate < 1 THEN
    SET NEW.armate = 1;
  ELSEIF NEW.armate > 10 THEN
    SET NEW.armate = 10;
  END IF;
END
//
DELIMITER ;
```

6.2.2 Un territorio è occupato da un solo giocatore durante la partita

Soddisfa il vincolo V1

```
DELIMITER //
CREATE TRIGGER territorio_insert BEFORE INSERT ON Occupa
FOR EACH ROW
BEGIN
  IF (NEW.territorio, NEW.partita) IN
    (SELECT
      territorio,partita
    FROM
      Occupa
    WHERE
```

```
      territorio = NEW.territorio
      AND partita = NEW.partita)
  THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Territorio già occupato';
  END IF;
END
//
DELIMITER ;
```

6.2.3 Una carta è posseduta a massimo un giocatore durante la partita

Soddisfa il vincolo V3

```
DELIMITER //
CREATE TRIGGER carta_insert BEFORE INSERT ON Possiede
FOR EACH ROW
BEGIN
  IF (NEW.carta, NEW.partita) IN
    (SELECT
      carta,partita
    FROM
      Possiede
    WHERE
      carta = NEW.carta
      AND partita = NEW.partita)
  THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Carta già posseduta';
  END IF;
END
//
DELIMITER ;
```

6.2.4 Un giocatore può possedere al massimo 5 carte

Soddisfa il vincolo V4

```
DELIMITER //
CREATE TRIGGER max_carte BEFORE INSERT ON Possiede
FOR EACH ROW
BEGIN
  IF (NEW.utente, NEW.partita) IN
    (SELECT
      utente,partita
```

```
    FROM
      Possiede
    WHERE
      utente = NEW.utente
      AND partita = NEW.partita
    GROUP BY (utente)
    HAVING
      COUNT(*)=5)
  THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Limite carte raggiunto';
  END IF;
END
//
DELIMITER ;
```

6.3 Stored procedure

6.3.1 Conquista territorio

```
DELIMITER //
CREATE PROCEDURE conquista (IN username VARCHAR(25), IN idpartita INTEGER, IN nometerritorio VARCHAR(25), IN numeroarmate INTEGER)
BEGIN
  UPDATE
    Occupa
  SET
    utente = username, armate = numeroarmate
  WHERE
    partita = idpartita AND
    territorio = nometerritorio;
END;
//
DELIMITER ;

CALL conquista(@username,@partita,@territorio,@armate);
```

6.3.2 Controllo vincitore di una partita

```
DELIMITER //
CREATE PROCEDURE vittoria (IN idpartita INTEGER, OUT vincitore VARCHAR(25))
BEGIN
  SELECT
    Giocatore.utente INTO vincitore
  FROM
    Giocatore
  WHERE
    (Giocatore.utente, Giocatore.partita) NOT IN
      (SELECT
        Giocatore.utente, Giocatore.partita
      FROM
        Giocatore LEFT JOIN Comprende ON
          Giocatore.obiettivo = Comprende.obiettivo AND
          Giocatore.partita = idpartita
        LEFT JOIN Occupa ON
          Comprende.territorio = Occupa.territorio AND
          Occupa.utente = Giocatore.utente AND
          Occupa.partita = Giocatore.partita
      WHERE
        Occupa.territorio IS NULL);
```

```
END;
//
DELIMITER ;

CALL vittoria(@partita, @vincitore);
SELECT @vincitore;
```

6.3 View

6.3.1 View utente

Questa view fornisce una maschera per la tabella Utente in cui sono salvate le password degli utenti.

```
CREATE VIEW ViewUtente AS
    SELECT
        username, nazione, lingua
    FROM
        Utente;
```

6.3.2 View percentuale plancia occupata

