JDBC – это стандарт взаимодействия приложения с различными СУБД. JDBC основан на концепции драйверов, позволяющей получать соединение с БД по специальному url. JDBC API находятся в пакетах java.sql и javax.sql. С помощью JDBC API можно создавать соединения с БД, выполнять SQL запросы, хранимые процедуры и обрабатывать результаты. JDBC API упрощает работу с базами данных из Java программ.

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

Различают четыре типа драйверов.Java программа работает с БД в двух частях. Первая часть это JDBC API, а вторая – драйвер, который и выполняет всю работу. Каждый тип определяет реализацию JDBC драйвера по возрастающей степени независимости от платформы, производительности и легкости администрирования. Эти четыре типа следующие:

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.

JDBC Connection – соединение, установленное с сервером базы данных. Это своего рода сессия или Socket Connection. Для создания JDBC connection требуется всего два шага:

1) Зарегистрировать и загрузить драйвер с помощью Class.forName(). Класс драйвера будет зарегистрирован для DriverManager и загружен в память.

2) Используя DriverManager.getConnection() получить объект Connection. В метод необходимо передать URL базы данных, имя и пароль.

Для этого у интерфейса Statement существует три различных метода: executeQuery(), executeUpdate(), а так же execute().

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.

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.

По умолчанию объект ResultSet не модифицируемый и поддерживает курсор, который способен только к движение вперед.

Объект ResultSet автоматически закрывается при закрытии объекта, который его сгенерировал. Так же закрытие произойдет при повторном выполнении запроса или возврату результата из другого набора результатов.

Каждый SQL запрос является транзакционным и выполняя какие-либо DML или DDL запросы по их завершению изменения будут приняты (сохранены) базой данных.

Для этого предусмотрен метод объекта Connection rollback(), который откатывает транзакцию. Будут отменены все изменения в транзакции и отменен lock базы данных от этого объекта Connection.