

Relazione Database

Relazione - Progetto di un'applicazione per la creazione di squadre Pokémon

Corso: Basi di Dati

Anno Accademico: 2024/2025

Studente: Javid Ameri

Email: [javid.ameri@studio.unibo.it]

Indice

1. [Analisi dei Requisiti](#)
 - 1.1 [Intervista](#)
 - 1.2 [Rilevamento delle Ambiguità e Correzioni Proposte](#)
 2. [Progettazione Concettuale](#)
 - 2.1 [Estrazione dei Concetti Principali](#)
 - 2.2 [Descrizione modulare dello schema E-R](#)
 - 2.3 [Schema E-R completo](#)
 3. [Progettazione Logica](#)
 - 3.1 [Stima del Volume Dati](#)
 - 3.2 [Operazioni Principali](#)
 - 3.3 [Schemi di Navigazione e Tabelle degli Accessi](#)
 - 3.4 [Raffinamento dello Schema](#)
 4. [Traduzione delle Operazioni in Query SQL](#)
 5. [Progettazione dell'Applicazione](#)
 - 5.1 [Architettura](#)
 - 5.2 [Schermate](#)
-

1. Analisi dei Requisiti

1.1 Intervista

Un primo testo ottenuto dall'intervista è il seguente:

L'obiettivo del progetto è quello di realizzare un sito web pensato per i fan dei videogiochi Pokémon, in particolare per coloro che amano costruire e organizzare squadre di Pokémon personalizzate, come accade nei giochi ufficiali. Il sistema sarà accessibile da qualsiasi browser e consentirà agli utenti registrati di creare, modificare e gestire i propri team, ciascuno composto da un massimo di sei Pokémon.

Ogni utente potrà accedere con le proprie credenziali e disporre di una propria area riservata in cui salvare uno o più team. La scelta dei Pokémon avverrà da un elenco completo disponibile all'interno della piattaforma. Per ogni Pokémon saranno mostrati dati come il nome, il numero nazionale del Pokédex (cioè la numerazione ufficiale), il tipo di appartenenza (ad esempio Fuoco, Acqua, ecc.) e la generazione del gioco in cui è stato introdotto.

Sono previsti due tipi di utenti:

- **Utente normale**, che potrà semplicemente creare e gestire squadre;
- **Amministratore**, che oltre alle stesse funzionalità degli altri utenti, potrà accedere a una sezione dedicata alle statistiche, in cui potrà visualizzare informazioni come il numero medio di Pokémon per squadra, il Pokémon più usato in assoluto, o il tipo più diffuso.

Ogni squadra sarà interamente personalizzabile. Ad esempio, oltre alla scelta dei Pokémon, l'utente potrà assegnare a ciascun membro del team un oggetto da tenere durante la lotta (come nelle meccaniche di gioco) e potrà consultare informazioni aggiuntive per ogni creatura: statistiche di base, mosse che può imparare, abilità speciali e l'elenco dei giochi in cui compare.

Il sito sarà quindi strutturato per offrire un'esperienza semplice e intuitiva, mettendo a disposizione tutti gli strumenti necessari per costruire con cura il proprio team ideale, come farebbe un vero allenatore Pokémon.

1.2 Rilevamento delle Ambiguità e Correzioni Proposte

Dall'intervista condotta sono emerse alcune possibili ambiguità e decisioni progettuali da chiarire prima di procedere con la progettazione concettuale.

1. Numero di squadre per utente

- *Ambiguità:* non era stato inizialmente specificato se un utente potesse creare più squadre o una sola.
- *Correzione proposta:* si è deciso che un utente può creare più squadre, ognuna indipendente.

2. Numero massimo di Pokémon per squadra

- *Ambiguità:* non era stato specificato il numero massimo e minimo di Pokémon presenti in una singola squadra.
- *Correzione proposta:* ogni squadra potrà contenere da 1 a 6 Pokémon, coerentemente con le regole tradizionali dei giochi Pokémon.

3. Tipo di utente

- *Ambiguità:* non era chiaro se usare due entità separate (Admin e Utente) o una sola.
- *Correzione proposta:* si userà una unica entità "Utente" con un attributo ruolo che assume i valori "standard" o "admin".

4. Presenza di diverse versioni di gioco

- *Ambiguità:* inizialmente non era stato specificato se il sito dovesse trattare una singola versione dei giochi Pokémon oppure gestire più versioni (Rosso/Blu, Giallo, Oro/Argento, Cristallo ecc.).
- *Correzione proposta:* si è deciso di introdurre una entità "Versione di Gioco". Al momento della creazione di una squadra, l'utente selezionerà la versione di gioco, che determinerà l'elenco dei Pokémon disponibili. Ogni squadra sarà associata esattamente a una versione.

5. Duplicati in squadra

- *Ambiguità:* inizialmente si era pensato di non poter permettere di avere più esemplari dello stesso Pokémon (es: due Pikachu) in una singola squadra, in quanto complicato da gestire poiché avrebbero presentato lo stesso attributo ID dell'entità Pokémon e lo stesso ID di squadra.
- *Correzione proposta:* alla fine, per mantenere coerenza con i giochi, si è trovato un modo per rendere possibile avere Pokémon duplicati nella stessa squadra, grazie alla creazione di una nuova entità "Pokémon in Squadra" che rappresenta il ruolo specifico che il Pokémon assume nel contesto della sua squadra. Ogni entità di questo tipo avrà un ID univoco, rendendolo quindi distinguibile anche in presenza di duplicati.

6. Regione e generazione

- *Ambiguità*: in una versione di gioco, e dunque in un Pokédex Regionale, possono apparire Pokémon provenienti da generazioni diverse.
- *Correzione proposta*: si modellerà un attributo generazione per ogni Pokémon, che potrà assumere il corrispondente valore numerico, a seconda dell'effettiva generazione di introduzione.

7. Gestione dei tipi Pokémon

- *Ambiguità*: non era chiaro se i tipi (es. Fuoco, Acqua, Erba, ecc.) dovessero essere semplicemente attributi
 - *Correzione proposta*: per garantire normalizzazione dei dati ed eventuali estensioni future (come relazioni tra tipi), si è deciso di modellare i Tipi Pokémon come entità separate collegate ai Pokémon tramite associazioni.
-
-

2. Progettazione Concettuale

2.1 Estrazione dei Concetti Principali

Concetto	Descrizione	Sinonimi
User	Utente registrato con ruolo (standard/admin)	Utente, Trainer, Allenatore
Team	Squadra Pokémon creata da un utente	Squadra, Party
Pokémon	Specie della creatura nel database	Mostriattolo tascabile, Specie
Type	Tipologia associata ai Pokémon (es. Fuoco, Acqua)	Elemento, Tipo
VideoGame	Edizione del gioco selezionata	Gioco, GameVersion, Versione
PokémonInTeam	Istanza di Pokémon all'interno di una squadra	—
Ability	Abilità posseduta da un Pokémon	Abilità
Move	Mossa che un Pokémon può apprendere	Attacco, Mossa
Item	Oggetto tenuto da un Pokémon in squadra	Strumento
Statistics	Valori base di un Pokémon	Statistiche, Stat
Pokedex	Elenco dei Pokémon ottenibili in una versione	—
PokemonInPokedex	Presenza di un Pokémon in uno specifico Pokédex	—

2.2 Descrizione modulare dello schema E-R

Per facilitare la lettura e la comprensione del modello concettuale, lo schema E-R viene scomposto in sottosezioni tematiche, ognuna delle quali descrive un gruppo di entità e relazioni fortemente connesse tra loro.

2.2.1 Utenti e Squadre

Il sistema gestisce due tipologie di utenti: normali e amministratori. Entrambi sono rappresentati dall'entità **User**, identificata dal proprio **username**.

Admin è una specializzazione di **User**, rappresentata nel modello come gerarchia.

Ogni utente può creare una o più squadre, modellate tramite l'entità **Team**, con la relazione **owns**.
Ogni **Team** è identificato da un **id** e contiene un nome personalizzabile.

La composizione delle squadre è gestita tramite l'entità **PokemonInTeam**, che collega ogni squadra a uno o più Pokémon.