```
CREATE VIEW ViewPercentuale AS
    SELECT
        utente, partita, ROUND(COUNT(territorio)/42*100) AS percentuale
    FROM
        Occupa
    GROUP BY
        utente, partita;
```

6.4 Security

6.4.1 Password utenti in chiaro

Nella tabella Utente sono salvate le password degli utenti iscritti nella password per permetterne l'autenticazione.
Quando l'utente tenta l'accesso inserendo username e password viene eseguita la query:

SELECT username **FROM** Utente **WHERE** username = *u-inserita* **AND** password = *p-inserita*;

Che permette il controllo sulla coppia username password restituendo l'username dell'utente.
La password è però salvata in chiaro, nel caso ci siano intrusioni nel server tutte le password degli utenti sarebbero esposte.
Si può quindi ristrutturare la tabella per permettere di archiviare le password usando la funzione di hashing SHA2 a 224 bit cambiando il tipo di dato della colonna password da VARCHAR(25) a CHAR(56).

ALTER TABLE Utente **ADD** sha2 **CHAR**(56);
UPDATE Utente **SET** sha2 = SHA2(password,224);
ALTER TABLE Utente **DROP COLUMN** password;
ALTER TABLE Utente **CHANGE** sha2 password **CHAR**(56) **AFTER** username;

La nuova query che utilizzeremo per verificare username e password per l'accesso diventa:

SELECT username **FROM** Utente **WHERE** username = *u-inserita* **AND** password = **SHA2**(*p-inserita*, 224);

Ad esempio per l'utente con username alefenix75 e password 2v913jnlf8:

SELECT username **FROM** Utente **WHERE** username = 'alefenix75' **AND** password = **SHA2**('2v913jnlf8', 224);

Otteniamo:

```
+------------+
| username   |
+------------+
| alefenix75 |
+------------+
1 row in set (0,01 sec)
```

L'inserimento di un nuovo utente viene adesso effettuato tramite la nuova insert:

INSERT INTO Utente **VALUES** ('*username*', **SHA2**('*password_in_chiaro*',224), '*data_nascita*', '*nazione*', '*lingua*');

6.4.2 Creazione account mysql con privilegi ridotti

Creando un account con privilegi ridotti per l'accesso al database da parte del software che gestirà la logica di gioco (creazione e svolgimento delle partite) possiamo limitare l'esposizione dei dati dell'utente nascondendo la password e impedendo operazioni in scrittura sulle tabelle Utente, Nazione, Lingua, Lingue tramite la ViewUtente.

Creiamo il nuovo account:
CREATE USER 'game'@'%' **IDENTIFIED BY** 'gamepass';

Eseguiamo la seguente query per ottenere in output i comandi necessari per garantire al nuovo account i privilegi su tutte le tabelle tranne quelle indicate sopra:

SELECT CONCAT('GRANT ALL PRIVILEGES ON ', table_name, ' TO game@\'%\';') **FROM** information_schema.tables **WHERE** table_schema **=** 'risiko' **AND** table_name **NOT IN** ('Utente', 'Nazione', 'Lingua', 'Lingue');

```
+------------------------------------------------------------------+
| CONCAT('GRANT ALL PRIVILEGES ON ', table_name, ' TO game@\'%\';') |
+------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON Amico TO game@'%';                       |
| GRANT ALL PRIVILEGES ON Carta TO game@'%';                       |
| GRANT ALL PRIVILEGES ON Comprende TO game@'%';                   |
| GRANT ALL PRIVILEGES ON Confina TO game@'%';                     |
| GRANT ALL PRIVILEGES ON Continente TO game@'%';                  |
| GRANT ALL PRIVILEGES ON Giocatore TO game@'%';                   |
| GRANT ALL PRIVILEGES ON Obiettivo TO game@'%';                   |
| GRANT ALL PRIVILEGES ON Occupa TO game@'%';                      |
| GRANT ALL PRIVILEGES ON Partita TO game@'%';                     |
| GRANT ALL PRIVILEGES ON Possiede TO game@'%';                    |
| GRANT ALL PRIVILEGES ON Territorio TO game@'%';                  |
| GRANT ALL PRIVILEGES ON ViewPercentuale TO game@'%';             |
| GRANT ALL PRIVILEGES ON ViewUtente TO game@'%';                  |
+------------------------------------------------------------------+
```

Inoltre è necessario garantire al nuovo utente anche i privilegi per l'esecuzione delle stored procedure:

GRANT EXECUTE ON PROCEDURE risiko.conquista **TO** 'game'@'%';
GRANT EXECUTE ON PROCEDURE risiko.vittoria **TO** 'game'@'%';

6.5 Ottimizzazione query

6.5.1. Creazione indice partita su tabella Occupa

Poiché le query più significative sono quelle “di gioco” (5.1.*), si è cercato di concentrarsi sull’ottimizzazione di quest’ultime.
Le query più dispendiose sono quelle che effettuano scansioni della tabella “Occupa” che contiene oltre 400 mila record.
Nelle query di gioco si fa sempre riferimento ad una determinata partita.
Analizziamo, ad esempio, la semplice query **5.1.2**

