**DOCUMENTAZIONE PROGETTAZIONE BASE DI DATI RdF**

Premessa: l’intera base di dati è stata sviluppata in lingua inglese per il solo scopo di mantenere lineare la scelta fatta riguardante il codice (scritto in inglese).

RACCOLTA E ANALISI DEI REQUISITI

**Requisiti dati**

ServerRdF fornisce servizi di supporto alla memorizzazione ed analisi dei dati dell'intera piattaforma di gioco. E` richiesto di progettare la base di dati che ServerRdF utilizzerà per gestire informazioni relative: ai concorrenti, agli osservatori, agli amministratori, alle manche e alle partite giocate, e alle frasi misteriose. In particolare, sono di interesse le seguenti informazioni:

1. il profilo dei concorrenti, degli osservatori e degli amministratori regolarmente registrati, caratterizzato dal cognome, il nome, un nickname, un indirizzo email, e una password.

2. Lo storico delle manche di gioco e delle partite che sono state giocate e che si stanno giocando, tenendo traccia di tutte le mosse fatte dai concorrenti, e dei punti acquisiti e persi con ogni mossa.

3. La collezione di frasi misteriose e dei relativi temi. Per ogni manche di gioco, RdF potrà selezionare solo frasi misteriose che non sono state già usate in manche in cui hanno giocato i concorrenti coinvolti nella manche considerata, o a cui hanno assistito come osservatori.

Come fonte dei requisiti per la prima fase di progettazione abbiamo fatto riferimento al testo RdF, più precisamente al capitolo Requisiti dati (qui sopra citato interamente). Da questa documentazione già comprensibile e ben definita abbiamo effettuato alcune operazioni di modifica, riorganizzando le frasi per concetti, sottolineando sinonimi e infine producendo un GLOSSARIO DEI TERMINI, che è stato molto utile per la successiva fase di progettazione.

PROGETTAZIONE CONCETTUALE

Dal documento di specifica abbiamo ottenuto uno schema ER per strategia mista: abbiamo realizzato un primo generico schema scheletro in cui abbiamo rappresentato i concetti principali (entità e associazioni), per poi raffinarlo e ottenere lo schema ER finale.

Nel primo schema scheletro abbiamo capito che era possibile eliminare l’associazione tra *Users* e *Phrases* dato che entrambi erano già associati all’entità *Manches*, evitando di ripetere troppe volte i dati. Abbiamo inoltre capito che era necessaria una distinzione tra concorrenti e admin e abbiamo quindi introdotto nel primo schema scheletro raffinato una gerarchia di generalizzazione tra *Users*, entità padre, e Administrators e Players, entità figlie. Tale generalizzazione è del tipo totale ed esclusiva. Infine, abbiamo aggiunto un’associazione ulteriore tra *Manche* e *Moves*, chiamata *Contains*, dato che le mosse vengono fatte all’interno di una manche specifica.

Ottenuto lo schema scheletro raffinato finale (II) abbiamo costruito lo schema ER aggiungendo gli attributi. Attributi scelti a partire dal documento di specifica (cosi come le entità) selezionando le parole che rappresentavano proprietà elementari (nome, cognome email…).

Di seguito abbiamo scritto per ogni entità nello schema ER ottenuto vincoli di identificazione, vincoli di cardinalità, dominio degli attributi e specifiche riguardanti la gerarchia di generalizzazione sopra citata.

ENTITA’ USERS

Attributi:

* Id: varchar, identificatore interno semplice;
* Name: varchar, obbligatorio e monovalore;
* Surname: varchar, obbligatorio e monovalore;
* Nickname: varchar, obbligatorio e monovalore;
* Password: varchar, obbligatorio e mono-valore;
* Email: varchar, obbligatorio e mono-valore;

Associazioni:

L’entità *Users* è collegata all’entità *Moves* tramite l’associazione binaria *Make*. Vincoli di cardinalità:

Nessuno o più utenti fanno mosse (0,n) mentre una e una sola mossa specifica(moveid) può essere fatta da un utente specifico (1,1).

L’entità *Users* è inoltre collegata a

* *Manches* tramite l’associazione *Observe*;

Vincoli di cardinalità:

l’utente può vincere nessuna, una o più manches (0,n) e una manche o è stata vinta oppure no (0,1).

* *Matches* tramite l’associazione *Play*;

Vincoli di cardinalità:

Un *Match* può essere o non essere vinto (0,1) mentre un utente può aver vinto nessuna, una o più partite (0,n).