Questa entità registra anche la posizione del Pokémon nella squadra (**positionInTeam**) e l'oggetto eventualmente tenuto, tramite la relazione **holds** con **Item**.

2.2.2 Pokémon, Tipi e Statistiche

L'entità **Pokemon** rappresenta ciascuna specie, contenente attributi come il numero Pokédex nazionale (**nationalId**), il nome, la generazione e l'immagine.

Ogni Pokémon ha:

- **Un profilo di statistiche** memorizzato in **Statistics**, collegato tramite la relazione 1:1 **has**
- **Una sola abilità**, collegata tramite la relazione **has_ability** con l'entità **Ability**
- **Uno o due tipi**, modellati dall'entità **Type** e connessi tramite la relazione **pokemon_type**

Le relazioni tra i tipi sono modellate tramite la relazione binaria **weakness**, che collega due istanze di **Type** specificando l'efficacia (**multiplicator**).

2.2.3 Mosse e modalità di apprendimento

L'entità **Move** rappresenta tutte le mosse disponibili, con attributi come nome, potenza, precisione, categoria, PP e descrizione.

Ogni mossa è associata a un solo tipo tramite la relazione **move_type** con l'entità **Type**.

Move è generalizzata in tre sottoentità specializzate, a seconda della modalità di apprendimento:

- **LevelMove** per le mosse apprese salendo di livello (relazione **learning** con **Pokemon**)
- **TMHMMove** per le mosse apprese tramite macchina tecnica (relazione **teaching** con **Pokemon**)
- **EggMove** per le mosse apprese tramite allevamento, che include anche i due genitori compatibili (relazione **born_with**)

Ogni specializzazione eredita l'identificatore dalla super-entità **Move**.

Le relazioni che collegano queste entità a **Pokemon** sono N:M e sono modellate come entità intermedie con attributi specifici (es. livello di apprendimento per **LevelMove**).

2.2.4 Versioni di gioco e Pokédex

Ogni squadra viene creata in riferimento a una specifica **VideoGame**, identificata da due versioni (es. "Rosso/Blu") e collegata tramite la relazione **regional_pokedex** a un **Pokedex** regionale.

Il contenuto di ogni Pokédex è definito dalla relazione `is` che collega `Pokemon` a `Pokedex` tramite l'entità `PokemonInPokedex`.

Quest'ultima registra il numero Pokédex regionale del Pokémon e una descrizione contestuale.

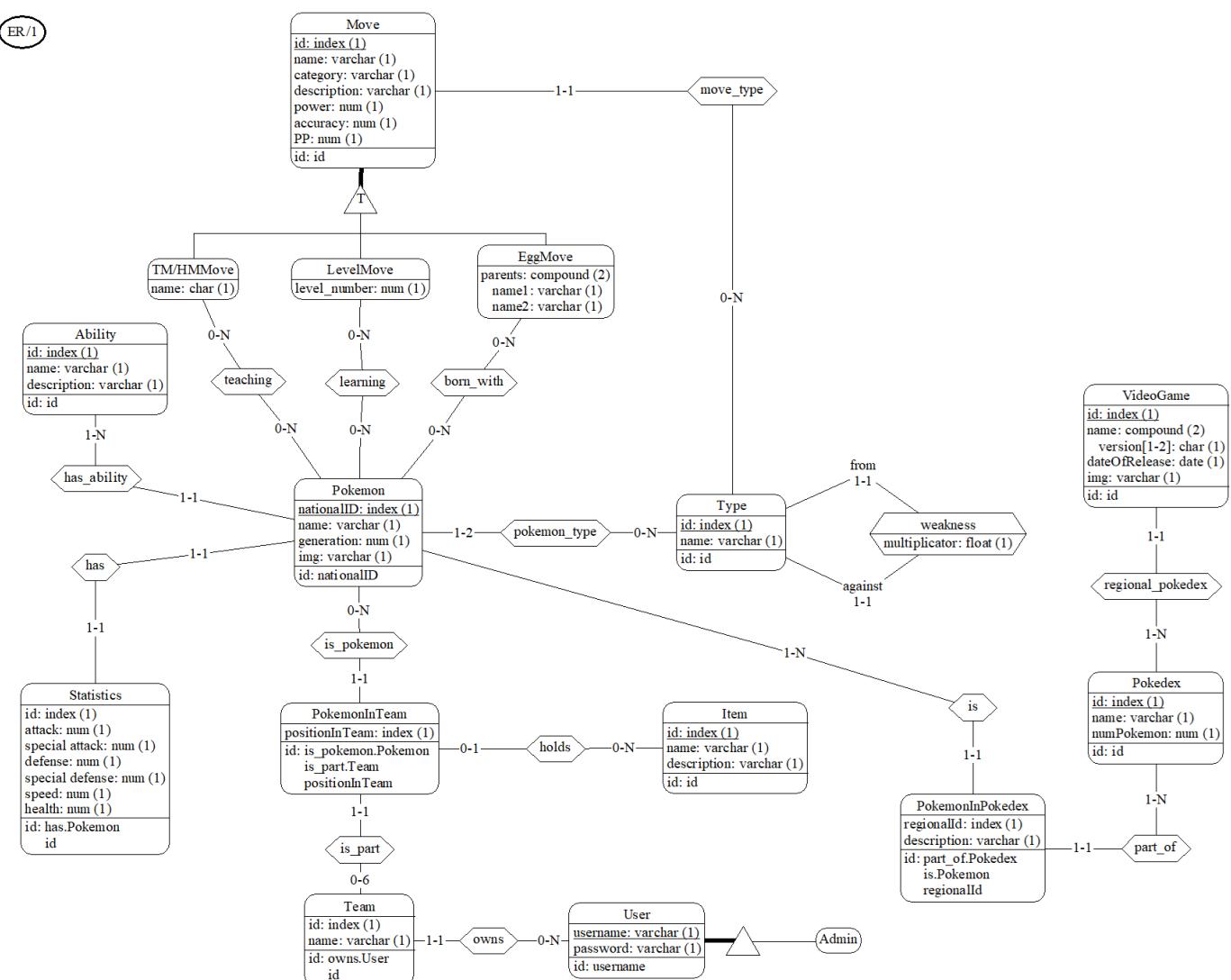
2.2.5 Oggetti

L'entità `Item` rappresenta gli oggetti che un Pokémon può tenere in squadra (es. strumenti, bacche, oggetti evolutivi).

L'associazione è modellata con la relazione `holds` tra `PokemonInTeam` e `Item`, a cardinalità 0:1, che consente di avere anche Pokémon senza oggetto.

2.3 Schema E-R completo

Di seguito si allega lo schema E-R nel suo complesso, costruito con DB-Main:



3. Progettazione Logica

3.1 Stima del Volume Dati

La seguente tabella riassume le entità e relazioni principali del sistema, insieme a una stima realistica del numero di occorrenze che ci si aspetta di gestire. Le stime sono state formulate considerando non solo la situazione attuale, ma anche una possibile evoluzione futura dell'applicazione, con il supporto esteso a tutte e nove le generazioni Pokémon esistenti.

Nome	Tipo	Stima approssimativa
User	E	1000
Team	E	3000
Pokémon	E	2000
Type	E	20
VideoGame	E	30
PokémonInTeam	E	15000
Ability	E	300
Move	E	1500
Item	E	100
Statistics	E	2000
Pokedex	E	30
PokemonInPokedex	E	19200
Pokémon_Tipo	R	3000
Pokémon_Move	R	140000
Pokémon_Ability	R	2800
PokémonInTeam_Item	R	5000
has (Pokémon-Statistics)	R	2000
part_of (PokemonInPokedex-Pokedex)	R	19200
is (PokemonInPokedex-Pokémon)	R	19200
regional_pokedex	R	30
weakness (Type-Type)	R	324

Stima del volume per le ENTITÀ:

- **User**

Si prevede un utilizzo del sistema da parte di una base utenti medio-piccola ma attiva, con una stima prudenziale di circa 1000 utenti registrati. Il valore considera una diffusione moderata dell'app e una sua longevità nel tempo.

