Relazione NASA Near-Earth Object Analytics

Gabriele Golfieri

Indice

1 Progettazione del Database

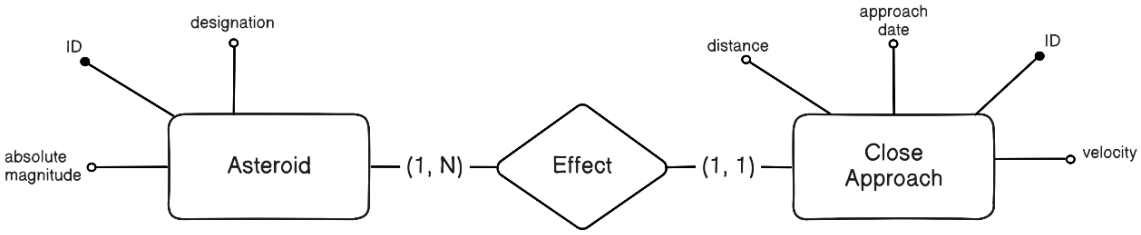

Figura 1: Schema ER

Entità Principali

- **Asteroid**: Rappresenta l'anagrafica dell'oggetto celeste. Contiene:
  - `ID` (Chiave Primaria).
  - `designation`: L'identificativo univoco fornito dalla NASA.
  - `absolute_magnitude`: La grandezza fisica dell'oggetto.
- **Close Approach**: Rappresenta lo storico e le previsioni dei passaggi vicini alla Terra. Contiene:
  - `ID` (Chiave Primaria).
  - `distance`: La distanza minima dalla Terra.
  - `approach_date`: Il timestamp esatto dell'evento.
  - `velocity`: La velocità relativa dell'asteroide.

  Nota di implementazione: È stato inserito un vincolo di unicità sulla coppia `(asteroid_id, approach_date)`. Questa accortezza previene la creazione di duplicati nel database durante le fasi di aggiornamento massivo dei dati.

2 API Selezionata e Motivazioni

Per lo sviluppo della piattaforma è stata scelta la *Close-Approach Data (CAD) API* del database SBDB della NASA. Le motivazioni principali dietro questa scelta sono due:

- **Divulgazione e Sicurezza Planetaria**: Il monitoraggio dei Near-Earth Objects (NEO) e degli asteroidi "Potenzialmente Pericolosi" (PHA) mi sembra un tema di forte fascino oltre a definire un reale bisogno di sicurezza globale.
- **Dinamicità dei Dati**: Le stime orbitali della NASA sono in continua evoluzione. Lavorare con un dataset dinamico mi ha permesso di implementare logiche di pulizia dei dati e prevenire presenza di duplicati, garantendo anche che la dashboard mostri sempre i calcoli di distanza e velocità più recenti.

3 Architettura della Soluzione

L'architettura del progetto separa in modo netto la gestione dei dati dalla loro visualizzazione, ed è suddivisa in quattro componenti principali:

3.1 Backend (Python & FastAPI)

È il motore che gestisce le richieste e comunica con la NASA. Il codice non è scritto in un unico file gigante, ma è stato organizzato in moduli separati (ad esempio: uno per la connessione al database, uno per le rotte dell'API e uno script `ingest.py` dedicato esclusivamente allo scaricamento iniziale dei dati). Ho scelto FastAPI perché è estremamente veloce e gestisce in modo eccellente le operazioni asincrone, una caratteristica fondamentale per scaricare ed elaborare migliaia di record senza bloccare il server.

3.2 Frontend (React + Vite)

È l'interfaccia utente, costruita come un'applicazione a singola pagina. Vite è stato utilizzato come tool di build per rendere l'avvio e lo sviluppo più rapidi. L'obiettivo principale di questa sezione è la reattività: grazie a React e alla libreria Recharts, l'utente può visualizzare i trend astronomici e filtrare centinaia di righe in tempo reale, senza mai dover ricaricare la pagina o subire fastidiosi rallentamenti.

3.3 Database (PostgreSQL)

È il database relazionale in cui vengono salvati tutti i dati storici e futuri. Avere un database proprietario evita di dover chiamare continuamente le API esterne della NASA, rendendo il caricamento della dashboard quasi istantaneo. Inoltre, PostgreSQL garantisce l'integrità delle informazioni: grazie ai vincoli impostati sulle tabelle, i dati possono essere aggiornati infinite volte senza mai generare duplicati.

3.4 Cloud & Deploy (Render)

L'intero progetto è stato pubblicato online utilizzando la piattaforma cloud Render. Invece di configurare i server a mano, ho optato per un approccio automatizzato tramite il file `render.yaml`. In questo modo, il sistema sa esattamente come costruire e avviare sia il backend che il frontend, rendendo i futuri rilasci di codice completamente automatici e sicuri.

