1. **Назначение и возможности JDBC?**

Для стандартизации работы с SQL-серверами взаимодействие с ней можно выполнять через единую точку — JDBC (Java DataBase Connectivity). Она представляет собой реализацию пакета java.sql для работы с СУБД. Производители всех популярных SQL-серверов выпускают для них драйверы JDBC.

1. **Что такое JDBC драйвер? Какие бывают типы?**

**JDBC Driver** – (*Java DataBase Connectivity — соединение с базами данных на Java*) — платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета java.sql, входящего в состав Java SE.

**Зачем нужен JDBC?**

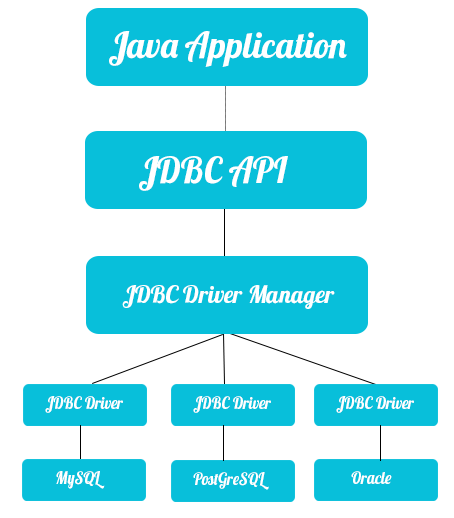
**JDBC**– позволяет получать доступ к БД, а также выполнять к ней SQL запросы.

Тип 1: JDBC-ODBC бридж (JDBC-ODBC Bridge plus ODBC Driver) — транслирует JDBC в ODBC и для взаимодействия с базой данных использует драйвер ODBC. Компания Sun включила в состав JDK один такой драйвер — мост JDBC/ODBC. Сейчас имеются более удачные реализации.

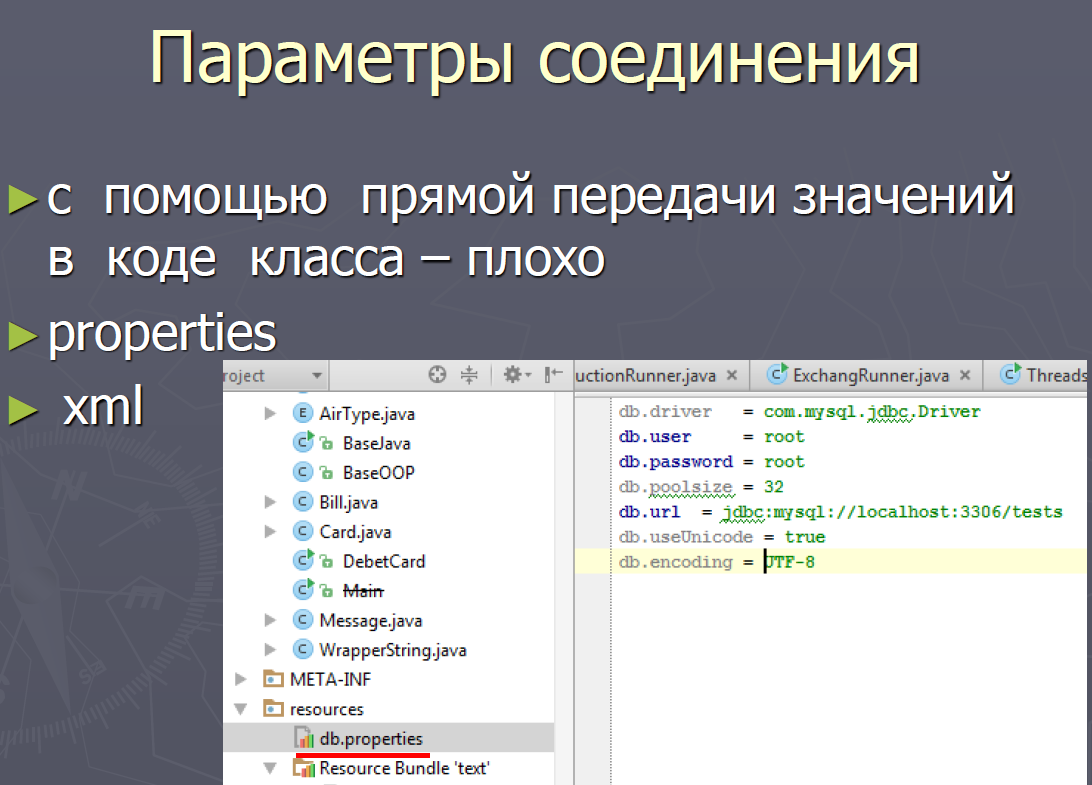
Тип 2: Нативный API/частично Java драйвер (Native API partly Java technology-enabled driver) — переводит вызовы JDBC в вызовы специфичные к СУБД таких как например SQL Server, Informix, Oracle или Sybase. Драйвер 2-го типа общается напрямую с сервером базы данных, следовательно он требует, чтобы какой то бинарный код был на стороне клиентской машины.