- **Team**

Ogni utente può creare più squadre. Stimando una media di 3 squadre per utente, si ottiene un totale di circa 3000 squadre registrate nel sistema.

- **Pokémon**

Anche se l'obiettivo iniziale era limitarsi alla seconda generazione, si è deciso di progettare il sistema pensando all'intero ecosistema Pokémon. Considerando le 9 generazioni attuali (con oltre 1000 Pokémon già noti) e una possibile espansione futura, la stima è stata portata a 2000 Pokémon totali.

- **Type**

Nei giochi Pokémon esistono attualmente 18 tipi base. Per tenere conto di possibili estensioni o varianti speciali, si considera una stima di 20 tipi complessivi.

- **VideoGame**

Seppur considerando i due titoli per ogni nuovo gioco come una versione unica, queste rimangono comunque numerose: Rosso-Blu, Oro-Argento, Rubino-Zaffiro, ecc. fino a Scarlatto-Violetto, incluse versioni speciali e remake. Dunque si stima un totale di circa 30 versioni gestite dal sistema.

- **PokémonInTeam**

Con 3000 squadre stimate e una media di 5 Pokémon per squadra (il massimo è 6, ma non tutti gli utenti lo raggiungeranno), si arriva a una previsione di circa 15.000 Pokémon in squadra.

- **Ability**

Un Pokémon può avere soltanto una e una sola abilità, ma un'abilità può essere comune a più Pokémon. Sommando tutte le abilità esistenti nelle 9 generazioni e volendosi tener larghi, si arriva a circa 300 abilità distinte.

- **Move**

I giochi Pokémon finora prevedono circa 900 diverse mosse, ma per una futura estendibilità e concretezza si è scelto di gestirne circa 1500 nel sistema, così da garantire una maggior flessibilità del database.

- **Item**

Si stimano circa 100 oggetti rilevanti tra strumenti da tenere, bacche, oggetti evolutivi, ecc., escludendo quelli non associabili direttamente ai Pokémon.

- **Statistics**

Ogni Pokémon è associato a un profilo di statistiche base (es. attacco, difesa, velocità). Il legame è 1:1 con l'entità Pokémon → 2000 record.

- **Pokedex**

Ogni versione di gioco dispone di un Pokédex regionale. Dunque, 30 Pokédex totali.

- **PokemonInPokedex**

Per modellare correttamente la disponibilità di ciascun Pokémon nelle varie versioni di gioco, si utilizza l'entità **PokemonInPokedex**, che rappresenta la presenza effettiva di un determinato Pokémon all'interno di uno specifico Pokédex regionale.

Non tutti i Pokémons sono presenti in tutte le versioni: alcuni, come Pikachu, compaiono in quasi tutti i giochi, mentre altri, soprattutto quelli delle generazioni più recenti, sono disponibili solo nella loro generazione di appartenenza.

Per ottenere una stima realistica del volume, si è considerata la seguente distribuzione:

- Il 20% dei Pokémons appare in circa 20 versioni,
- Il 50% in 10 versioni,
- Il restante 30% solo in 2 versioni.

Applicando questa distribuzione ponderata a 2000 Pokémons, si ottiene 19200 record.

Stima del volume per le RELAZIONI:

- **Pokémon_Tipo**

Ogni Pokémon ha almeno un tipo e al massimo due. In media, si può stimare 1,5 tipi per Pokémon. Applicando questa media ai 2000 Pokémons previsti, si ottengono circa 3000 record nella tabella che collega Pokémon e Tipo.

- **Pokémon_Move**

Ogni Pokémon può apprendere numerose mosse, spesso più di 50. Secondo delle ricerche svolte, il Pokémon medio può impararne circa 70 in totale, tenendo conto di tutti e tre i metodi di apprendimento (TM, livello, uovo), dunque: $2000 \times 70 = 140.000$ record.

- **Pokémon_Ability**

Considerando che ogni Pokémon abbia una sola abilità, si ottiene una stima di 2000.

- **PokémonInTeam_Item**

Non tutti i Pokémons in squadra tengono un oggetto. Stimando che circa un terzo dei 15.000 Pokémons presenti in squadra ne possieda uno, si ottengono circa 5000 record nella tabella di associazione tra PokémonInTeam e Item.

- **has (Pokémon-Statistics)**

Relazione 1:1 tra Pokémon e profilo di statistiche. Coincide con il numero di Pokémons → 2000.

- **part_of (PokemonInPokedex-Pokedex)**

Collega ogni **PokemonInPokedex** al **Pokedex** a cui appartiene. Volume identico alla cardinalità di **PokemonInPokedex** → 19200.

- **is (PokemonInPokedex-Pokémon)**

Ogni **PokemonInPokedex** rappresenta l'associazione di un Pokémon a un Pokédex. Il volume è pari a 19200.

- **regional_pokedex**

Relazione 1:1 tra ogni **Pokedex** e una **Versione**. Con 30 Pokédex si stimano 30 record.

- **weakness (Type-Type)**

Relazione binaria con attributo **multiplicator** che rappresenta l'efficacia tra tipi. Con 18 tipi

noti: $18 \times 18 = 324$ combinazioni possibili.

3.2 Operazioni Principali

Operazione	Descrizione	Frequenza stimata (per utente/settimana)
Registrazione utente	Creazione di un nuovo account tramite form	0.4
Login	Accesso al sistema con credenziali	7-20
Scelta della versione di gioco	Selezione della versione desiderata per costruire un team	5-15
Visualizzazione elenco Pokémon	Mostra i Pokémon disponibili nella versione scelta	5-15
Filtraggio Pokémon per tipo/gen.	Restringe l'elenco per facilitare la selezione	5-15
Creazione squadra	Inserimento di una nuova squadra	5-15
Assegnazione o rimozione oggetto	Gestione dello strumento tenuto da un Pokémon nella squadra	2-5
Modifica squadra	Rimozione/sostituzione di Pokémon nella squadra salvata	2-5
Eliminazione squadra	Rimozione definitiva di una squadra salvata	0.5-1
Salvataggio squadra	Conferma della composizione del team	5-15
Visualizzazione squadre salvate	Consultazione delle proprie squadre già create	7-20
Visualizzazione statistiche aggregate	Interrogazioni statiche aggregate sul sistema (funzionalità solo admin)	<1
Visualizzazione dettagli Pokémon	Mostra informazioni complete su un Pokémon selezionato dal team	10-30
Logout	Chiusura della sessione utente	7-20

3.3 Schemi di navigazione e tabelle degli accessi

Legenda

- **Concetto:** rappresenta una entità del modello E-R coinvolta nell'operazione.
- **Costrutto:** specifica se si tratta di una *Entità* o di una *Relazione* del database.
- **Numero di accessi:** indica quante volte l'elemento viene letto o scritto durante l'operazione.
- **Tipo:** indica la modalità di accesso:
 - **L** → Lettura (Read)
 - **S** → Scrittura (Write)
 - **D** → Cancellazione (Delete)
- **Peso settimanale:** riportato alla fine di ogni operazione, calcolato assegnando:
 - **1 punto** per ogni accesso in lettura (L)
 - **2 punti** per ogni accesso in scrittura (S) o cancellazione (D)
 - Moltiplicato per la frequenza stimata dell'operazione nella settimana

OP 1 - Registrazione utente

Il sistema consente all'utente di registrarsi compilando un form. I dati vengono salvati nel database.

Concetto	Costrutto	Numero di accessi	Tipo
User	Entità	1	S

Totale: 1S → 400 alla settimana → Peso: 800

OP 2 - Login

L'utente inserisce le credenziali e accede al sistema.

Concetto	Costrutto	Numero di accessi	Tipo
User	Entità	1	L

Totale: 1L → 15000 alla settimana → Peso: 15000

OP 3 - Scelta della versione di gioco

Durante la creazione o modifica di una squadra, l'utente seleziona una versione.

Concetto	Costrutto	Numero di accessi	Tipo
VideoGame	Entità	30	L
Pokedex	Entità	30	L
regional_pokedex	Relazione	30	L

Totale: 90L → 6000 alla settimana → Peso: 540000

OP 4 - Visualizzazione e filtraggio Pokémon

Il sistema mostra i Pokémon disponibili con possibilità di filtrarli.

