POWERBRICKS

Specifiche di Progetto & Architettura del Visual Query Builder

1. Sintesi del Progetto

PowerBricks è un'applicazione desktop leggera pensata per consentire a utenti non tecnici (analisti, manager, colleghi di reparto) di interrogare autonomamente e liberamente basi dati complesse senza conoscere la sintassi SQL. Il sistema si basa sul paradigma della programmazione visuale a blocchi (ispirato a Scratch/Blockly), traducendo le composizioni geometriche degli utenti in query SQL standard ottimizzate ed eseguendole direttamente in locale.

2. Architettura Tecnologica (Senza Browser)

Per garantire massima reattività sui dati locali, sicurezza e una distribuzione semplificata senza infrastrutture web complesse, l'applicazione adotta un approccio **ibrido Desktop/Web-View**:

- **Frontend (Interfaccia Utente):** Sviluppato in HTML, CSS e JavaScript sfruttando la libreria open-source **Google Blockly**. Questa scelta permette di delegare a un motore solido la gestione degli incastri logici, dei vincoli di forma e dei colori dei mattoncini.
- **Backend (Motore di Calcolo):** Gestito in **Python con Flask** configurato come un'API REST pura locale. Flask non effettua il rendering di pagine web storiche, ma riceve esclusivamente le strutture ad albero dei blocchi in formato JSON, occupandosi della validazione, della traduzione in SQL e dell'interazione con i file di dati.
- **Contenitore Desktop (PyWebView):** Utilizzato per incapsulare il frontend all'interno di una finestra desktop nativa isolata. Sotto la scocca viene creato un ponte diretto tra JavaScript (*pywebview.api*) e il backend Python, eliminando la necessità di aprire un browser web tradizionale.

3. Ambito Dati e Gestione dei Domini (Caso Mediaset)

L'applicativo è configurato per operare sulla base dati aziendale, organizzata inizialmente in 4 macro-domini logici. Ciascun dominio viene rappresentato visivamente da mattoncini dedicati con colorazioni e vincoli specifici:

Dominio	Contenuto Informativo	Esempio di Campi / Blocchi
Anagrafica	Metadati dei contenuti, informazioni strutturali di programmi, film e serie. Fuoco centrale delle relazioni.	Titolo, Regista, Anno, Genere, ID_Titolo, Codice_Opera
Emesso	Storico delle messe in onda, canali, palinsesti e dettagli temporali di trasmissione.	Data_Canale, Ora_Inizio, Canale, Durata, ID_Titolo
Diritti	Informazioni contrattuali, licenze di sfruttamento, scadenze e vincoli legali di trasmissione.	Codice_Contratto, Data_Inizio_Validità, Data_Scadenza
Box Office	Dati economici, incassi cinematografici, performance commerciali e biglietteria delle opere.	Incasso_Totale, Presenze, Codice_Opera, Weekend_Apertura

4. Risoluzione Automatica delle JOIN (Il Motore di Traduzione)

> **Il Principio Cardine:** L'utente finale non deve conoscere i concetti di `LEFT JOIN` o le chiavi primarie/esterne. La complessità relazionale viene risolta interamente "sotto la scocca" dal backend Python.

Attraverso un **Grafo delle Relazioni** hardcoded nel backend Python, PowerBricks analizza l'elenco dei campi richiesti e dei filtri applicati nei blocchi. Se l'utente unisce elementi appartenenti a domini differenti, Python ricostruisce automaticamente i passaggi relazionali necessari.

Al fine di evitare la perdita di dati nel caso in cui un'entità sia presente in un dominio ma non ancora censita in un altro (es. un film d'autore presente in Anagrafica ma mai passato al cinema e quindi assente in Box Office), il traduttore utilizzerà sistematicamente clausole di **`LEFT JOIN`** facenti perno sul dominio principale (Anagrafica).

Esempio di Flusso Dati (JSON a SQL)

Quando l'utente incastra i blocchi per richiedere il Titolo (Anagrafica) e l'Incasso (Box Office) filtrando per genere, il frontend genera e invia a Python il seguente oggetto:

```
{
  "campi_richiesti": ["Anagrafica.Titolo", "BoxOffice.Incasso_Totale"],
  "filtri": [{"campo": "Anagrafica.Genere", "operatore": "=", "valore": "Cinema"}]
}
```

Il backend Python interviene applicando le regole del grafo ed esegue automaticamente la compilazione del testo SQL standard ottimizzato per motori locali (come **SQLite** o per interrogazioni dirette su file **Parquet** tramite librerie ad alte prestazioni come *DuckDB*):

```
SELECT
    Anagrafica.Titolo,
    BoxOffice.Incasso_Totale
FROM Anagrafica
LEFT JOIN BoxOffice ON Anagrafica.Codice_Opera = BoxOffice.Codice_Opera
WHERE Anagrafica.Genere = 'Cinema';
```

5. Vantaggi e Sviluppi Futuri

- **Zero Errori di Sintassi:** L'interfaccia a blocchi impedisce fisicamente il collegamento di tipi di dati incompatibili (es. impossibile inserire un testo dentro un filtro di data).
- **Prestazioni Locali Elevate:** L'adozione combinata di Parquet e SQLite permette di processare milioni di righe in frazioni di secondo direttamente sul PC dell'utente, senza sovraccaricare i server centrali.
- **Flessibilità Multidominio:** Gli utenti possono porre domande complesse trasversali ai 4 domini muovendo semplicemente il mouse.