1. JDBC (Java Database Connectivity) - это API для взаимодействия Java-приложений с реляционными базами данных. Он позволяет Java-приложениям выполнять SQL-запросы, обновлять данные в базе данных, извлекать данные из базы данных и управлять транзакциями.
2. JDBC драйвер - это компонент, который обеспечивает соединение между Java-приложением и базой данных. Бывают 4 типа JDBC драйверов:

* JDBC-ODBC мостовой драйвер
* JDBC-сетевой драйвер (pure Java)
* JDBC-серверный драйвер
* JDBC-встроенный драйвер

1. Чтобы извлечь данные из базы данных, необходимо выполнить следующие шаги:

* Загрузить JDBC драйвер
* Создать соединение с базой данных
* Создать объект Statement или PreparedStatement для выполнения запроса к базе данных
* Выполнить запрос, используя методы execute, executeQuery или executeUpdate объекта Statement или PreparedStatement
* Извлечь результаты запроса из объекта ResultSet

1. Параметры соединения задаются в строке подключения (connection string), которая содержит информацию о базе данных, сервере базы данных, порте, пользователе и пароле. Строка подключения передается в метод DriverManager.getConnection().
2. Основными компонентами JDBC API являются:

* DriverManager: класс для управления соединением с базой данных
* Connection: интерфейс для соединения с базой данных
* Statement: интерфейс для выполнения SQL-запросов к базе данных
* PreparedStatement: интерфейс для выполнения подготовленных SQL-запросов к базе данных
* ResultSet: интерфейс для хранения результатов выполнения SQL-запросов

1. Метод execute используется для выполнения любых SQL-запросов, метод executeQuery используется для выполнения SELECT-запросов, которые возвращают набор результатов (ResultSet), а метод executeUpdate используется для выполнения запросов, которые изменяют данные в базе данных (INSERT, UPDATE, DELETE).
2. Интерфейс ResultSet в Java представляет результаты запроса, выполняемого к базе данных. Он позволяет перебирать и извлекать данные из таблицы, полученные в результате выполнения запроса. ResultSet имеет курсор, который указывает на текущую позицию в таблице. Метод next() перемещает курсор на следующую строку результата и возвращает true, если строка существует. Методы ResultSet позволяют получить данные из текущей строки результата, а также метаданные о столбцах таблицы.
3. PreparedStatement используется для выполнения SQL-запросов, которые содержат параметры, значения которых определяются во время выполнения программы. Он используется для предотвращения SQL-инъекций и повышения производительности. Запросы в PreparedStatement компилируются в SQL-запросы перед выполнением, что позволяет повторно использовать запросы и избежать повторного парсинга SQL-запроса.
4. Транзакции используются в базах данных для группировки одного или нескольких запросов в одну логическую единицу работы. Транзакции обеспечивают атомарность, согласованность, изолированность и долговечность (ACID) базы данных. Атомарность означает, что транзакция либо выполняется полностью, либо не выполняется вообще. Согласованность означает, что выполнение транзакции приводит базу данных к правильному состоянию. Изолированность означает, что транзакции не мешают друг другу и не могут читать или записывать данные друг друга. Долговечность означает, что результаты выполненной транзакции сохраняются в базе данных и не будут потеряны из-за сбоев в системе.
5. Типы чтения данных в транзакциях включают:

* Чтение не фиксируемых данных (dirty read) - это чтение данных, которые еще не были подтверждены транзакцией и могут быть изменены или удалены позже. Это может привести к несогласованности данных.
* Не повторяемое чтение данных (non-repeatable read) - это чтение данных, которые были изменены другой транзакцией между двумя чтениями той же транзакцией. Это может привести к несогласованности данных.
* Фантомное чтение данных (phantom read) - это чтение данных, которые были добавлены или удалены другой транзакцией между двумя чтениями той же транзакцией. Это может привести к несогласованности данных.

Уровни изоляции определяют, какие типы чтения данных в транзакциях разрешены. Уровень изоляции может быть установлен для каждой транзакции, и он может быть разным для разных транзакций в системе. Некоторые из уровней изоляции включают уровень чтения не фиксируемых данных (read uncommitted), уровень чтения фиксируемых данных (read committed), уровень повторяемого чтения (repeatable read) и уровень сериализации (serializable).

1. DatabaseMetaData - это интерфейс, который предоставляет информацию о базе данных, такую как названия таблиц, типы данных, ограничения, индексы и т.д. С помощью этого интерфейса можно получить информацию о структуре базы данных и использовать ее для создания динамических SQL-запросов.
2. DAO – необходим для работы с базами данных