Тип 3: Сетевой протокол/«чистый» Java драйвер (Pure Java Driver for Database Middleware) — использует трехуровневую архитектуру, где вызовы JDBC посылаются на промежуточный т.н. сервер приложений, далее этот сервер транслирует вызовы (явно или косвенно) в вызовы специфичного к СУБД нативного интерфейса для дальнейшего обращения к базе данных. Если сервер среднего слоя написан на Java то он может использовать для трансляции JDBC драйверы 1 и 2 типов.

Тип 4: Нативный протокол/«чистый» Java драйвер (Direct-to-Database Pure Java Driver) — конвертирует вызовы JDBC в специфический протокол вендора СУБД, так что клиентские приложения могут напрямую обращаться с сервером базы данных. Драйверы 4-го типа полностью реализуются на Java с целью достижения платформенной независимости и устранения проблем администрирования и развертывания.



1. Какие шаги (перечислите) нужно выполнить для извлечения данных?
2. **Как и где задать параметры соединения?**



1. **Перечислите основные JDBC API компоненты.**

Основные компоненты **JDBC Api** включают:

**DriverManager**:

Это класс, использующийся для управления списком **Driver**(database drivers).

***Driver***:

Это интерфейс, использующийся для соединения коммуникации с базой данных, управления коммуникации с базой данных. Когда загружается Driver, программисту не нужно конкретно вызывать его.

***Connection***:

Интерфейс со всеми методами связи с базой данных. Он описывает коммуникационный контекст. Вся связь с базой данных осуществляется только через объект соединения (connection).

***Statement***:

Это интерфейс, включающий команду SQL отправленный в базу данных для анализа, обобщения, планирования и выполнения.

***ResultSet*** представляет набор записей, извлеченных из-за выполнения запроса.

1. **Какая разница между еxecute, executeQuery, executeUpdate?**

Существует несколько способов выполнять SQL-запросы в зависимости от типа этого запроса. Для этого у интерфейса Statement существует три различных метода: executeQuery(), executeUpdate(), а так же execute().

Рассмотрим их отдельно.

Самый базовый метод executeQuery() необходим для запросов, результатом которых является один единственный набор значений, таких как у запросов SELECT. Возвращает ResultSet, который не может быть null даже если у результата запроса не было найдено значений.

Метод execute() используется, когда операторы SQL возвращают более одного набора данных, более одного счетчика обновлений или и то, и другое. Метод возвращает true, если результатом является ResultSet, как у запроса SELECT. Вернет false, если ResultSet отсутствует, например при запросах вида Insert, Update. С помощью методов getResultSet() мы можем получить ResultSet, а getUpdateCount() — количество обновленных записей.

Метод executeUpdate() используется для выполнения операторов INSERT, UPDATE или DELETE, а также для операторов DDL (Data Definition Language — язык определения данных), например, CREATE TABLE и DROP TABLE. Результатом оператора INSERT, UPDATE, или DELETE является модификация одной или более колонок в нуле или более строках таблицы. Метод executeUpdate() возвращает целое число, показывающее, сколько строк было модифицировано. Для выражений типа CREATE TABLE и DROP TABLE, которые не оперируют над строками, возвращаемое методом executeUpdate() значение всегда равно нулю.

