```
# Appendix C18 - test_table_definition.py

from klassify.src.db_handler import DBHandler
from klassify.src.tables import Topic, Subtopic, Document
import pytest
import sqlalchemy

def test_db():
    database_name = "test_klassify"

    DBH = DBHandler(database_name, echo=False)
    session = DBH.session
    # create a topic, subtopic and document
    test_topic = Topic(title="HMRC", base_path="/hmrc")
    test_subtopic_1 = Subtopic(title="HMRC payments", base_path="/payments")
    test_subtopic_2 = Subtopic(title="HMRC refunds", base_path="/refunds")
    test_document_1 = Document(
        title="Self assessment deadlines",
        base_path="/self-assessment",
        html="<strong>PAY NOW</strong>")
    test_document_2 = Document(
        title="Starting a business",
        base_path="/start-business",
        html="<strong>START NOW</strong>")
    test_document_3 = Document(
        title="Payment and refunds",
        base_path="/payments-and-refunds",
        html="<h1>payments and refunds</h1>")

    # create relationships
    test_topic.subtopics    = [test_subtopic_1, test_subtopic_2]
    test_subtopic_1.documents = [test_document_1, test_document_2]
    test_document_3.subtopics = [test_subtopic_1, test_subtopic_2]

    # add topic to session
    session.add_all([
        test_topic,
        test_subtopic_1,
        test_subtopic_2,
        test_document_1,
        test_document_2,
        test_document_3
    ])

    session.commit()

    # Table properties
    assert session.query(Topic).get(test_topic.id).title == test_topic.title
    assert session.query(Topic).get(test_topic.id).base_path == test_topic.base_path
    assert session.query(Subtopic).get(test_subtopic_1.id).title == test_subtopic_1.title
    assert session.query(Subtopic).get(test_subtopic_1.id).base_path == test_subtopic_1.b
ase_path
    assert session.query(Document).get(test_document_1.id).title == test_document_1.title
    assert session.query(Document).get(test_document_1.id).base_path == test_document_1.b
ase_path

    # test relationships
    topics_and_subtopics = session.query(Topic).get(test_topic.id).subtopics
    subtopics_titles = [subtopic.title for subtopic in topics_and_subtopics]
    assert test_subtopic_1.title in subtopics_titles
    assert test_subtopic_2.title in subtopics_titles

    subtopics_and_documents = session.query(Subtopic).get(test_subtopic_1.id).documents
    documents_titles = [document.title for document in subtopics_and_documents]
    assert test_document_1.title in documents_titles
    assert test_document_2.title in documents_titles

```

```
    documents_and_subtopics = session.query(Document).get(test_document_3.id).subtopics
    subtopics_titles = [subtopic.title for subtopic in documents_and_subtopics]
    assert test_subtopic_1.title in subtopics_titles
    assert test_subtopic_2.title in subtopics_titles

    # Test Document->Topics relation
    doc = session.query(Document).get(test_document_1.id)
    topic = session.query(Topic).get(test_topic.id)
    assert topic in doc.topics()
    assert topic.title in doc.topic_titles()

    # Test Topic->Documents relation
    doc = session.query(Document).get(test_document_1.id)
    topic = session.query(Topic).get(test_topic.id)
    assert doc in topic.documents()
    assert doc, topic.title in topic.documents_with_labels()

    # test unique constraint on basepath
    clone_topic = Topic(title="Clone topic", base_path="/hmrc")
    clone_subtopic = Subtopic(title="Clone subtopic", base_path="/refunds")
    clone_document = Document(
        title="Clone document",
        base_path="/payments-and-refunds",
        html="<h1>payments and refunds</h1>")
    clones = [clone_topic, clone_subtopic, clone_document]
    for clone in clones:
        with pytest.raises(sqlalchemy.exc.IntegrityError):
            session.rollback()
            session.add_all([clone])
            session.commit()

    # terminate session and delete test db
    session.close()
    DBH.destroy_db_if_present()
```