

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green color. They are positioned diagonally, with the blue one in front of the green one.

CRUD Operation and SQLAlchemy



Understanding CRUD Operation and SQLAlchemy

CRUD stand for :(create,read, update ,Delete)

Create : Creating new records in the database.

Read : Retrieving existing records in the database

Update : Modifying and updating existing records in the database

Delete : Removing records from the database

IMPORTANCE OF CRUD

1. CRUD operations are the fundamental building blocks of the database interaction
2. They enable applications to manage data effectively
3. Together they form the backbone of user interactions and data management in applications

##NB: SQLAlchemy allows developers to interact with database using python objects



The Role of SQLAlchemy in Database Operations

Importance of ORM:

1. An Object-Relational Mapping(ORM) just like SQLAlchemy simplifies the interaction between code and databases
2. It allows developers to work with python objects instead of writing complex SQL queries
3. The abstraction reduces errors , accelerates development and enhances maintainability

Installing SQLAlchemy

To use SQLAlchemy, one needs to install it.

We use the 'pip' package manager for installation simply by running this command

```
pip install sqlalchemy
```



Understanding object-Relational mapping and defining models

ORM CONCEPT

- ORM is a technique that connects the world of programming objects with relational databases
- It maps database tables to python classes and rows to objects
- ORM bridges the gap between the object-oriented and relational paradigms, simplifying database interaction

Defining Models with SQLAlchemy

- SQLAlchemy provides a declarative syntax for defining data models
- Models are Python classes that represent database tables
- columns within the table are defined as class attributes



Creating Data using SQLAlchemy

Create Operation:

- The create operation is a fundamental part of CRUD
- It involve adding new records or entries to the database
- this operation is essential for initializing data or capturing user input

Using SQLAlchemy to create

- SQLAlchemy simplifies the creation process using python objects

Example

- 1.Create an instance of the model class representing the table

```
new_user = User(username = 'john_doe', email='john@example.com')
```

- 2.add an instance to a session to prepare for addition

```
session.add(new_user)
```

```
session.commit()
```



Reading Data with SQLAlchemy

Read Operation:

- It involves retrieving data from the database
- It is essential for displaying information or performing analysis

Using SQLAlchemy Querying API:

- SQLAlchemy provides a rich Querying API to retrieve data
- Begin by creating a query object on the model class
- Perform operations on the query object to filter and retrieve data

Example :Query All users

```
users = session.query(User).all()
```

Example : Query User by ID

```
user = session.query(User).get(user_id)
```



Updating Data using SQLAlchemy

Update Operation:

- The Update operation involves modifying existing records in the database.
- It's crucial for keeping data accurate and up-to-date as application requirements change.

Updating Records with SQLAlchemy:

- SQLAlchemy streamlines the update process using Python objects.
- Retrieve the desired record using queries.
- Make changes to the object attributes.
- Commit the changes to the database session.

Example : updating a user's Email

```
user = session.query(User).get(user_id)
```

```
user.email = 'new_email@example.com'
```

```
session.commit()
```



Deleting Data with SQLAlchemy

Delete Operation:

- The Delete operation involves removing records from the database.
- It's essential for data management and maintaining a clean database.

Deleting Records with SQLAlchemy:


- SQLAlchemy simplifies the delete process using Python objects.
- Query the session to fetch the record to be deleted.
- Use the delete() method on the session to remove the object.
- Commit the changes to finalize the deletion.

Example: Deleting a User:

```
user = session.query(User).get(user_id)

session.delete(user)

session.commit()
```

Ensuring Data Integrity and Effective session management

Importance of Exception Handling:

- Exception handling is vital for maintaining data integrity.
- It prevents unexpected errors from leaving the database in an inconsistent state.
- Properly handled exceptions ensure that data remains accurate and usable.

Effective Session Management:

- Improper session management can lead to resource leaks and performance issues.
- Sessions represent a connection to the database and should be properly managed and closed.

Using Context Managers for Session Handling:

- Context managers (e.g., with statements) ensure proper session closure.
- They automatically handle opening and closing sessions, preventing resource leaks.
- This approach minimizes the risk of leaving open connections to the database.



CONCLUSION

- Key Points Summary:
 - CRUD operations (Create, Read, Update, Delete) are fundamental for managing data.
 - SQLAlchemy simplifies database interactions using Python objects.
 - ORM bridges the gap between objects and relational databases.

HAPPY CODING !!!!!