```
SELECT
    utente, territorio, armate
FROM
    Occupa
WHERE
    partita = @partita
ORDER BY
    territorio;
```

L’esecuzione della query richiede 0,24 secondi.

Utilizzando il comando EXPLAIN otteniamo:

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Occupa
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 446803
     filtered: 10.00
        Extra: Using where; Using filesort
```

Non potendo utilizzare nessuna chiave la query richiede la scansione dell’intera tabella. La colonna partita, utilizzata nella WHERE, non è un indice in quanto parte di una chiave primaria multi-colonna.

Definendo la chiave sulla singola colonna:
CREATE INDEX occupapartita ON Occupa.partita;

Otteniamo come risultato dell’EXPLAIN sulla query precedente:

```
*************************** 1. row ***************************
           id: 1
```

```
  select_type: SIMPLE
        table: Occupa
   partitions: NULL
         type: ref
possible_keys: occupapartita
          key: occupapartita
      key_len: 4
          ref: const
         rows: 42
     filtered: 100.00
        Extra: Using index condition; Using filesort
```

Definendo l'indice sulla colonna partita la query può limitarsi a scansionare i record che soddisfano la condizione espressa nella WHERE.
L'esecuzione della stessa query adesso richiede meno di un centesimo di secondo.

La chiave rende più performante la query **5.1.9** che passa da un tempo di esecuzione di 0,43 secondi a 0,01 secondi e la query **5.1.10** che passa da 0,76 secondi a 0,01 secondi.

Anche la query statistica **5.2.3** dimezza il suo tempo di esecuzione da 1,78 secondi a 0,81 secondi.

6.5.2 Modifica query 5.1.4

Dopo aver creato l'indice sulla colonna partita nella tabella Occupa, l'unica query di gioco che ancora non risponde in un tempo immediato è la query **5.1.4**: l'esecuzione della query richiede 0,81 secondi.

Utilizzando il comando EXPLAIN otteniamo:

```
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: Giocatore
   partitions: NULL
         type: index
possible_keys: NULL
          key: partita
      key_len: 4
          ref: NULL
         rows: 40049
     filtered: 100.00
        Extra: Using where; Using index
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: Giocatore
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,partita
```

```
           key: PRIMARY
       key_len: 31
           ref: func,func
          rows: 1
      filtered: 100.00
         Extra: NULL
*************************** 3. row ***************************
            id: 2
  select_type: DEPENDENT SUBQUERY
         table: Comprende
    partitions: NULL
          type: ref
possible_keys: PRIMARY
           key: PRIMARY
       key_len: 4
           ref: risiko.Giocatore.obiettivo
          rows: 21
      filtered: 100.00
         Extra: Using where; Using index
*************************** 4. row ***************************
            id: 2
  select_type: DEPENDENT SUBQUERY
         table: Occupa
    partitions: NULL
          type: eq_ref
possible_keys: PRIMARY,territorio,occupapartita
           key: PRIMARY
       key_len: 58
           ref: risiko.Giocatore.utente,risiko.Giocatore.partita,risiko.Comprende.territorio
          rows: 1
      filtered: 100.00
         Extra: Using where; Not exists; Using index
```

Anche in questa query il limite alle prestazione è la scansione quasi intera della tabella Giocatore. La soluzione è esplicitare la condizione che Giocatore.partita sia uguale alla partita richiesta affinché la query possa applicare un filtro e scansionare solo le righe necessarie.

```
SELECT
        Giocatore.utente, Giocatore.partita
FROM
        Giocatore
WHERE
        Giocatore.partita = @partita AND
        (Giocatore.utente, Giocatore.partita) NOT IN
                (SELECT
                        Giocatore.utente, Giocatore.partita
                FROM
                        Giocatore LEFT JOIN Comprende ON
                                Giocatore.obiettivo = Comprende.obiettivo AND
                                Giocatore.partita = @partita
```

```
            LEFT JOIN Occupa ON
                Comprende.territorio = Occupa.territorio AND
                Occupa.utente = Giocatore.utente AND
                Occupa.partita = Giocatore.partita
        WHERE
            Occupa.territorio IS NULL);
```