Concetto	Costrutto	Numero di accessi	Tipo
PokemonInPokedex	Entità	400	L
Pokémon	Entità	400	L
Type	Entità	600	L
is	Relazione	400	L
pokemon_type	Relazione	400	L

Totale: 2200L → 15000 alla settimana → Peso: 33000000

OP 5 - Creazione nuova squadra

L'utente salva una nuova squadra, collegandola a versione e utente.

Concetto	Costrutto	Numero di accessi	Tipo
Team	Entità	3	S
User	Entità	1	L
VideoGame	Entità	1	L
Pokedex	Entità	1	L

Concetto	Costrutto	Numero di accessi	Tipo
owns	Relazione	1	S
regional_pokedex	Relazione	1	L

Totale: 4S, 4L → 3000 alla settimana → Peso: 24000

OP 6 - Composizione della squadra

L'utente seleziona Pokémon e oggetti per completare la squadra.

Concetto	Costrutto	Numero di accessi	Tipo
Pokémon	Entità	400	L
PokemonInPokedex	Entità	400	L
Type	Entità	600	L
Item	Entità	100	L
PokémonInTeam	Entità	6	S
holds	Relazione	6	S
is	Relazione	400	L
pokemon_type	Relazione	400	L
is_pokemon	Relazione	400	L
part_of	Relazione	400	L

Totale: 12S + 3100L → 8000 alla settimana → Peso: 24992000

OP 7 - Assegnazione o rimozione oggetto

L'utente assegna o rimuove un oggetto tenuto da uno specifico Pokémon nella squadra.

Concetto	Costrutto	Numero di accessi	Tipo
PokémonInTeam	Entità	1	L
Item	Entità	100	L/S
holds	Relazione	1	L/S

Totale: 1L + 101L/S → 3500 alla settimana → Peso: ≤353500

OP 8 - Modifica squadra

L'utente modifica Pokémon o oggetti nella squadra. Si ripresenta l'interfaccia di selezione come in fase di creazione.

Concetto	Costrutto	Numero di accessi	Tipo
Pokémon	Entità	400	L
PokemonInPokedex	Entità	400	L
Type	Entità	600	L
Item	Entità	100	L
PokémonInTeam	Entità	6	S
holds	Relazione	6	S
is	Relazione	400	L
pokemon_type	Relazione	400	L
is_pokemon	Relazione	400	L
part_of	Relazione	400	L

Totale: 12S + 3100L → 8000 alla settimana → Peso: 24992000

OP 9 - Eliminazione squadra

L'utente elimina una squadra salvata.

Concetto	Costrutto	Numero di accessi	Tipo
Team	Entità	1	D

Totale: 1D → 500 alla settimana → Peso: 1000

OP 10 - Salvataggio squadra

Conferma e salvataggio della composizione finale.

Concetto	Costrutto	Numero di accessi	Tipo
PokémonInTeam	Entità	≤ 6	S
Item	Entità	≤ 6	S
holds	Relazione	≤ 6	S

Totale: $\leq 18S \rightarrow 4000$ alla settimana → Peso: ≤ 72000

OP 11 - Visualizzazione squadre salvate

L'utente consulta le proprie squadre salvate.

Concetto	Costrutto	Numero di accessi	Tipo
Team	Entità	3	L
PokémonInTeam	Entità	3×6	L
Pokémon	Entità	3×6	L
is_pokemon	Relazione	18	L

Totale: $57L \rightarrow 8000$ alla settimana → Peso: 456000

OP 12 - Visualizzazione statistiche (admin)

L'amministratore esegue interrogazioni aggregate (es. numero medio di Pokémons per squadra, Pokémons più usati, tipi più frequenti).

Concetto	Costrutto	Numero di accessi	Tipo
Trainer	Entità	1000	L
Team	Entità	3000	L
PokémonInTeam	Entità	15000	L
Pokémon	Entità	2000	L
Type	Entità	20	L
owns	Relazione	3000	L
is_part	Relazione	15000	L

Concetto	Costrutto	Numero di accessi	Tipo
is_pokemon	Relazione	15000	L
pokemon_type	Relazione	2000	L

Totale: 21.020L (entità) + 35.000L (relazioni) = 56.020L → 10 alla settimana → Peso: 560.200

OP 13 - Visualizzazione dettagli di un Pokémon

L'utente seleziona un Pokémon in squadra per consultare tutte le sue informazioni, incluse: dettagli base, tipi, mosse apprendibili (per livello, TM/HM e uovo), Pokédex regionali in cui è disponibile, punti di forza e debolezza.

Concetto	Costrutto	Numero di accessi	Tipo
Pokémon	Entità	1	L
Type	Entità	2	L
Move	Entità	70	L
EggMove	Entità	20	L
LevelMove	Entità	25	L
TMHMMove	Entità	25	L
PokemonInPokedex	Entità	1	L
Pokedex	Entità	1	L
has_type	Relazione	2	L
weakness	Relazione	2	L
is_pokemon	Relazione	1	L
born_with	Relazione	20	L
learning	Relazione	25	L
teaching	Relazione	25	L

Totale: 170L (entità) + 100L (relazioni) = 270L → 30 alla settimana → Peso: 8100

OP 14 - Logout

L'utente termina la sessione.

Concetto	Costrutto	Numero di accessi	Tipo
—	—	—	—

Totale: 0 → 15000 alla settimana → Peso: 0

3.4 Raffinamento dello schema

In questa sezione si riportano le trasformazioni adottate per rendere lo schema concettuale compatibile con il modello relazionale. Tali trasformazioni riguardano l'eliminazione di elementi non supportati (come attributi composti, generalizzazioni e identificatori esterni) e un'analisi delle ridondanze.

Eliminazione delle gerarchie

Nel modello concettuale sono presenti due esempi di gerarchia, trattati come segue:

- **Admin / Trainer**

L'entità **Admin** è una specializzazione di **Trainer**.

In fase di raffinamento, questa distinzione è stata **semplificata** utilizzando un **attributo booleano**

is_admin nell'entità **Trainer**, che assume valore **true** solo per gli amministratori. Questa scelta evita l'uso di una gerarchia e rende il controllo dei permessi più diretto a livello applicativo.

- **TM/HM Move, Level Move, Egg Move / Move**

Le tre entità rappresentano tre diverse modalità tramite cui un Pokémon può apprendere una mossa (Move).

È stata adottata la tecnica della **sostituzione con associazioni**, secondo cui ogni modalità di apprendimento viene modellata come **una nuova entità** (**TMHMMove**, **LevelMove**, **EggMove**), contenente una chiave esterna verso **Move**.

Attributi composti o multivaleure

Anche gli attributi composti sono stati normalizzati. In particolare:

- **Version [1-2]** di **VideoGame**:

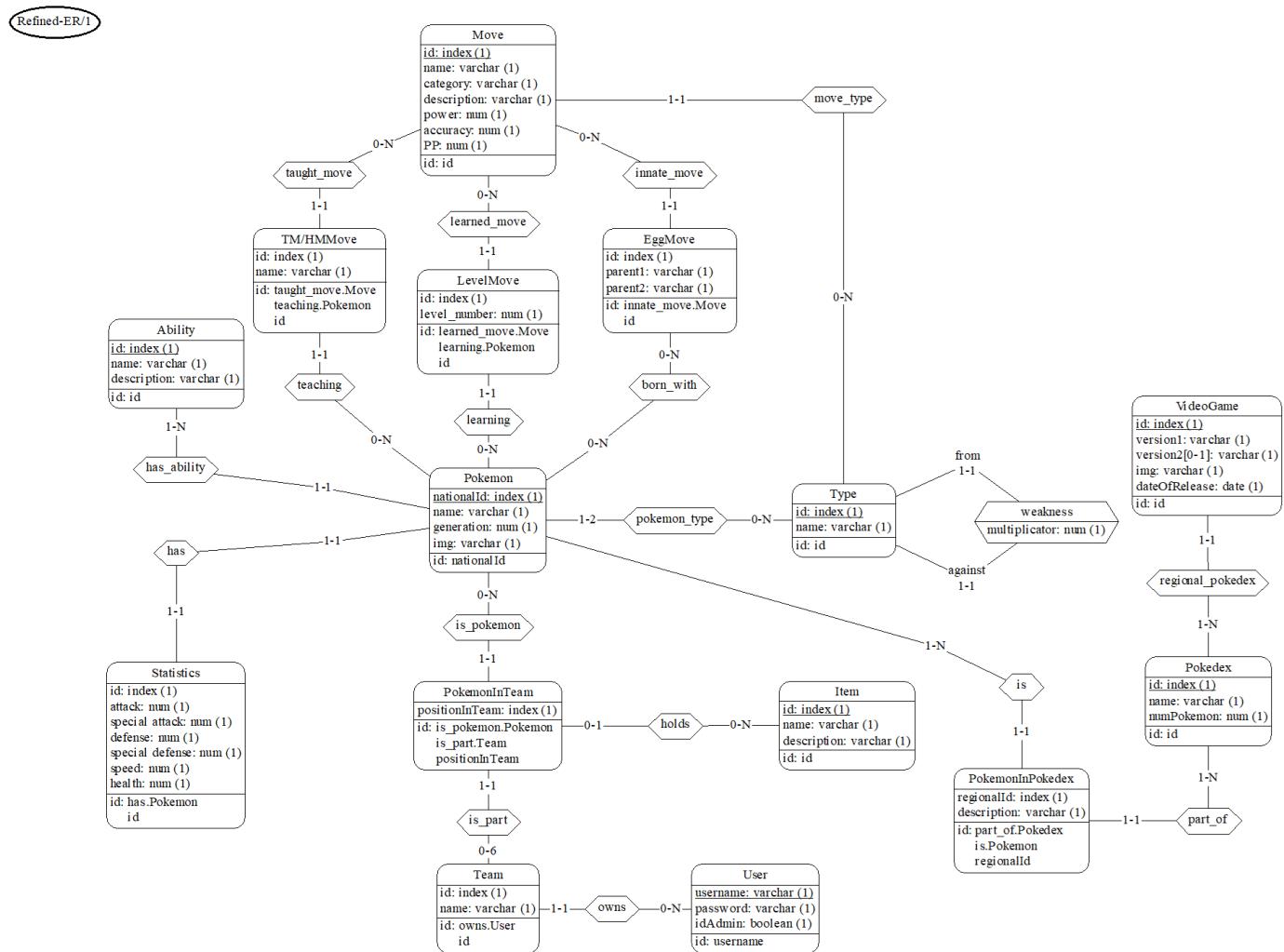
Poiché si tratta di una coppia di titoli (es. Rosso e Blu), l'attributo è stato **scisso** in due campi distinti: **version1** e **version2**, entrambi di tipo varchar.

- **Parents [2-2]** dell'entità **EggMove**:

Anche questo è un attributo composto rappresentante i due genitori compatibili. È stato **splittato**

in due attributi distinti: `parent1` e `parent2`, entrambi di tipo varchar.

Di seguito si riporta lo schema ER raffinato:



Eliminazione degli identificatori esterni

Nel passaggio dal modello E-R al modello logico relazionale, tutte le relazioni vengono eliminate e sostituite tramite **chiavi esterne** o **tabelle associative**, in base alla loro cardinalità.

- **owns (User → Team)**

La chiave primaria `username` dell'entità `User` viene importata in `Team` come chiave esterna `trainerUsername`.

- **is_part (PokemonInTeam → Team)**

L'identificatore `id` dell'entità `Team` viene importato in `PokemonInTeam` come chiave esterna `teamId`.

- **is_pokemon(PokemonInTeam → Pokémon)**

L'identificatore `nationalId` dell'entità `Pokémon` viene importato in `PokemonInTeam` come chiave esterna `pokemonId`.

- **holds (PokemonInTeam → Item)**

La chiave primaria `id` dell'entità `Item` viene importata in `PokemonInTeam` come chiave esterna

`itemId`.

La cardinalità 0:1 è rispettata rendendo il campo **nullable**.

- **has (Pokemon → Statistics)**

Il campo `pokemon_id` di `Statistics` funge da chiave esterna riferita a `Pokemon.nationalId`.

La relazione 1:1 è gestita posizionando la chiave esterna nella tabella dipendente `Statistics`.

- **has_ability (Pokemon → Ability)**

L'identificatore `id` dell'entità `Ability` viene importato in `Pokemon` come chiave esterna `abilityId`, riflettendo la relazione 1:1 (ogni Pokémon ha al massimo una abilità nel modello attuale).

- **pokemon_type (Pokemon ↔ Type)**

La relazione molti-a-molti è eliminata creando la tabella `pokemon_type`, che contiene le foreign key `pokemonId` e `typeId`.

- **weakness (Type ↔ Type)**

La relazione binaria tra tipi è trasformata nella tabella `weakness`, che contiene le foreign key `fromTypeId` e `againstTypeId`, più l'attributo `multiplicator`.

- **part_of (PokemonInPokedex → Pokedex)**

L'identificatore `id` dell'entità `Pokedex` è importato in `PokemonInPokedex` come chiave esterna `pokedexId`.

- **is (PokemonInPokedex → Pokemon)**

L'identificatore `nationalId` dell'entità `Pokemon` è importato in `PokemonInPokedex` come chiave esterna `pokemonId`.

- **regional_pokedex (Pokedex → VideoGame)**

La chiave esterna `videogame_id` è posizionata in `Pokedex` e fa riferimento alla chiave primaria `id` dell'entità `VideoGame`.

- **learned_move (LevelMove → Move) e learning (Pokemon → LevelMove)**

La chiave primaria `id` di `Move` è importata in `LevelMove` come `moveId`.

Analogamente, `LevelMove.pokemonId` è chiave esterna verso `Pokemon.nationalId`.

- **taught_move (TmHmMove → Move) e teaching (Pokemon → TmHmMove)**

La chiave primaria `id` di `Move` è importata in `TmHmMove` come `moveId`.

Analogamente, `TmHmMove.pokemonId` è chiave esterna verso `Pokemon.nationalId`.

- **innate_move (EggMove → Move) e born_with (Pokemon → EggMove)**

La chiave primaria `id` di `Move` è importata in `EggMove` come `moveId`.

L'entità `EggMove` rappresenta la definizione delle mosse apprese tramite uovo.

Per modellare l'associazione `born_with`, è stata introdotta una seconda entità `BornWith`, dove:

- `pokemonId` è una chiave esterna verso `Pokemon.nationalId`,
- `eggMoveId` è una chiave esterna verso `EggMove.id`,
- `moveId` è una chiave esterna ridondante verso `Move.id` (in quanto già implicita tramite `eggMoveId`, ma mantenuta per facilitare le interrogazioni).

Durante la progettazione logica è stata valutata la possibilità di introdurre **alcuni attributi ridondanti** al fine di ottimizzare prestazioni di lettura in operazioni frequenti. Si riportano qui due casi rappresentativi, uno che ha portato all'adozione della ridondanza, l'altro al suo rifiuto.

Caso 1 - Ridondanza accettata: `numPokemon` in `Pokedex`

In un primo momento si era previsto di calcolare dinamicamente il numero di Pokémon presenti in ciascun Pokédex tramite una query `COUNT(*)` sulla tabella `PokemonInPokedex`.

Tuttavia, questo approccio è stato considerato inefficiente: anche se il dato in sé non cambia mai, ogni accesso avrebbe richiesto una lettura completa della tabella `PokemonInPokedex`, con un costo significativo nel lungo periodo.

Considerazione sulle operazioni

Supponendo **15.000 accessi settimanali** a questa funzionalità, e stimando che ogni `COUNT(*)` comporti la lettura di circa **1.000 righe** (tutti i Pokémon di un Pokédex), si arriva a:

- $15.000 * 1.000 = 15.000.000$ operazioni di lettura a settimana.

