

fx-test_flush_no_pk

Link

<https://github.com/sqlalchemy/sqlalchemy/blob/8503dc2e948908199cd8ba4e6b1d1ddcf92f402>

▼ Code

```
@Profiler.profile
def test_flush_no_pk(n):

    session = Session(bind=engine)
    for chunk in range(0, n, 1000):
        session.add_all(
            [
                Customer(
                    name="customer name %d" % i,
                    description="customer description %d" % i,
                )
                for i in range(chunk, chunk + 1000)
            ]
        )
        session.flush()
    session.commit()
```

▼ Documentation

```
"""INSERT statements via the ORM (batched with RETURNING if available),
    fetching generated row id"""
```

Prompt

As a code documentation assistant, you are programmed to document at the function level of the given code snippet. Your approach involves placing comments directly under the def statement of the function. The output should be the entire code along with the documentation of the function written.

Code is as follows:

```
"""
@Profiler.profile
def test_flush_no_pk(n):

    session = Session(bind=engine)
    for chunk in range(0, n, 1000):
```

```

    session.add_all(
        [
            Customer(
                name="customer name %d" % i,
                description="customer description %d" % i,
            )
            for i in range(chunk, chunk + 1000)
        ]
    )
    session.flush()
    session.commit()
"""

```

Responses

ChatGPT

```

@Profiler.profile
def test_flush_no_pk(n):
    """
    This function populates the database with customer records in chunks and flushes the session

    Args:
        n (int): The total number of customer records to be created.

    Returns:
        None

    Raises:
        None

    This function is used to efficiently insert a large number of customer records into the data
    which represents the total number of customer records to be created. It creates customer rec
    the session after each chunk to minimize memory usage and improve performance.
    """

    # Create a new session with the database engine.
    session = Session(bind=engine)

    # Iterate over the range of 'n' in chunks of 1000.
    for chunk in range(0, n, 1000):
        # Create a list of customer records within the current chunk.
        customers = [
            Customer(
                name="customer name %d" % i,
                description="customer description %d" % i,
            )
            for i in range(chunk, chunk + 1000)
        ]

```

```

    # Add all customer records to the session.
    session.add_all(customers)

    # Flush the session to persist changes to the database.
    session.flush()

# Commit the changes to the database.
session.commit()

```

GPT4

```

from sqlalchemy.orm import Session

@Profiler.profile
def test_flush_no_pk(n):
    """
    This function tests the flushing of a session with a specified number of Customer entries wi

    Parameters:
    n (int): The total number of Customer entries to be added and flushed in the session.

    The function creates a session bound to an engine. It then iteratively adds 'Customer' objec
    Each 'Customer' object is created with a name and description formatted with an index.
    After each chunk of 1000 Customers, the session is flushed to execute the pending changes,
    but without committing them, allowing further additions. Finally, it commits the session to
    """

    session = Session(bind=engine)
    for chunk in range(0, n, 1000):
        session.add_all(
            [
                Customer(
                    name="customer name %d" % i,
                    description="customer description %d" % i,
                )
                for i in range(chunk, chunk + 1000)
            ]
        )
        session.flush()
    session.commit()

```

Bard

```

@Profiler.profile
def test_flush_no_pk(n):
    """
    This function tests the behavior of flushing a Session object without primary keys.

    Args:
        n (int): The number of customers to create.
    """

```

```

Returns:
    None
"""

session = Session(bind=engine)
for chunk in range(0, n, 1000):
    session.add_all(
        [
            Customer(
                name="custoamer name %d" % i,
                description="customer description %d" % i,
            )
            for i in range(chunk, chunk + 1000)
        ]
    )
    session.flush()
session.commit()

```

LLama2

```

@Profiler.profile
def test_flush_no_pk(n):
    """
    Tests the flush method without primary key.

    :param n: Number of customers to create
    :type n: int
    """
    session = Session(bind=engine)
    for chunk in range(0, n, 1000):
        session.add_all(
            [
                Customer(
                    name="customer name {}".format(i),
                    description="customer description {}".format(i),
                )
                for i in range(chunk, chunk + 1000)
            ]
        )
        session.flush()
    session.commit()

```

StarChat

```

@Profiler.profile
def test_flush_no_pk(n):
    """This function adds `n` customers without primary keys into the database using flush().

    Args:
        n (int): Number of customer records to add.
    """
    # Create a new SQLAlchemy engine
    engine = create_engine("sqlite:///example.db")

```

```
# Start a new session
session = Session(bind=engine)

# Loop through chunks of data
for chunk in range(0, n, 1000):
    # Add multiple rows to the session
    session.add_all([Customer(name=f'custoamer name {i}',
                               description=f'customer description {i}')
                     for i in range(chunk, chunk+1000)])

    # Flush the changes made during this iteration to the database
    session.flush()

# Commit any unflushed changes and close the session
session.commit()
session.close()
```