Infatti adesso il comando EXPLAIN sulla nuova query ci dice:

```
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: Giocatore
   partitions: NULL
         type: ref
possible_keys: partita
          key: partita
      key_len: 4
          ref: const
         rows: 4
     filtered: 100.00
        Extra: Using where; Using index
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: Giocatore
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,partita
          key: PRIMARY
      key_len: 31
          ref: func,func
         rows: 1
     filtered: 100.00
        Extra: NULL
*************************** 3. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: Comprende
   partitions: NULL
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: risiko.Giocatore.obiettivo
         rows: 21
     filtered: 100.00
        Extra: Using where; Using index
*************************** 4. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: Occupa
```

```
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,territorio,occupapartita
          key: PRIMARY
      key_len: 58
          ref:
risiko.Giocatore.utente,risiko.Giocatore.partita,risiko.Comprende.territorio
         rows: 1
     filtered: 100.00
        Extra: Using where; Not exists; Using index
```

L'esecuzione della query adesso impiega 0,01 secondi.

6.5.3 Modifica query 5.2.5

Un'altra query molto dispendiosa che può essere migliorata è la **5.2.5**. La sua esecuzione richiede 32,04 secondi. Utilizzando il comando EXPLAIN otteniamo:

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Dif
   partitions: NULL
         type: ALL
possible_keys: PRIMARY,territorio,occupapartita
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 446803
     filtered: 100.00
        Extra: NULL
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: GD
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,partita
          key: PRIMARY
      key_len: 31
          ref: risiko.Dif.utente,risiko.Dif.partita
         rows: 1
     filtered: 100.00
        Extra: Using index
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: Confina
   partitions: NULL
         type: ref
possible_keys: PRIMARY,territoriob
          key: territoriob
```

```
      key_len: 27
          ref: risiko.Dif.territorio
         rows: 3
     filtered: 100.00
        Extra: Using index
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: GA
   partitions: NULL
         type: ref
possible_keys: PRIMARY,partita
          key: partita
      key_len: 4
          ref: risiko.Dif.partita
         rows: 4
     filtered: 90.00
        Extra: Using where; Using index
*************************** 5. row ***************************
           id: 1
  select_type: SIMPLE
        table: Att
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,territorio,occupapartita
          key: PRIMARY
      key_len: 58
          ref: risiko.GA.utente,risiko.Dif.partita,risiko.Confina.territorioa
         rows: 1
     filtered: 33.33
```

La query produce un prodotto cartesiano tra l'intera tabella Occupa (che deve essere necessariamente scansionata tutta) per mediamente 3 elementi della tabella Confina (per ogni territorio occupato vengono selezionati i territori confinanti) e il risultato viene poi ancora moltiplicato per mediamente 4 elementi della tabella Giocatore (per ogni territorio confinante vengono selezionati i giocatori che partecipano alla partita).
Questo ultimo passaggio potrebbe essere evitato in quanto ogni territorio può appartenere solamente ad un giocatore.

Si è quindi provato a creare un indice nella tabella Occupa per le due colonne territorio e partita:
CREATE INDEX partitaterritorio ON Occupa;

Analizzando la query tramite il comando EXPLAIN otteniamo adesso:

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Dif
   partitions: NULL
         type: ALL
possible_keys: PRIMARY,territorio,occupapartita,partitaterritorio
          key: NULL
```

```
      key_len: NULL
          ref: NULL
         rows: 446803
     filtered: 100.00
        Extra: NULL
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: Confina
   partitions: NULL
         type: ref
possible_keys: PRIMARY,territoriob
          key: territoriob
      key_len: 27
          ref: risiko.Dif.territorio
         rows: 3
     filtered: 100.00
        Extra: Using index
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: Att
   partitions: NULL
         type: ref
possible_keys: PRIMARY,territorio,occupapartita,partitaterritorio
          key: partitaterritorio
      key_len: 31
          ref: risiko.Dif.partita,risiko.Confina.territorioa
         rows: 1
     filtered: 33.33
        Extra: Using index condition; Using where
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: GA
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,partita
          key: PRIMARY
      key_len: 31
          ref: risiko.Att.utente,risiko.Dif.partita
         rows: 1
     filtered: 100.00
        Extra: Using index
*************************** 5. row ***************************
           id: 1
  select_type: SIMPLE
        table: GD
   partitions: NULL
         type: eq_ref
possible_keys: PRIMARY,partita
          key: PRIMARY
      key_len: 31
```

```
          ref: risiko.Dif.utente,risiko.Dif.partita
         rows: 1
     filtered: 100.00
        Extra: Using index
```

La query adesso, una volta individuate tutte le coppie *territorio del difensore* - *territorio confinante* con la prima join, può direttamente effettuare il join con la tabella Occupa cercando a quale giocatore appartiene il *territorio confinante* semplicemente individuandolo grazie all'indice territoriopartita.
L'esecuzione della query dopo la definizione dell'indice richiede 9,77 secondi.

