Movie Database Management System - System Documentation

1. Database Schema

The database consists of the following tables:

- **genres (id, genre_name)**: Stores unique movie genres.
- **directors (id, director_name)**: Stores movie directors.
- **actors (id, actor_name, biography)**: Stores actor details with full-text search on biographies.
- **movies (id, title, description, genre_id, director_id)**: Stores movies with references to genres and directors.
- **film_actor (id, movie_id, actor_id)**: Many-to-many relationship between movies and actors.

Database Design Rationale

- **Normalization (3NF):** Ensures minimal redundancy and data integrity.
- **Foreign Keys:** Maintain referential integrity.
- **Many-to-Many Relationships:** film_actor bridges movies and actors efficiently.
- **Alternative Design Considerations:**
    - **Denormalization:** Could improve read speed but increase data redundancy.
    - **Single Table for Movies & People:** Would simplify structure but reduce efficiency for queries.

2. Database Optimizations

- **Indexes:**
    - **Primary Keys & Foreign Keys:** Speed up joins and lookups.
    - **Full-Text Indexes:** Optimize search in movies.title, movies.description, and actors.biography.
    - **Indexes on genre_id and director_id:** Enhance filtering speed.
- **Partitioning:** Not used but could be implemented for very large datasets.
- **Query Optimization:** Uses efficient joins and subqueries to reduce unnecessary scans.

3. Main Queries & Database Support

1. **Movie Search:**
    - Uses MATCH(title, description) AGAINST() for full-text search.
    - Indexing supports fast retrieval.
2. **Actor Search:**
    - Uses full-text search in biography to find relevant actors.

3. **Most Popular Genre:**
    - Aggregates movies per genre_id, utilizing indexes for fast computation.
4. **Actor Collaborations:**
    - Self-join on film_actor enables finding actor pairs who co-starred in multiple movies.
5. **Directors with Diverse Genres:**
    - Uses COUNT(DISTINCT genre_id) HAVING >= 3 to identify versatile directors.

4. Code Structure & API Usage

- **create_db_script.py** - Initializes the database schema.
- **queries_db_script.py** - Contains predefined SQL queries.
- **api_data_retrieve.py** - Fetches and updates movie-related data.
- **populate_db.py** - Fetches external movie data and populates the database.
- **API Integration:**
    - Retrieves movie data, genres, actors, and directors.
    - Handles rate limits and errors.

End of Document