**Objective:**

The Python code implements two different querying models, **NxN** and **N+N**, for retrieving seller product catalogs and consolidating them for buyers. These models are designed to compare performance and execution time when multiple buyers query seller product data.

**Key Functions:**

**1. get\_db\_connection():**

This function establishes a connection to a MySQL database using the pymysql library. The connection details (e.g., host, user, password, and database name) are hardcoded. The cursor class used is DictCursor to fetch query results as dictionaries for easy access.

**2. fetch\_seller\_catalogs():**

This function retrieves product catalogs for each seller in the database:

* Queries the Sellers table to fetch all sellers.
* Iterates through each seller to fetch their product catalog by querying Products, Categories, and ProductInventory tables.
* Builds a dictionary mapping seller names to their product catalogs.
* Closes the database connection before returning the data.

**3. nxn\_query(buyers, sellers):**

Implements the **NxN model**, where each buyer queries every seller individually:

* Loops through all buyers and sellers.
* Appends the products from each seller’s catalog to a results list.
* Prints a message for each query operation.

**4. n\_plus\_n\_query(buyers):**

Implements the **N+N model**, where a consolidated product catalog is built first:

* Calls fetch\_seller\_catalogs() to retrieve all seller catalogs.
* Uses consolidate\_catalogs() to create a Global Catalog Repository (GCR).
* Each buyer queries this single consolidated catalog.
* Appends the GCR to the results list for all buyers.

**5. consolidate\_catalogs(sellers):**

* Combines all individual seller catalogs into a single list of products.

**6. Main Execution Flow:**

* Buyers are defined as a list of strings (e.g., "Buyer A", "Buyer B", etc.).
* Execution times for both models (NxN and N+N) are measured and compared.
* Results of both models are printed for inspection.

**Database Schema:**

The database includes the following tables:

1. **Categories**:
   * CategoryID (Primary Key)
   * CategoryName
   * CategoryDescription
   * CreatedAt
2. **CatalogSettings**:
   * SettingID (Primary Key)
   * SettingName
   * SettingValue
   * CreatedAt
3. **Sellers**:
   * SellerID (Primary Key)
   * SellerName
   * Email (Unique)
   * Password
   * CreatedAt
4. **Products**:
   * ProductID (Primary Key)
   * ProductName
   * Description
   * Price
   * CategoryID (Foreign Key referencing Categories)
   * CreatedAt
5. **ProductInventory**:
   * InventoryID (Primary Key)
   * ProductID (Foreign Key referencing Products)
   * SellerID (Foreign Key referencing Sellers)
   * Quantity
   * CreatedAt
6. **Orders**:
   * OrderID (Primary Key)
   * SellerID (Foreign Key referencing Sellers)
   * OrderDate
   * Total
7. **OrderItems**:
   * OrderItemID (Primary Key)
   * OrderID (Foreign Key referencing Orders)
   * ProductID (Foreign Key referencing Products)
   * Quantity

**Performance Comparison:**

**NxN Model:**

* Each buyer queries every seller individually.
* **Time Complexity:** O(N \* M), where N = number of buyers and M = number of sellers.

**N+N Model:**

* A consolidated catalog (GCR) is created once and queried by all buyers.
* **Time Complexity:** O(M + N), where M = number of sellers and N = number of buyers.

**Advantages and Disadvantages:**

**NxN Model:**

* **Advantages:**
  + Each buyer queries sellers directly, allowing for tailored and specific data retrieval.
  + No need for a central repository.
* **Disadvantages:**
  + High query overhead due to repeated operations.
  + Slower execution for larger datasets and more buyers.

**N+N Model:**

* **Advantages:**
  + Faster execution by consolidating catalogs into a GCR.
  + Reduced database queries.
* **Disadvantages:**
  + Additional memory required for the GCR.
  + Assumes that all buyers require access to the entire catalog.

**Output:**

1. **Execution Time:**
   * **NxN Model:** Prints the time taken for NxN queries.
   * **N+N Model:** Prints the time taken for N+N queries.

****

1. **Query Results:**
   * Lists the product details retrieved by both models.

A screen shot of a computer screen

Description automatically generated

**Conclusion:**

The following project was built using Python and MySQL and was run on a system running windows 11 Pro with a Core i9-12900H CPU (14 Cores, 20 Logical processors) with 16GB RAM.

This code provides a comparative analysis of two query models, demonstrating the trade-offs between individual and consolidated querying approaches. While the NxN model offers flexibility, the N+N model is more efficient for scenarios with multiple buyers querying shared catalogs.