Grazie al nuovo indice sulla tabella Occupa adesso è anche possibile ottenere le stesse informazioni in maniera più efficiente effettuando il join solamente tra le tabelle Occupa (una per il difensore e una per l'attaccante) e la tabella Confina:

SELECT
    Att.partita, Att.utente, Att.territorio, Att.armate, Dif.utente, Dif.territorio, Dif.armate
FROM
    Occupa Att, Occupa Dif, Confina
WHERE
    Att.utente != Dif.utente **AND**
    Att.partita = Dif.partita **AND**
    Att.armate > Dif.armate +1 **AND**
    Att.territorio = Confina.territorioa **AND**
    Dif.territorio = Confina.territoriob;

La query in quest'ultima versione richiede per l'esecuzione 9,02 secondi.

Appendice 1 : Giocatore, reificazione della relazione Utente-Partita

Per l'individuazione dell'entità "Giocatore" durante la fase di progettazione concettuale si è utilizzato il design pattern "reificazione di attributo di relazione".
L'entità "Giocatore" è la reificazione della relazione "Gioca" tra "Utente" e "Partita". La relazione "Gioca", per poter esprimere la regola del gioco per la quale un utente in una partita possiede esattamente un obiettivo, dovrebbe avere come attributo l'obiettivo stesso. Poiché l'obiettivo è un concetto rilevante per l'applicazione è necessario reificarlo e, per farlo, bisogna reificare anche la relazione della quale è attributo[2].
L'individuazione dell'entità "Giocatore" permette anche di evitare relazioni le relazioni ternarie "Occupa" (in cui partecipano le entità "Utente", "Partita" e "Territorio") e "Possiede" (in cui partecipano "Utente", "Partita" e "Carta"), la nuova entità "Giocatore", infatti, individuando una coppia del tipo Utente-Partita individua univocamente un utente che sta giocando una partita.

[2] dal Libro "Basi di dati", IV edizione, pag.246

PARTE 2: MongoDB

1. Analisi

Si è scelto di realizzare una base di dati utilizzando il DBMS non relazione MongoDB che rappresenti le partite e tutti i dati correlati necessari a descriverle accuratamente, tralasciando le informazioni riguardo le nazionalità e le lingue.

I dati da rappresentare saranno quindi le partite, i giocatori, gli obiettivi, i territori e i continenti con tutte le relazioni individuate tra queste entità necessarie a rappresentare correttamente lo stato di una partita (es. territori occupati, obiettivo di un giocatore).

A differenza di un DBMS relazionale, nel quale lo schema del database segue lo schema intrinseco dei dati da rappresentare, MongoDB con i suoi “schemi flessibili” permette di modellare la rappresentazione dei dati a seconda delle esigenze applicative.

Nella fase di analisi e progettazione diventa quindi necessario focalizzarsi su come l’applicazione interagisce con la base di dati scegliendo di conseguenza dove utilizzare delle referenze per rappresentare le relazioni e dove utilizzare l’embedding, e quindi un modello di dati denormalizzato.

2. Collections

2.1 Partite

Per rappresentare accuratamente lo stato di uno partita è necessario conoscere:

1. i giocatori che partecipano alla partita;
2. l'obiettivo di ogni giocatore;
3. i territori occupati da ogni giocatore, con il numero di armate;
4. le carte possedute da ogni giocatore.

Queste informazioni sono continuamente richieste dall'applicazione e sono soggette al maggior numero di update (ad esempio durante un attacco, o quando un giocatore pesca una carta). Per questo motivo è utile includerle tutte nel documento "Partita".

Struttura

```
{
	"_id": 1,
	"giocatori": [
	{
		"username":"Valerio53",
		"obiettivo":6,
		"territori": [
		{
			"nome":"Siam",
			"armate":4
		},
		… (altri territori occupati dal giocatore)
		],
		"carte": ["fante","cannone","cannone"]
	},
	… (altri giocatori della partita)
	]
}
```

Inserimento

Inserimento di 10000 documenti da file creato tramite programma Java.