Con una ridondanza controllata (campo `numPokemon`), il numero è già incluso nei dati letti in pagina, evitando ogni operazione aggiuntiva:

- $15.000 * 1 = 15.000$ letture totali (già necessarie per la visualizzazione).

Tabella di confronto

Aspetto	Senza ridondanza	Con ridondanza
Accessi settimanali	15.000	15.000
Operazioni di lettura totali	15.000.000	15.000
Dato aggiornabile	No	No
Rischio di inconsistenza	Nessuno	Nessuno

Conclusione

Poiché si tratta di un **dato statico** e non soggetto a modifiche né da parte dell'utente né da logiche di sistema, la ridondanza è **accettabile e vantaggiosa**. Essa consente di ridurre drasticamente il carico di lettura senza introdurre problemi di coerenza. Si è quindi deciso di **inserire il campo `numPokemon` nella tabella `Pokedex`** per migliorare l'efficienza del sistema.

Caso 2 - Ridondanza rifiutata: `numTeam` in `User`

In fase progettuale si era valutata la possibilità di memorizzare direttamente nel profilo utente il numero di squadre create (`numTeam`), così da poterlo consultare rapidamente, ad esempio nella sezione riservata agli amministratori.

Alternativa A - Senza ridondanza

Il dato può essere calcolato con una semplice `COUNT(*)` sulla tabella `Team`, filtrando per ciascun utente.

- Visualizzato solo in OP 12 (statistiche admin)
- Frequenza stimata: **10 accessi/settimana**
- In media, ogni accesso comporta **una COUNT(*) per ciascun utente presente** (ad esempio, 3 squadre per utente → 3 letture)
- **Costo settimanale stimato:** `10 accessi * 3 letture = 30 letture`

Alternativa B - Con ridondanza

Memorizzando `numTeam` in `User`, si eviterebbero letture ma si dovrebbero aggiornare i dati:

- Ogni volta che un utente **crea o elimina una squadra**
- Le operazioni di questo tipo sono frequenti (stimati **8500 aggiornamenti/settimana**)
- **Costo settimanale stimato: 8500 scritture**

Tabella di confronto

Aspetto	Alternativa A (senza ridondanza)	Alternativa B (con ridondanza)
Query <code>COUNT(*)</code> eseguite	~30/settimana	0
Scritture aggiuntive settimanali	0	8.500
Complessità di implementazione	Minima	Alta (aggiornamenti frequenti)
Rischio di inconsistenza	Nessuno	Alto (se non mantenuto correttamente)

Conclusione

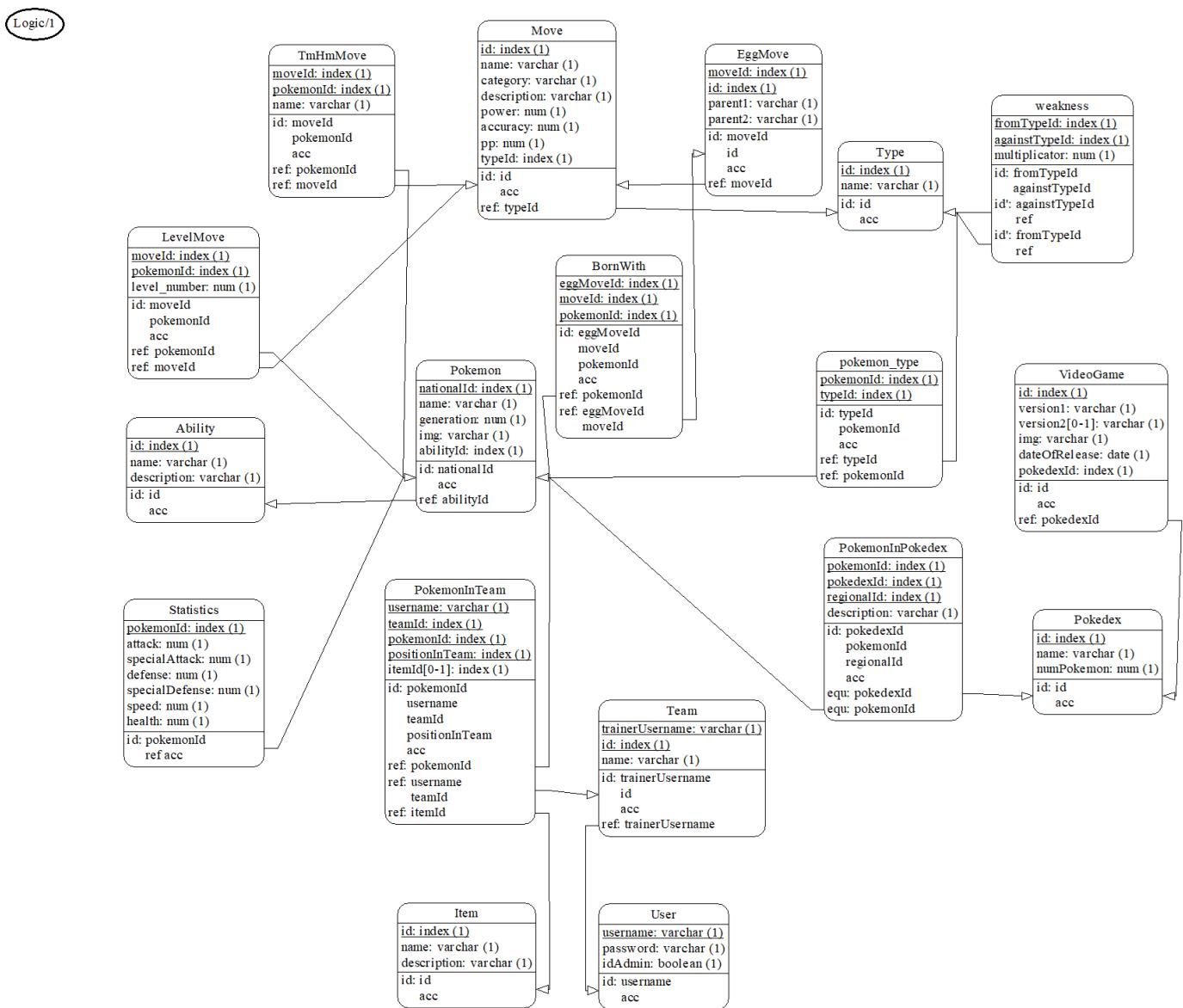
Poiché il dato è consultato **raramente** e può essere calcolato dinamicamente con **costo molto contenuto**, mentre la sua gestione tramite aggiornamento diretto sarebbe onerosa e soggetta a errori, si è deciso di **non introdurre la ridondanza**. La lettura dinamica del numero di squadre è preferibile in questo contesto.

Traduzione di entità e associazioni in relazioni

- **User**(username, password, isAdmin)
 - **Team**(id, name, trainerUsername)
 - FK: trainerUsername REFERENCES UTENTE
 - **Pokemon**(nationalId, name, generation, img, abilityId)
 - **Type**(id, name)
 - **Ability**(id, name, description)
 - **Move**(id, name, typeId, power, accuracy, category, pp, description)
 - FK: typeId REFERENCES TYPE
 - **Statistics**(attack, defense, speicalAttack, specialDefense, speed, health, pokemonId)
 - FK: pokemonId REFERENCES POKEMON
 - **Item**(id, name, description)
 - **PokemonInTeam**(positionInTeam, teamId, pokemonId, username, itemId)
 - FK: teamId REFERENCES TEAM
 - FK: pokemonId REFERENCES POKEMON
 - FK: itemId REFERENCES ITEM
 - **VideoGame**(id, version1, version2, dateOfRelease, img, pokedexId)
 - **Pokedex**(id, name, numPokemon)
 - **PokemonInPokedex**(regionalId, pokedexId, pokemonId, description)
 - FK: pokedexId REFERENCES POKEDEX
 - FK: pokemonId REFERENCES POKEMON
 - **pokemon_type**(pokemonId, typeId)
 - FK: pokemonId REFERENCES POKEMON
 - FK: typeId REFERENCES TYPE
 - **weakness**(fromTypeId, againstTypeId, multiplicator)
 - FK: fromTypeId REFERENCES TYPE
 - FK: againstTypeId REFERENCES TYPE
 - **LevelMove**(pokemonId, moveId, levelNumber)
 - FK: pokemon_id REFERENCES POKEMON
 - FK: move_id REFERENCES MOVE