Все методы выполнения SQL-запросов закрывают предыдущий набор результатов (result set) у данного объекта Statement. Это означает, что перед тем как выполнять следующий запрос над тем же объектом Statement, надо завершить обработку результатов предыдущего (ResultSet).

1. **Расскажите об интерфейсе ResultSet.**

JDBC ResultSet — интерфейс, объект которого создается в результате запроса к базе данных. Его можно представить в виде таблицы данных, которая была сформирована в ответ на запрос.

Объект ResultSet поддерживает курсор, который указывает на текущую строку данных. При инициализации курсор устанавливается до первой строки. Для движение по строкам используется метод next(). При наличии строк после текущей позиции, метод next() возвращает true, что можно использовать для итерации по таблице полученных результатов.

**8. Как и для чего используют PreparedStatement?**

Объект PreparedStatement используется для выполнения прекомпилированных SQL-запросов с или без входных (IN) параметров. Мы можем использовать сеттеры для установки значений в запрос. Т.к. PreparedStatement является предкомпилированным, то он может быть эффективно использован множество раз. PreparedStatement считается лучшим выбором нежели Statement, т.к. он автоматически обрабатывает специальные символы, а так же предотвращает, так называемые, SQL injection attack (когда в запрос можно подставить свой код).

1. **Как и для чего используют транзакции?**

Транзакция — это осуществление одного или нескольких изменений базы данных. Например, если вы создаете, обновляете или удаляете запись из таблицы, вы выполняете в этой таблице транзакцию. Важно контролировать транзакции, чтобы обеспечить целостность данных и обрабатывать ошибки базы данных.

Практически вы собираете множество SQL-запросов в группу, и они будут выполняться вместе как часть транзакции.

Свойства транзакций

Транзакции имеют следующие четыре стандартных свойства, обычно обозначаемых аббревиатурой ACID.

Атомарность – обеспечивает, чтобы все операции входящие в единицу работы были завершены успешно. В противном случае транзакция прерывается в момент сбоя, и все предыдущие операции возвращаются в прежнее состояние.

Согласованность — обеспечивает, чтобы база данных надлежащим образом изменяла состояние при успешной транзакции.

Изолированность — позволяет транзакциям работать независимо друг от друга и прозрачно.

Долговечность — гарантирует, что результат совершенной транзакции сохранится в случае сбоя системы.

Управление транзакциями

Для управления транзакциями используются следующие команды.

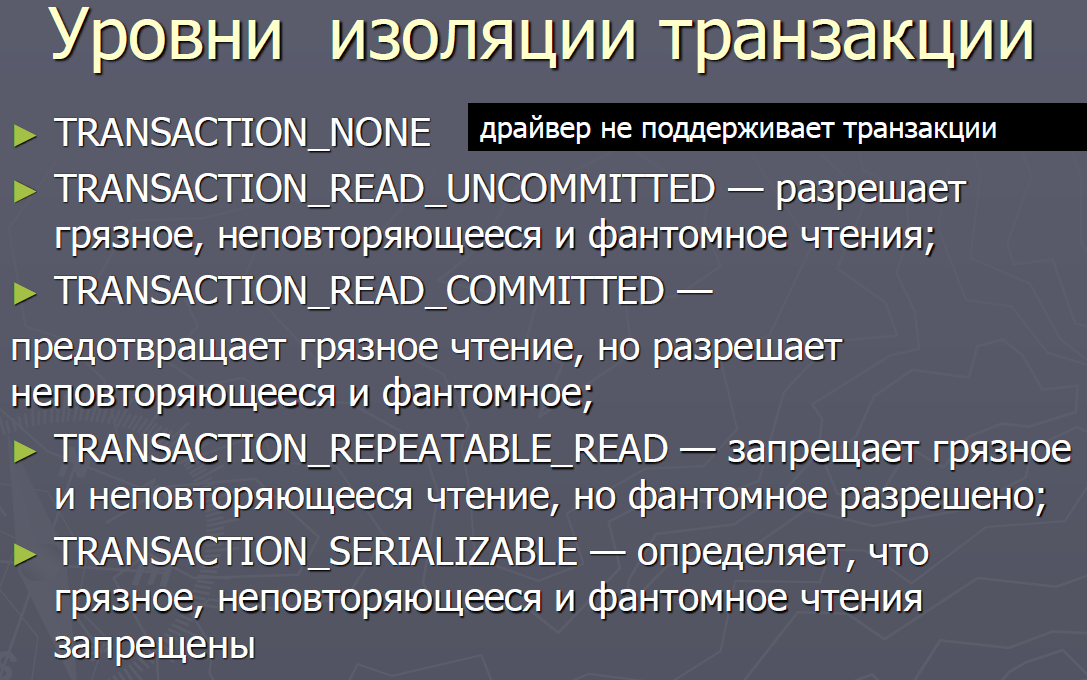
COMMIT — сохранить изменения.

