1. Как расшифровывается JDBC?

JDBC (англ. 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](https://ru.wikipedia.org/wiki/Java_SE" \o "Java SE)

1. Что такое JDBC драйвер? Какие бывают типы?
2. JDBC-ODBC bridge
3. Native-API driver
4. Network-Protocol driver ([Middleware](https://en.wikipedia.org/wiki/Middleware" \o "Middleware) driver)
5. Database-Protocol driver (Pure Java driver) or thin driver.

Драйвер JDBC - это программный компонент, позволяющий Java-приложению взаимодействовать с базой данных. Драйверы JDBC аналогичны драйверам ODBC, поставщикам данных ADO.NET и поставщикам OLE DB. Драйверы технологии JDBC вписываются в одну из четырех категорий:

1. 1. Мост JDBC-ODBC
2. 2. Драйвер Native-API
3. 3. Драйвер сетевого протокола (драйвер промежуточного ПО)
4. 4. Драйвер протокола базы данных (драйвер Pure Java) или тонкий драйвер.
5. Какие шаги (перечислите) нужно выполнить для извлечения данных?

JDBC основан на концепции так называемых драйверов, позволяющих получать соединение с базой данных по специально описанному [URL](https://ru.wikipedia.org/wiki/URL). Драйверы могут загружаться динамически (во время работы программы). Загрузившись, драйвер сам регистрирует себя и вызывается автоматически, когда программа требует [URL](https://ru.wikipedia.org/wiki/URL), содержащий протокол, за который драйвер отвечает.

1. Как и где задать параметры соединения?

Constructor of class Connection

1. Перечислите основные JDBC API компоненты.

JDBC [API](https://ru.wikipedia.org/wiki/API) содержит два основных типа интерфейсов: первый — для разработчиков приложений и второй (более низкого уровня) — для разработчиков драйверов.

Соединение с базой данных описывается классом, реализующим интерфейс java.sql.Connection. Имея соединение с базой данных, можно создавать объекты типа Statement, служащие для исполнения запросов к базе данных на языке [SQL](https://ru.wikipedia.org/wiki/SQL).

Существуют следующие виды типов Statement, различающихся по назначению:

* java.sql.Statement — Statement общего назначения;
* java.sql.PreparedStatement — Statement, служащий для выполнения запросов, содержащих подставляемые параметры (обозначаются символом '?' в теле запроса);
* java.sql.CallableStatement — Statement, предназначенный для вызова [хранимых процедур](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%B0).

Интерфейс java.sql.ResultSet позволяет легко обрабатывать результаты запроса.

1. Какая разница между еxecute, executeQuery, executeUpdate?

Интерфейс Statement предоставляет три различных метода выполнения SQL-выражений: executeQuery, executeUpdate иexecute, в зависимости от SQL-запроса.

Метод executeQuery необходим для запросов, результатом которых является один единственный набор значений, таких как запросов SELECT.

Метод executeUpdate используется для выполнения операторов INSERT, UPDATE или DELETE, а также для операторов DDL (Data Definition Language - язык определения данных), например, CREATETABLE 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.

Метод execute используется, когда операторы SQL возвращают более одного набора данных, более одного счетчика обновлений или и то, и другое.

1. Расскажите об интерфейсе ResultSet.

ResultSet содержит все строки, удовлетворяющие условиям в SQL-выражении и предоставляет доступ к данным в этих строках посредством набора get-методов, которые организуют доступ к колонкам текущей строки. Метод ResultSet.next используется для перемещения к следующей строке ResultSet, делая ее текущей.

Набор данных результата (result set) является таблицей с заголовками колонок и соответствующих значений, возвращенных запросом.

1. Как и для чего используют PreparedStatement?

Объект Statement используется для выполнения SQL-запросов к БД. Существует три типа объектов Statement. Все три служат как бы контейнерами для выполнения SQL-выражений через данное соединение: Statement, PreparedStatement, наследующий отStatement, и CallableStatement, наследующий отPreparedStatement.

PreparedStatement используется для выполнения прекомпилированных SQL-запросов с или без входных (IN) параметров;

1. Как и для чего используют транзакции?

Когда мы работает с JDBC, то по умолчанию наше соединение работает в режиме **auto-commit**, это означает, что каждый SQL – запрос будет выполнен и результаты будут сохранены в таблице нашей базы данных (далее – БД).

Для простых приложений это крайне удобно. Но, если мы хотим увеличить производительность, использовать распределённые транзакции, либо интегрировать бизнес-логику, то нам необходимо выключить режим **auto-commit**для управления нашими транзакциями.

Транзакцию дают нам возможность контролировать когда и где сохранять изменения в БД. Благодаря этому мы, например, можем объединить группу SQL – запросов в одну логическую группу и, если один из запросов не пройдёт – мы отменяем всю транзакцию.

Для того, чтобы получить доступ к управлению транзакциями, нам необходимо использовать метод setAutoCommit().

10. Перечислите и поясните типы чтения данных в транзакциях. Что

такое уровни изоляции?

*При*[*параллельном выполнении транзакций*](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B)*возможны следующие проблемы:*

* *потерянное обновление (*[*англ.*](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*lost update) — при одновременном изменении одного блока данных разными транзакциями одно из изменений теряется;*
* *«грязное» чтение (*[*англ.*](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*dirty read) — чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится);*
* *неповторяющееся чтение (*[*англ.*](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*non-repeatable read) — при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными;*
* *фантомное чтение (*[*англ.*](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)*phantom reads) — одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. Другая транзакция в интервалах между этими выборками добавляет или удаляет строки или изменяет столбцы некоторых строк, используемых в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк.*

**Уровень изолированности транзакций** — условное значение, определяющее, в какой мере в результате выполнения логически параллельных [транзакций](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) 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.

11. Что такое DatabaseMetaData интерфейс?

**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.