

[What’s New in SQLAlchemy 2.0? — SQLAlchemy 2.0 Documentation](https://docs.sqlalchemy.org/en/20/changelog/whatsnew_20.html)

the two most significant front-facing portions of SQLAlchemy are the **Object Relational Mapper (ORM)** and the **Core**.

Core contains the breadth of SQLAlchemy’s SQL and database integration and description services, the most prominent part of this being the **SQL Expression Language**.

The SQL Expression Language is a toolkit on its own, independent of the ORM package, which provides a system of constructing SQL expressions represented by composable objects, which can then be “executed” against a target database within the scope of a specific transaction, returning a result set. Inserts, updates and deletes (i.e. [DML](https://docs.sqlalchemy.org/en/20/glossary.html#term-DML)) are achieved by passing SQL expression objects representing these statements along with dictionaries that represent parameters to be used with each statement.

The ORM builds upon Core to provide a means of working with a domain object model mapped to a database schema. When using the ORM, SQL statements are constructed in mostly the same way as when using Core, however the task of DML, which here refers to the persistence of business objects in a database, is automated using a pattern called [unit of work](https://docs.sqlalchemy.org/en/20/glossary.html#term-unit-of-work), which translates changes in state against mutable objects into INSERT, UPDATE and DELETE constructs which are then invoked in terms of those objects. SELECT statements are also augmented by ORM-specific automations and object-centric querying capabilities.

**Basics of SQLAlchemy**

1. **What is SQLAlchemy, and why is it used?**
   * Discuss its role as an Object Relational Mapper (ORM) and its advantages.
2. **What are the key components of SQLAlchemy?**
   * Explain the role of components like Engine, Session, Query, Base, and Model.
3. **How do you define a table in SQLAlchemy?**
   * Example of using declarative\_base() to define a model.
4. **What is the difference between Core and ORM in SQLAlchemy?**
   * Compare SQLAlchemy Core (SQL Expression Language) with the ORM layer.

**Database Connections**

1. **How do you connect to a database using SQLAlchemy?**
   * Explain create\_engine and provide an example.
2. **What is the purpose of the echo parameter in create\_engine?**
   * Discuss how it logs SQL statements for debugging.
3. **How do you configure connection pooling in SQLAlchemy?**
   * Describe pool\_size, max\_overflow, and pool\_recycle.

**Model Definition and Relationships**

1. **How do you define a primary key and foreign key in a model?**
   * Example of using primary\_key=True and ForeignKey.
2. **What are the different types of relationships in SQLAlchemy?**
   * Explain one-to-one, one-to-many, and many-to-many with examples using relationship().
3. **What is the difference between relationship() and backref()?**
   * Explain how they work together to simplify bidirectional relationships.

**CRUD Operations**

1. **How do you perform basic CRUD operations in SQLAlchemy?**
   * Provide examples for Create, Read, Update, and Delete.
2. **What is the difference between add() and merge() in a SQLAlchemy session?**
   * Explain when to use each method.
3. **How do you update records in SQLAlchemy?**
   * Discuss both ORM-based and Core-based approaches.

**Sessions and Transactions**

1. **What is the purpose of a Session in SQLAlchemy?**
   * Discuss session lifecycle and its role in ORM.
2. **What is the difference between commit() and flush() in SQLAlchemy?**
   * Explain their roles in persisting changes to the database.
3. **How do you handle transactions in SQLAlchemy?**
   * Example using session.begin() and rollback scenarios.

**Querying**

1. **How do you perform a filter query in SQLAlchemy?**
   * Example using filter(), filter\_by(), and comparison operators.
2. **What is the difference between Query.all() and Query.first()?**
   * Discuss their use cases.
3. **How do you use joins in SQLAlchemy?**
   * Provide examples of inner and outer joins.
4. **What is eager loading, and how do you implement it in SQLAlchemy?**
   * Explain joinedload and subqueryload.

**Advanced Topics**

1. **What are hybrid properties in SQLAlchemy?**
   * Discuss how they combine SQL expressions and Python logic.
2. **How does SQLAlchemy handle migrations?**
   * Discuss integration with Alembic for schema migrations.
3. **What are the differences between lazy, joined, and subquery loading in SQLAlchemy?**
   * Explain with examples and scenarios for use.
4. **How do you define a composite primary key in SQLAlchemy?**
   * Example using PrimaryKeyConstraint.
5. **What is the purpose of declarative\_base() in SQLAlchemy?**
   * Discuss how it serves as the base class for ORM models.

**Performance and Optimization**

1. **How do you optimize queries in SQLAlchemy?**
   * Techniques like indexing, batch processing, and query optimization.
2. **What is the difference between query.count() and len(query.all())?**
   * Discuss performance implications.
3. **How do you use bulk inserts in SQLAlchemy?**
   * Explain bulk\_save\_objects() and bulk\_insert\_mappings().

**Error Handling**

1. **How do you handle exceptions in SQLAlchemy?**
   * Example using try-except blocks for handling IntegrityError and other exceptions.
2. **What is the difference between session.rollback() and session.close()?**
   * Discuss their roles in error handling.

**Scenario-Based Questions**

1. **How would you implement a soft delete in SQLAlchemy?**
   * Example of adding a deleted flag and filtering it out in queries.
2. **How do you implement a polymorphic relationship in SQLAlchemy?**
   * Explain inheritance and \_\_mapper\_args\_\_.
3. **How would you model hierarchical data (e.g., categories and subcategories)?**
   * Discuss adjacency list or nested set approaches.