Why did you pick your particular your design?

1. Scalability is one of the key aspect while choosing the design. Queries related to *point one* (show all orders from a particular company) and *point two* (show all orders to a particular address) are more similar in context. Therefore, a single function **getAllOrders** can process both queries. For the point three (delete a particular order given an order ID), if it requires to delete multiple orders either based on companyName, CustomerAddress, a single function can do the task.

2. Frequent database connection calls overload the server. Hence, a single instance of the database connection is created and used for processing the queries.

3. Query optimization techniques impact performance of the application. Therefore, limit is set to all orders queries of particular company and customer address to improve the performance.

What assumptions did you make, and what tradeoffs did you consider?

Assumptions

1. orderID in the input file is unique. Loading input file is only once when the server stars running and server is ready to process requests.

2. This application is assumed to be for demo purpose.

Tradeoffs

1. Heavy requests can overload the server performance. This can be optimized by caching get request- response, database connection.

2. Query optimization requires indexing. As fields are updated, associated indexes must be maintained, incurring additional CPU and disk I/O overhead.

What is the complexity of your operations?

1. As queries are performed on every field, all the fields are indexed considering query design techniques. For Search queries, without indexing the complexity is O (n) with *Collscan* and with indexing O (log (n)) with *IXscan*