4 Query Principali e Logica SQL

All'interno del codice sono state scritte query suddivise tra operazioni di scrittura (incapsulate in funzioni specifiche) e operazioni di lettura e controllo (eseguite direttamente nelle rotte dell'API):

- `save_asteroid`: Questa funzione gestisce l'inserimento dei dati identificativi e fisici dei Near Earth Objects (NEO).

```
INSERT INTO asteroids (id, name, absolute_magnitude, estimated_diameter_max)
VALUES (p_id, p_name, p_mag, p_dia)
ON CONFLICT (id)
DO UPDATE SET
    name = EXCLUDED.name,
    absolute_magnitude = EXCLUDED.absolute_magnitude,
    estimated_diameter_max = EXCLUDED.estimated_diameter_max;
```

- **save_approach**: Questa funzione gestisce l'inserimento degli eventi di avvicinamento. Utilizza ON CONFLICT DO UPDATE per mantenere i dati sempre aggiornati alle ultime stime NASA.

```
INSERT INTO close_approaches (asteroid_id, approach_date, distance_au, velocity_km_s)
VALUES (p_ast_id, p_date, p_dist, p_vel)
ON CONFLICT (asteroid_id, approach_date)
DO UPDATE SET
    distance_au = EXCLUDED.distance_au,
    velocity_km_s = EXCLUDED.velocity_km_s;
```

- **Query di recupero dati combinati**: Questa query è il motore principale per la visualizzazione dei dati sulla dashboard. Poiché le informazioni sono normalizzate in due tabelle separate, viene utilizzata una clausola JOIN per ricongiungere l'identità dell'asteroide con i dettagli del suo avvicinamento.

```
SELECT a.designation, a.absolute_magnitude, c.approach_date, c.distance_au, c.velocity_km_s
FROM asteroids a
JOIN close_approaches c ON a.id = c.asteroid_id
ORDER BY c.approach_date DESC;
```

- **Query di verifica dello stato del database**: Questa istruzione interroga il database per contare il numero totale di record presenti nella tabella principale asteroids.

```
SELECT COUNT(*) FROM asteroids;
```

5 Insight Ricavati

L'esplorazione analitica e visiva dei dati esposti dalla dashboard ha permesso di isolare pattern fondamentali relativi al comportamento dei Near Earth Objects (NEO):

- **Distribuzione del Rischio e Soglie di Allerta**: L'analisi dei dati conferma che una porzione statisticamente rilevante degli oggetti monitorati passa vicino alla Terra a distanze inferiori della soglia critica di 0.05 Unità Astronomiche (AU). Questa informazione, resa immediatamente chiara dai grafici della dashboard, permette di identificare tali corpi celesti di interesse astronomico prioritario, assimilandoli per livello di rischio ai cosiddetti Potentially Hazardous Asteroids (PHA).
- **Frequenza degli Avvicinamenti**: L'osservazione del flusso dei dati, visibile attraverso l'aggiornamento dei record nelle tabelle della dashboard, smentisce l'immaginario comune che relega i passaggi ravvicinati a eventi rari. La presenza quotidiana di nuove rilevazioni evidenzia un volume costante di approcci.

6 Riflessione Personale

Un'esperienza fondamentale per il mio approccio allo sviluppo è stata la realizzazione di un precedente progetto web utilizzando PHP, HTML e Bootstrap. In quell'occasione avevo costruito il database partendo da zero, scegliendo e inserendo manualmente ogni singolo record. È stato un ottimo allenamento: mi sono divertito a sperimentare con la UI, cercando il modo visivo più efficace e accattivante per presentare le informazioni all'utente. Tuttavia, l'inserimento dei dati era statico e chiuso.

L'esperienza passata quindi mi ha aiutato a curare la dashboard in modo intuitivo, ma l'uso delle API mi ha spinto a progettare per la prima volta logiche di ingestion e aggiornamento lato database, unendo la cura del frontend alla dinamicità di un backend connesso all'esterno.

7 Proposta Innovativa

Attualmente, l'aggiornamento del database è gestito su richiesta tramite un'azione esplicita nell'interfaccia. Una naturale evoluzione architetturale sarebbe introdurre un meccanismo di aggiornamento automatico che, senza bisogno di alcun intervento manuale da parte dell'utente, interrogherebbe periodicamente i server della NASA. A questa automazione si potrebbe legare uno sviluppo di un modulo di alerting dinamico: qualora il sistema intercetti un nuovo avvicinamento sotto una soglia critica (es. < 0.01 AU), innescherebbe l'invio immediato di una notifica all'utente, trasformando il software in uno strumento di monitoraggio costante.