ROLLBACK — отменить изменения.

SAVEPOINT — создает точки сохранения в группах транзакций.

SET TRANSACTION — помещает имя в транзакцию.

1. **Перечислите и поясните типы чтения данных в транзакциях. Что такое уровни изоляции?**

****

Уровень изолированности транзакций — условное значение, определяющее, в какой мере в результате выполнения логически параллельных транзакций в СУБД допускается получение несогласованных данных. Шкала уровней изолированности транзакций содержит ряд значений, проранжированных от наинизшего до наивысшего; более высокий уровень изолированности соответствует лучшей согласованности данных, но его использование может снижать количество физически параллельно выполняемых транзакций. И наоборот, более низкий уровень изолированности позволяет выполнять больше параллельных транзакций, но снижает точность данных. Таким образом, выбирая используемый уровень изолированности транзакций, разработчик информационной системы в определённой мере обеспечивает выбор между скоростью работы и обеспечением гарантированной согласованности получаемых из системы данных.

**«Грязное» чтение**

Чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится).

**Неповторяющееся чтение**

Ситуация, когда при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными.

**Чтение «фантомов»**

Ситуация, когда при повторном чтении в рамках одной транзакции одна и та же выборка дает разные множества строк.

**11. Что такое DatabaseMetaData интерфейс?**

JDBC включает два интерфейса (DatabaseMetaData, ResultSetMetaData) для получения дополнительной информации MetaData о параметрах подключения к серверу БД, самой базе данных и результирующему набору, извлекаемому из базы данных.

Интерфейс DatabaseMetaData

Метод getMetaData текущего подключения Connection возвращает объект DatabaseMetaData.

DatabaseMetaData входит в состав пакета java.sql.\* и содержит методы, описывающие таблицы базы данных, поддержку SQL, хранимые процедуры и другие сведения, относящиеся к базе данных и текущему Connection, которые не относятся непосредственно к выполнению команд и извлечению результирующих наборов.

Типы всех столбцов таблицы БД можно получить с помощью метода getColumns класса DatabaseMetaData :

Методу getColumns необходимо передать четыре параметра: наименование каталога, наименование схемы, наименование таблицы и наименование столбца. В следующем коде из базы данных извлекаются метаданные колонок таблицы :

Значение имени каталога определено как null. Это указывает на то, что параметр должен быть удалён из поиска. Не все типы драйверов позволяют передавать имя каталога как null. Символ "%" определен для шаблона имени колонки, который означает, что необходимо получить ВСЕ колонки для таблицы.

**12. Зачем используют шаблон DAO?**

DAO (Data Access Object) — это объект, основная задача которого сохранять данные в базу данные, а также извлекать их из неё. DTO (Data Transfer Object) — это объект, предназначенный для транспортировки данных. Поэтому его основная задача — хранить эти данные. Никакой логики он не содержится. Кроме того, он должен быть сериализуемым, так как транспортировка объектов обычно происходит с помощью сериализации-десериализации.