- **EggMove**(moveId, id, parent1, parent2)
FK: moveId REFERENCES MOVE
- **BornWith**(moveId, eggMoveId, pokemonId)
FK: moveId REFERENCES MOVE
FK: eggMoveId REFERENCES EGGMOVE
FK: pokemonId REFERENCES POKEMON
- **TmHmMove**(pokemonId, moveId, name)
FK: pokemonId REFERENCES POKEMON
FK: moveId REFERENCES MOVE

Di seguito si allega lo schema logico risultante:



4. Traduzione delle operazioni in query SQL

OP 1 - Registrazione utente

```
INSERT INTO User (username, password, isAdmin)
VALUES ('ashketchum', 'pika123', false);
```

OP 2 - Login

```
SELECT username, password
FROM User
WHERE username = 'ashketchum' AND password = 'pika123';
```

OP 3 - Scelta della versione di gioco

```
SELECT
CASE
    WHEN version2 IS NULL THEN version1
    ELSE CONCAT(version1, '/', version2)
END AS version_name
FROM VideoGame;
```

OP 4 - Visualizzazione elenco Pokémon

```
-- elenco di TUTTI i Pokémon che appaiono in una versione di gioco:
SELECT Pokemon.name
FROM Pokemon
JOIN PokemonInPokedex ON Pokemon.nationalId = PokemonInPokedex.pokemonId
JOIN Pokedex ON PokemonInPokedex.pokedexId = Pokedex.id
JOIN VideoGame ON Pokedex.id = VideoGame.pokedexId
WHERE VideoGame.version1 = 'Cristallo' OR VideoGame.version2 = 'Cristallo';
```

```
-- (FILTRAGGIO) elenco di solo i Pokémon DI TIPO ACQUA che appaiono in una versione di gioco:  
SELECT Pokemon.name  
FROM Pokemon  
JOIN PokemonInPokedex ON Pokemon.nationalId = PokemonInPokedex.pokemonId  
JOIN Pokedex ON PokemonInPokedex.pokedexId = Pokedex.id  
JOIN VideoGame ON Pokedex.id = VideoGame.pokedexId  
JOIN pokemon_type ON Pokemon.nationalId = pokemon_type.pokemonId  
JOIN Type ON pokemon_type.typeId = Type.id  
WHERE (VideoGame.version1 = 'Cristallo' OR VideoGame.version2 = 'Cristallo')  
AND Type.name = 'Acqua';
```

```
-- (FILTRAGGIO) elenco di solo i Pokémon DI 2a GENERAZIONE che appaiono in una versione di gioco:  
SELECT Pokemon.name  
FROM Pokemon  
JOIN PokemonInPokedex ON Pokemon.nationalId = PokemonInPokedex.pokemonId  
JOIN Pokedex ON PokemonInPokedex.pokedexId = Pokedex.id  
JOIN VideoGame ON Pokedex.id = VideoGame.pokedexId  
WHERE (VideoGame.version1 = 'Cristallo' OR VideoGame.version2 = 'Cristallo')  
AND Pokemon.generation = 2;
```

OP 5 - Creazione nuova squadra

```
-- inserimento di un nuovo team  
INSERT INTO Team (name, trainerUsername)  
VALUES ('Team Nuzlocke', 'ashketchum');
```

OP 6 -Composizione della squadra

```
-- popolamento del team inserendo tre nuovi Pokémon e i loro eventuali oggetti  
INSERT INTO PokemonInTeam (teamId, pokemonId, itemId)  
VALUES  
((SELECT id FROM Team WHERE name = 'Team Nuzlocke' AND trainerUsername =
```

```
'ashketchum'), 1, NULL),  
((SELECT id FROM Team WHERE name = 'Team Nuzlocke' AND trainerUsername =  
'ashketchum'), 61, 79),  
((SELECT id FROM Team WHERE name = 'Team Nuzlocke' AND trainerUsername =  
'ashketchum'), 25, 72);
```

OP 7 - Assegnazione o rimozione oggetto

```
-- modifica dell'oggetto:  
UPDATE PokemonInTeam  
SET itemId = 24  
WHERE teamId = 101 AND positionInTeam = 3;
```

```
-- rimozione dell'oggetto:  
UPDATE PokemonInTeam  
SET itemId = NULL  
WHERE teamId = 101 AND positionInTeam = 3;
```

OP 8 - Modifica Squadra

```
-- modifica di un Pokémon della squadra:  
UPDATE PokemonInTeam  
SET pokemonId = 94  
WHERE teamId = 101 AND positionInTeam = 2;
```

```
-- rimozione di un Pokémon dalla squadra:  
DELETE FROM PokemonInTeam  
WHERE teamId = 101 AND positionInTeam = 2;
```

OP 9 - Eliminazione di una squadra

```
DELETE FROM Team  
WHERE trainerUsername = 'ashketchum' AND id = 3;
```

OP 10 - Salvataggio della squadra

```
INSERT INTO PokemonInTeam (teamId, pokemonId, itemId)  
VALUES  
((SELECT id FROM Team WHERE name = 'Team Nuzlocke' AND trainerUsername =  
'ashketchum'), 1, NULL),  
((SELECT id FROM Team WHERE name = 'Team Nuzlocke' AND trainerUsername =  
'ashketchum'), 61, 79),  
((SELECT id FROM Team WHERE name = 'Team Nuzlocke' AND trainerUsername =  
'ashketchum'), 25, 72);
```

OP 11 - Visualizzazione squadre salvate

```
SELECT Team.name AS team_name,  
       Pokemon.name AS pokemon_name,  
       Item.name AS item_name  
FROM Team  
JOIN PokemonInTeam ON Team.id = PokemonInTeam.teamId  
JOIN Pokemon ON PokemonInTeam.pokemonId = Pokemon.nationalId  
LEFT JOIN Item ON PokemonInTeam.itemId = Item.id  
WHERE Team.trainerUsername = 'ashketchum'  
ORDER BY Team.name, PokemonInTeam.positionInTeam ASC;
```

OP 12 - Visualizzazione statistiche (admin)

```
-- Numero medio di Pokémon per squadra:  
SELECT AVG(num_pokemon) AS media_pokemon_per_team  
FROM (  
    SELECT COUNT(*) AS num_pokemon  
    FROM PokemonInTeam  
    GROUP BY teamId  
) AS conteggi;
```

```
-- Numero medio di squadre per utente:  
SELECT AVG(num_team) AS media_team_per_user  
FROM (  
    SELECT COUNT(*) AS num_team  
    FROM Team  
    GROUP BY trainerUsername  
) AS conteggi;
```

```
-- Pokémon che compare in più team in assoluto:  
SELECT Pokemon.name, COUNT(*) AS num_occurrences  
FROM PokemonInTeam  
JOIN Pokemon ON PokemonInTeam.pokemonId = Pokemon.nationalId  
GROUP BY Pokemon.name  
ORDER BY num_occurrences DESC  
LIMIT 1;
```

```
-- Tipo che compare in più team in assoluto:  
SELECT Type.name, COUNT(*) AS num_occurrences  
FROM PokemonInTeam  
JOIN Pokemon ON PokemonInTeam.pokemonId = Pokemon.nationalId  
JOIN pokemon_type ON pokemon_type.pokemonId = Pokemon.nationalId  
JOIN Type ON Type.id = pokemon_type.typeId  
GROUP BY Type.name  
ORDER BY num_occurrences DESC  
LIMIT 1;
```