2.2 Territori

Informazioni utili durante lo svolgimento di una partita riguardanti i singoli territori sono il continente di appartenenza, il punteggio associato al territorio e i suoi confini.
Si è scelto di non incorporare queste informazioni dentro i documenti delle partite per evitare costose ripetizioni.

Struttura

```
{
	“_id”:“Brasile”,
	“continente”:“America del Sud”,
	“punteggio”:4,
	“confini”:[“Africa del Nord”,“Argentina”,“Perù”,“Venezuela”]
}
```

Inserimento
Inserimento di 42 documenti da file creato tramite programma Java.

2.3 Continenti

I continenti servono a stabilire il numero di eventuali armate bonus in caso in cui un giocatore occupi tutti i territori di un continente. Le informazioni da salvare per ogni continente sono quindi il bonus associato e i territori che lo compongono.

Struttura
```
{
	“_id”:“America del Sud”,
	“bonus”:2,
	“territori”:[“Brasile”,“Argentina”,“Perù”,“Venezuela”]
}
```

Inserimento
Inserimento di 6 documenti da file.

2.4 Obiettivi

Gli obiettivi servono a determinare se un giocatore ha occupato tutti i territori indicati nel suo obiettivo. Per ogni obiettivo dobbiamo quindi salvare la lista dei territori che lo compongono.

Struttura
```
{
	“_id”:9,
	“territori”:[“Afghanistan”,“Cina”, … ]
}
```

Inserimento
Inserimento di 16 documenti da file creato tramite programma Java.

2.5 Utenti

Gli utenti non forniscono informazioni riguardo una singola partita ma forniscono informazioni utili all’applicazione (login, dati statistici, ecc…).

Struttura

```
{
	“_id”:“Valerio53”,
	“password”:“94d3f3eb4e8015209e9679c726e921f81cedfbccd54a80300bd3f3e0”,
	“nazione”:“Italy”,
	“lingua”:“it”
}
```

Inserimento

Inserimento di 1707 documenti da file creato tramite programma Java.

3. Query

3.1 Situazione partita

Possiamo ottenere le stesse informazioni che ottenevano con le query 5.1.1, 5.1.2 e 5.1.3 di MySQL con una semplice query sulla collection “partite”:

db.partite.find({"_id":8033}).pretty()

ESECUZIONE


```
{
	"_id" : 8033,
	"giocatori" : [
		{
			"username" : "davide84",
			"obiettivo" : 11,
			"territori" : [
				{
					"nome" : "Afghanistan",
					"armate" : 9
				},
				{
					"nome" : "Africa del Nord",
					"armate" : 10
				},
				…
			],
			"carte" : [
				"fante",
				"cannone",
				"cannone"
			]
		},
		{
			"username" : "erik2",
			"obiettivo" : 14,
			"territori" : [
				{
					"nome" : "Alaska",
					"armate" : 9
				},
				{
					"nome" : "Cita",
					"armate" : 10
				},
				…
			],
```

```
                        "carte" : [
                                "fante",
                                "fante",
                                "cavaliere",
                                "cannone",
                                "jolly"
                        ]
                },
                {
                        "username" : "francesco80",
                        "obiettivo" : 2,
                        "territori" : [ ],
                        "carte" : [
                                "fante",
                                "cavaliere",
                                "cavaliere",
                                "cannone",
                                "cannone"
                        ]
                },
                {
                        "username" : "torloni67",
                        "obiettivo" : 12,
                        "territori" : [ ],
                        "carte" : [
                                "cavaliere",
                                "cannone",
                                "cannone"
                        ]
                }
        ]
}
```


Possiamo inoltre utilizzare le projection nel metodo find per ottenere una proiezione di parte dei dati.

