```
class HookedClassCreationMeta(type):
    def __new__(cls, name, bases, dct):
        # Intercept attribute addition
        for attribute_name, attribute_value in dct.items():
            if not attribute_name.startswith("__"):
                print(f"Intercepted attribute addition: {attribute_name} = {attribute_value}")

        # Intercept method creation
        for key, value in dct.items():
            if callable(value) and not key.startswith("__"):
                print(f"Intercepted method creation: {key}")

        # Create the class using the default type.__new__
        return super().__new__(cls, name, bases, dct)

# Example usage of the metaclass
class MyClass(metaclass=HookedClassCreationMeta):
    # Attributes
    attribute1 = "Value 1"
    attribute2 = 42

    # Methods
    def method1(self):
        print("Method 1")

    def method2(self):
        print("Method 2")

# When the class is created, the metaclass intercepts attribute addition
# and method creation and prints the relevant information.
```

```
Log: User created
Saving to database: {'user_id': 1, 'name': 'John'}
```