Query, Key, and Value Concepts

🔍

Query (Q)

What information am I looking for?

🔑

Key (K)

What information do I contain?

💎

Value (V)

The actual information I store

💡 Database Analogy

Query matches **Keys** to retrieve **Values**

Linear Projections

1. Each token generates Q, K, V
2. Through learned transformations:

$$Q = XW_Q$$

$$K = XW_K$$

$$V = XW_V$$

3. Projection matrices **learned during training**

> **X** = Input token embeddings
> **W_Q, W_K, W_V** = Trainable weight matrices

Numerical Example: Computing Q, K, V

📥 Input Token Embeddings (X)

5 tokens × 3 dimensions

```
X = [
  [1.0, 0.5, 0.2],  # Token 1
  [0.3, 1.2, 0.8],  # Token 2
  [0.7, 0.4, 1.1],  # Token 3
  [1.5, 0.9, 0.3],  # Token 4
  [0.6, 1.3, 0.7]   # Token 5
]
```

W_Q (Query)

```
[[ 1.0, 0.0, 0.5],
 [ 0.5, 1.0, 0.0],
 [ 0.0, 0.5, 1.0]]
```

W_K (Key)

```
[[ 0.8, 0.2, 0.3],
 [ 0.3, 0.9, 0.1],
 [ 0.2, 0.1, 0.8]]
```

W_V (Value)

```
[[ 1.0, 0.3, 0.2],
 [ 0.2, 1.0, 0.4],
 [ 0.1, 0.3, 1.0]]
```

⬇️ Matrix Multiplication

$Q = XW_Q$

```
[[1.10, 0.60, 0.70],
 [1.14, 0.95, 0.95],
 [1.25, 0.90, 1.45],
 [1.95, 0.90, 1.05],
 [1.31, 1.30, 1.00]]
```

$K = XW_K$

```
[[1.01, 0.47, 0.46],
 [0.64, 1.29, 0.26],
 [0.98, 0.63, 1.09],
 [1.53, 1.14, 0.69],
 [1.01, 1.29, 0.74]]
```

$V = XW_V$

```
[[1.12, 0.56, 0.54],
 [0.62, 1.41, 1.12],
 [0.89, 0.74, 1.40],
 [1.71, 1.02, 0.90],
 [0.89, 1.49, 1.22]]
```

💡 Key Insight

Each of the 5 tokens now has its own:

- **Query vector** (what it's looking for)
- **Key vector** (what it contains)
- **Value vector** (its actual information)

These will be used in the attention mechanism to determine which tokens should attend to which!