3.1.1 Giocatori in una partita con rispettivi obiettivi

db.partite.find({"_id":8033}, {"giocatori.username":1,"giocatori.obiettivo":1}).pretty()

ESECUZIONE


```
{
        "_id" : 8033,
        "giocatori" : [
                {
                        "username" : "davide84",
                        "obiettivo" : 11
                },
                {
                        "username" : "erik2",
                        "obiettivo" : 14
```

```
		},
		{
			"username" : "francesco80",
			"obiettivo" : 2
		},
		{
			"username" : "torloni67",
			"obiettivo" : 12
		}
	]
}
```

3.1.2 Giocatori in una partita con rispettivi territori occupati e numero armate

db.partite.find({"_id":8033}, {"giocatori.username":1,"giocatori.territori":1}).pretty()

ESECUZIONE

```
{
	"_id" : 8033,
	"giocatori" : [
		{
			"username" : "davide84",
			"territori" : [
				{
					"nome" : "Afghanistan",
					"armate" : 9
				},
				{
					"nome" : "Africa del Nord",
					"armate" : 10
				},
				…
			]
		},
		{
			"username" : "erik2",
			"territori" : [
				{
					"nome" : "Alaska",
					"armate" : 9
				},
				{
					"nome" : "Cita",
					"armate" : 10
				},
				…
			]
		},
		{
			"username" : "francesco80",
```

```
                        "territori" : [ ]
                },
                {
                        "username" : "torloni67",
                        "territori" : [ ]
                }
        ]
}
```


3.1.3 Giocatori in una partita con rispettive carte possedute

db.partite.find({"_id":8033}, {"giocatori.username":1,"giocatori.carte":1}).pretty()

ESECUZIONE


```
{
        "_id" : 8033,
        "giocatori" : [
                {
                        "username" : "davide84",
                        "carte" : [
                                "fante",
                                "cannone",
                                "cannone"
                        ]
                },
                {
                        "username" : "erik2",
                        "carte" : [
                                "fante",
                                "fante",
                                "cavaliere",
                                "cannone",
                                "jolly"
                        ]
                },
                {
                        "username" : "francesco80",
                        "carte" : [
                                "fante",
                                "cavaliere",
                                "cavaliere",
                                "cannone",
                                "cannone"
                        ]
                },
                {
                        "username" : "torloni67",
                        "carte" : [
                                "cavaliere",
                                "cannone",
```

```
                        "cannone"
                    ]
                }
            ]
        }
```

3.2 Vincitore partita

Per conoscere se un giocatore ha vinto una partita (Query 5.1.4 in MySQL) è necessario accedere a due collections: “partite” e “obiettivi”. Con una prima query, usando l’operatore di aggregazione, creiamo una nuova collection temporanea “vittoria” che contenga un documento per ogni giocatore della partita in cui sia inclusa anche la lista dei territori richiesti dall’obiettivo.

```
db.partite.aggregate([
        {$match:{"_id":8033}},
        {$project:{"_id":0,"giocatori.username":1,"giocatori.obiettivo":1,"giocatori.territori.
        nome":1}},
        {$unwind:{path:"$giocatori"}},
        {$lookup:{from:"obiettivi",localField:"giocatori.obiettivo",foreignField:"_id",as:"list
        a"}},
        {$out:"vittoria"}]).pretty()
```

Eseguiamo poi una query sulla collection “vittoria” per controllare se ci sia qualche giocatore per il quale ogni territorio indicato dall’obiettivo sia presente nella lista dei territori posseduti.

```
db.vittoria.find("this.lista[0].territori.every(o => {for (var i=0;
i<this.giocatori.territori.length; i++) {if(this.giocatori.territori[i].nome == o) return
true};});").pretty()
```

ESECUZIONE


```
{
        "_id" : ObjectId("5756ebd3d86af92157db7f03"),
        "giocatori" : {
                "username" : "davide84",
                "obiettivo" : 11,
                "territori" : [
                        {
                                "nome" : "Afghanistan"
                        },
                        …
                ]
        },
        "lista" : [
                {
                        "_id" : 11,
                        "territori" : [
                                "Afghanistan",
                                …
                        ]
                }
        ]
}
```

3.3 Partite di un utente

db.partite.find({"giocatori.username":"davide84"}).pretty()

3.4 Totale armate di ogni giocatore in una partita

(Query 5.1.11 di MySQL)

db.partite.aggregate([
{$match:{_id:8033}},
{$unwind:"$giocatori"},
{$unwind:"$giocatori.territori"},
{$group:{_id:{partita:"$_id",giocatore:"$giocatori.username"},totaleArmate:{$sum:
"$giocatori.territori.armate"}}}]).pretty()

ESECUZIONE

```
{
        "_id" : {
                "partita" : 8033,
                "giocatore" : "erik2"
        },
        "totaleArmate" : 32
}
{
        "_id" : {
                "partita" : 8033,
                "giocatore" : "davide84"
        },
        "totaleArmate" : 251
}
```