OP 13 - Visualizzare info Pokémon

----- INFO POKEMON CON ID=25 -----

```
-- Informazioni base del Pokémon  
SELECT *  
FROM Pokemon  
WHERE nationalId = 25;
```

```
-- Tipi del Pokémon
SELECT Type.name
FROM pokemon_type
JOIN Type ON pokemon_type.typeId = Type.id
WHERE pokemon_type.pokemonId = 25;

-- Pokédex regionali in cui è presente
SELECT VideoGame.version1, VideoGame.version2, Pokedex.name
FROM PokemonInPokedex
JOIN Pokedex ON PokemonInPokedex.pokedexId = Pokedex.id
JOIN VideoGame ON Pokedex.id = VideoGame.pokedexId
WHERE PokemonInPokedex.pokemonId = 25;

-- Mosse apprese salendo di livello
SELECT Move.name, LevelMove.levelNumber
FROM LevelMove
JOIN Move ON LevelMove.moveId = Move.id
WHERE LevelMove.pokemonId = 25
ORDER BY LevelMove.levelNumber;

-- Mosse apprese tramite TM/HM
SELECT Move.name
FROM TmHmMove
JOIN Move ON TmHmMove.moveId = Move.id
WHERE TmHmMove.pokemonId = 25;

-- Mosse apprese tramite uovo
SELECT Move.name
FROM EggMove
JOIN Move ON EggMove.moveId = Move.id
WHERE EggMove.pokemonId = 25;

-- Tipi contro cui il Pokémon è forte
SELECT DISTINCT T2.name AS target_type
FROM Pokemon
JOIN pokemon_type pt1 ON Pokemon.nationalId = pt1.pokemonId
JOIN Type T1 ON pt1.typeId = T1.id
JOIN weakness ON weakness.fromTypeId = T1.id
```

```
JOIN Type T2 ON weakness.againstTypeId = T2.id  
WHERE Pokemon.nationalId = 25 AND weakness.multipliator > 1;
```

-- Tipi contro cui il Pokémon è debole

```
SELECT DISTINCT T2.name AS target_type  
FROM Pokemon  
JOIN pokemon_type pt1 ON Pokemon.nationalId = pt1.pokemonId  
JOIN Type T1 ON pt1.typeId = T1.id  
JOIN weakness ON weakness.fromTypeId = T1.id  
JOIN Type T2 ON weakness.againstTypeId = T2.id  
WHERE Pokemon.nationalId = 25 AND weakness.multipliator < 1;
```

OP 14 - Logout

```
--  
-- Non richiede una query  
--
```

5. Progettazione dell'Applicazione

5.1 Architettura

L'architettura dell'applicazione web è di tipo **client-server**, in cui il **frontend** interagisce con il **backend** per la gestione dei dati e l'interfaccia utente.

L'applicazione è stata realizzata utilizzando:

- **PHP** come linguaggio di programmazione lato server;
 - **MySQL** come sistema di gestione di base di dati relazionale (RDBMS);
 - **HTML, CSS e Bootstrap** per il frontend e il design responsivo;
 - **JavaScript (opzionalmente con fetch API)** per alcune operazioni asincrone lato client;
 - **Sessioni PHP** per la gestione dell'autenticazione utente.
-

Componenti principali dell'architettura:

1. Database MySQL

Il cuore dell'applicazione è il database relazionale, progettato per gestire Pokémon, squadre, allenatori, tipi, mosse, Pokédex e le relazioni tra queste entità.

Tutte le interazioni di salvataggio, recupero e aggiornamento dei dati passano tramite query SQL ben strutturate e sicure.

2. Strato di accesso ai dati (`DbHelper.php`)

È stato realizzato un helper PHP per encapsulare la logica di connessione e accesso al database. Questo approccio favorisce la manutenibilità e separazione delle responsabilità.

3. Backend (PHP)

Il backend gestisce le logiche applicative principali:

- autenticazione e autorizzazione (inclusa la distinzione tra utenti e admin);
- gestione delle squadre Pokémon;
- estrazione delle informazioni dettagliate per ogni Pokémon;
- visualizzazione dinamica delle mosse, statistiche e match-up di tipo.

4. Frontend (HTML/CSS + Bootstrap)

Il frontend fornisce un'interfaccia utente semplice e intuitiva per:

- navigare tra i Pokémon;
- visualizzare e creare squadre;
- accedere all'area riservata per amministratori;
- consultare dati aggregati/statistici.

5. Controllo degli accessi

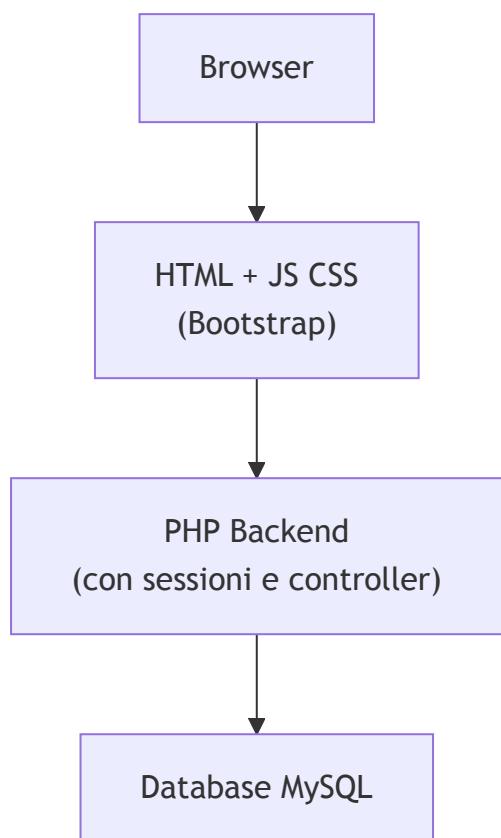
È previsto un meccanismo di controllo basato su sessione. L'attributo `isAdmin` nella tabella `Trainer` permette di distinguere gli utenti con privilegi speciali, che possono accedere ad una pagina amministrativa con statistiche aggregate.

6. Sicurezza

Tutti i dati sensibili sono gestiti con attenzione:

- le password sono salvate in formato hashato;
 - le query utilizzano **prepared statements** per prevenire attacchi SQL Injection;
 - l'accesso a pagine riservate è protetto tramite controlli di sessione.
-

Schema semplificato:



Questa architettura modulare ha permesso di garantire **scalabilità, manutenibilità e sicurezza** nell'implementazione dell'intero progetto.

5.2 Schermate

Di seguito sono presentati alcuni screenshots dimostrativi dell'app:



The main interface of the app, titled "PokéTeam". At the top right are links for "Home", "Squadre", and "Logout". A central circular icon contains a white "O". Below it, the text "Selezione gioco:" is displayed in red. The screen is organized into a grid of game covers. Row 1: Rosso / Blu (Game Boy), Giallo (Game Boy), Oro / Argento (Game Boy Color), Cristallo (Game Boy Color), Rubino / Zaffiro (Game Boy Advance), Smeraldo (Game Boy Advance). Row 2: Diamante / Perla (NDS), Platino (NDS), Bianco / Nero (NDS), Bianco 2 / Nero 2 (NDS), X / Y (3DS), Sole / Luna (3DS). Row 3: Ultrasole / Ultraluna (3DS).



Componi il tuo team:

**SALVA TEAM**Generazione: [Tutte](#) [Tipi](#): [Tutti](#)

#155 Cyndaquil

Generazione: 2 Presente nei Pokédex: Johto, Hoenn

Tipo: Fire Abilità: Blaze

**STATISTICHE BASE:**

Attacco: 52	Attacco Speciale: 60	Difesa: 43
Difesa Speciale: 50	Velocità: 65	Salute: 39

Debole contro: Water, Rock, Ground**Superefficace contro:** Ice, Bug, Grass, Steel

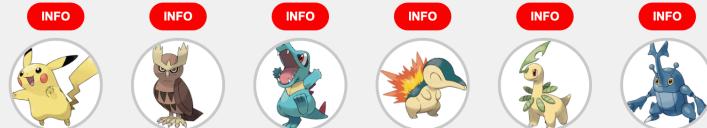
MOSSE IMPARABILI:

Mosse per livello:



Le tue squadre:

Johto Champions League

[EDIT](#)[DELETE](#)

Advanced Battle



! ADMIN ONLY !

Statistica	Valore stimato
Numero medio di Pokémon per squadra	5.21
Numero medio di squadre per utente	5.67
Pokémon più presente nelle squadre	Pikachu (7 squadre)
Tipo più presente nelle squadre	Water (40 occorrenze)