ENTITA’ MATCHES

Attributi:

* Id: varchar, identificatore interno semplice;
* Date: date, obbligatorio e monovalore;
* Time: time, obbligatorio e monovalore;

Associazioni:

Come detto prima l’entità *Matches* è collegata ad *Users* tramite *Play*. Inoltre, è collegata a *Manches* tramite *Made Of*.

Vincoli di cardinalità:

un match è fatto da 5 manches, ma nel caso in cui un giocatore abbandoni la partita prima del termine il match viene interrotto (1,5), una manche invece è di uno e un solo match (1,1).

ENTITA’ MANCHES

Attributi:

* Number: int, identificatore misto composto (insieme a id dell’entità *Matches*);

Associazioni:

*Manches* è collegata a *Matches* da *Made Of* e ad *Users* tramite *Observe*. Inoltre, è collegata a

* *Moves* da *Contain*.

Vincoli di cardinalità:

una manche contiene n mosse ma nel caso in cui una partita venga interrotta immediatamente può non contenere mosse (0,n) invece una mossa specifica è contenuta in una e in una sola manche(1,1).

* *Phrases* da *Use*.

Vincoli di cardinalità:

una manche utilizza una e una sola frase (1,1) e una frase è usata nessuna volta, una volta, oppure più volte (0,n).

ENTITA’ MOVES

Attributi:

* Moveid: varchar; identificatore semplice e interno;
* Outcome: {0,-1}, obbligatorio e monovalore;
* MoveType: {”perde”,”errore”, ”passa”, ”jolly”, ”soluzione”, ”consonante”, ”vocale”}, obbligatorio e monovalore;

Associazioni:

*Moves* è collegata a *Users* da *Make* e con *Manches* da *Contain*.

ENTITÀ PHRASES

Attributi:

* Id: contatore, identificatore semplice interno, obbligatorio;
* Phrase: varchar, attributo obbligatorio;
* Theme: varchar, attributo obbligatorio;

Associazioni:

*Phrases* è collegata a *Manches* da *Use*.

GERARCHIA DI GENERALIZZAZIONE

Dopo un primo approccio nello schema scheletro, abbiamo deciso di riprendere nello schema ER una suddivisione più definita dell’entità *Users*. Più precisamente abbiamo utilizzato una gerarchia di generalizzazione tra *Users* (entità padre) e Administrators e Players(entità figlie). Abbiamo optato per questa scelta in quanto ogni istanza delle entità condivideva gli stessi attributi (nome, cognome, nickname, …). La gerarchia è del tipo totale ed esclusiva, infatti ogni istanza dell’entità padre è istanza di almeno una delle entità figlie (totale) e ogni istanza può essere solo e soltanto di un’entità figlia(esclusiva).

PROGETTAZIONE LOGICA:

FASE DI RISTRUTTURAZIONE

In questa fase bisogna eliminare gerarchie di generalizzazione, attributi composti o multi-valore dallo schema ER precedentemente sviluppato.

Nella gerarchia di generalizzazione totale ed esclusiva tra le entità Players e Administrators abbiamo deciso di eliminare le entità figlie, accorpandole all’entità padre (*Users*). Abbiamo così inserito nell’entità padre l’attributo <type> che permette di distinguere a quale entità figlia esso si riferisca. Dato che la gerarchia è del tipo totale, per rispettare i vincoli di cardinalità il tipo non può mai essere nullo. La scelta fatta permette un risparmio nella memoria del database in quanto le entità figlie accorpate nell’entità padre non introducono più valori nulli.

PROGETTAZIONE LOGICA:

FASE DI TRADUZIONE

Lo schema ER ristrutturato è stato tradotto nello schema relazionale equivalente applicando regole di trasformazione. Dalle regole sono nate nuove tabelle che prima rappresentavano associazioni: Make, Play e Observe. Queste sono state poi rinominate alla fine della progettazione in Manchewinner (prima Observe), Matchwinner (prima Play), Manchejoiners (prima Make) perché meglio rappresentavano il loro contenuto. Inoltre sempre alla fine della progettazione sono sorte delle lacune riguardanti alcuni aspetti non modellati (come per esempio la gestione degli osservatori). Abbiamo così optato per l’inserimento di nuovi attributi nelle tabelle già esistenti:

* observer, attributo che indica la partecipazione di un utente ad una partita da osservatore (true of false);
* amount, attributo numeric che serve a tener conto dei punti accomulati nella manche e nel match.