1. Order Preserving Encryption:

An encryption technique known as order-preserving encryption (OPE) keeps the data values in their original order even after they have been encrypted. Put differently, the ciphertext values for two plaintext values should likewise remain in the same order if they are sorted in a certain manner. This feature comes in handy when sorting or database queries need the data to be in a specific order.

The operation of order-preserving encryption may be summarized as follows:

Mapping to Numeric Domain:

Mapping the values in plaintext to a numeric domain is the first step. Usually, a continuous numeric domain containing the whole range of potential plaintext values is used.

A numeric value from this domain is given to each plaintext value in such a way that the order of the numeric values matches the order of the plaintext values.

We have used order-preserving encryption (OPE) scheme for the Weight attribute and implementing support for range queries involves careful consideration of security, efficiency, and the specific requirements of your application. One commonly used OPE scheme is the Deterministic Order-Preserving Encryption (DOPE) scheme. Here's an example implementation.

1. Implementation for Weight Attribute:

1. Key Generation: To begin, a safe encryption key must be created. The Weight values will be transformed using this key in a way that maintains their order.

2. Encryption Function: We employ the Electronic Codebook (ECB) mode of the AES encryption method for every Weight value. To provide a consistent format for encryption, the Weight is transformed into a fixed-size byte representation. The outcome is a weight that has been encrypted while keeping the original values' order.

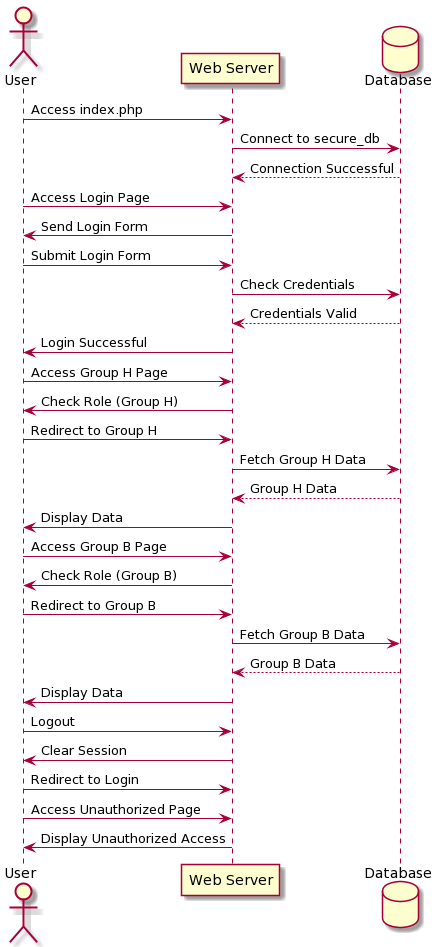
3. Decryption Function: We build a decryption function to extract the original Weight values from the encrypted versions. This function takes the encrypted Weight and turns the bytes back into an integer by reversing the encryption. The decryption process maintains the original sequence.

4. Range Query Function: We may obtain data items with Weight values that fall inside a given range by using the range query function. We encrypt the Weight of each item in the collection and verify that the decrypted value is within the designated range. If so, the item appears in the results.

5. Use Case: We offer a basic case study illustrating the utilization of the range query function. For the intended range, we provide a lower and upper bound given a dataset containing Weight values. Following that, the function provides a list of dataset objects whose Weight values are within that range.

Secure range queries on the encrypted data are made possible by this approach, which makes sure that the order of Weight values is maintained both during encryption and decryption. It should be noted that AES was selected as the encryption technique due to its simplicity; nevertheless, in a real-world setting, other factors and optimizations can be required depending on the requirements and security concerns of the application.

1. System Architecture:



1. Implementation:

**Step 1:** Clone this repository to local machine.

**Step 2:** Set up a SQL database system, such as MySQL, on local environment.

**Step 3:** Here we taken two user groups (Group H and Group B) with different access permissions.

**Step 4:** Created the database tables as specified in the project requirements.

**Step 5:** we implement the security features and access control mechanisms such as

- User authentication and password hashing.

- Access control for different user groups (Group H and Group B).

- Query integrity protection and data item integrity checks.

- Data confidentiality protection for sensitive attributes.

**Step 6:** Accessed the main page (`index.php`) to interact with the secure database.

**Step7:** Registered and login as a user to access respective user group's features.

**Step 8:** Explored Group H and Group B pages to see the access control mechanisms in action.