|  |  |
| --- | --- |
| |  | | --- | | **JDBC Driver Types** | |

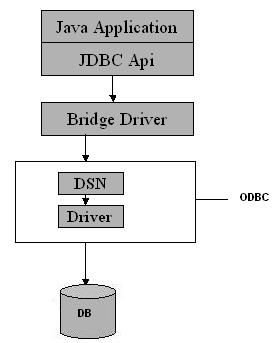
**JDBC drivers** are divided into four types or levels. The **different types of jdbc drivers** are:

**Type 1:** JDBC-ODBC Bridge driver (Bridge)  
**Type 2:** Native-API/partly Java driver (Native)  
**Type 3:** AllJava/Net-protocol driver (Middleware)  
**Type 4:** All Java/Native-protocol driver (Pure)  
  
**4 types of jdbc drivers** are elaborated in detail as shown below:

**Type 1 JDBC Driver**

**JDBC-ODBC Bridge driver**

The Type 1 driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver. ODBC is a generic API. The JDBC-ODBC Bridge driver is recommended only for experimental use or when no other alternative is available.



**Type 1: JDBC-ODBC Bridge**

**Advantage**

The JDBC-ODBC Bridge allows access to almost any database, since the database's ODBC drivers are already available.

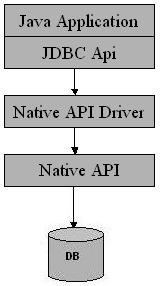
**Disadvantages**

1. Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable.  
2. A performance issue is seen as a JDBC call goes through the bridge to the ODBC driver, then to the database, and this applies even in the reverse process. They are the slowest of all driver types.  
3. The client system requires the ODBC Installation to use the driver.  
4. Not good for the Web.

**Type 2 JDBC Driver**

**Native-API/partly Java driver**

The distinctive characteristic of type 2 jdbc drivers are that Type 2 drivers convert JDBC calls into database-specific calls i.e. this driver is specific to a particular database. Some distinctive characteristic of type 2 jdbc drivers are shown below. Example: Oracle will have oracle native api.



**Type 2: Native api/ Partly Java Driver**

**Advantage**

The distinctive characteristic of type 2 jdbc drivers are that they are typically offer better performance than the JDBC-ODBC Bridge as the layers of communication (tiers) are less than that of Type  
1 and also it uses Native api which is Database specific.

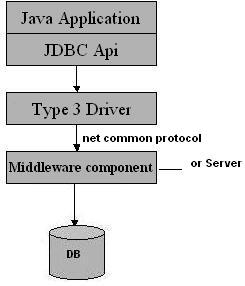
**Disadvantage**

1. Native API must be installed in the Client System and hence type 2 drivers cannot be used for the Internet.   
2. Like Type 1 drivers, it’s not written in Java Language which forms a portability issue.   
3. If we change the Database we have to change the native api as it is specific to a database  
4. Mostly obsolete now  
5. Usually not thread safe.

**Type 3 JDBC Driver**

**All Java/Net-protocol driver**

Type 3 database requests are passed through the network to the middle-tier server. The middle-tier then translates the request to the database. If the middle-tier server can in turn use Type1, Type 2 or Type 4 drivers.



**Type 3: All Java/ Net-Protocol Driver**

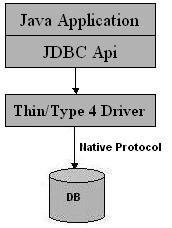
**Advantage**

1. This driver is server-based, so there is no need for any vendor database library to be present on client machines.  
2. This driver is fully written in Java and hence Portable. It is suitable for the web.  
3. There are many opportunities to optimize portability, performance, and scalability.   
4. The net protocol can be designed to make the client JDBC driver very small and fast to load.   
5. The type 3 driver typically provides support for features such as caching (connections, query results, and so on), load balancing, and advanced   
system administration such as logging and auditing.  
6. This driver is very flexible allows access to multiple databases using one driver.  
7. They are the most efficient amongst all driver types.  
  
**Disadvantage**

It requires another server application to install and maintain. Traversing the record set may take longer, since the data comes through the backend server.

**Type 4 JDBC Driver**

**Native-protocol/all-Java driver**

The Type 4 uses java networking libraries to communicate directly with the database server.  
  


**Type 4: Native-protocol/all-Java driver**

**Advantage**

1. The major benefit of using a type 4 jdbc drivers are that they are completely written in Java to achieve platform independence and eliminate deployment administration issues. It is most suitable for the web.   
2. Number of translation layers is very less i.e. type 4 JDBC drivers don't have to translate database requests to ODBC or a native connectivity interface or to pass the request on to another server, performance is typically quite good.   
3. You don’t need to install special software on the client or server. Further, these drivers can be downloaded dynamically.

**Disadvantage**  
  
With type 4 drivers, the user needs a different driver for each database.

**Question 4: What is 2 phase commit?**

Answer : This is one of the most popular JDBC Interview question and asked at advanced level, mostly to senior Java developers on J2EE interviews. Two phase commit is used in distributed environment where multiple process take part in distributed transaction process. In simple word we can understand **like if any transaction is executing and it will effect multiple database then two phase commit will be used to make all database synchronized with each other.**

In two phase commit, commit or rollback is done by two phases:

1.       **Commit request phase**: in this phase main process or coordinator process take vote of all other process that they are complete their process successfully and ready to commit if all the votes are “**yes**” then they go ahead for next phase. And if “No “then rollback is performed.

**2.**       **Commit phase:** according to vote if all the votes are yes then commit is done.

DB

DB

JDBC interview questions:

<http://javarevisited.blogspot.com/2012/12/top-10-jdbc-interview-questions-answers.html>

**Question 6: How cursor works in scrollable result set?**

Answer : Another  tough JDBC Interview question, not many Java programmer knows about using Cursor in Java.

in JDBC 2.0 API new feature is added to move cursor in resultset backward forward and also in a particular row .

There are three constant define in result set by which we can move cursor.

          **TYPE\_FORWARD\_ONLY**: creates a nonscrollable result set, that is, one in which the cursor moves only forward

          **TYPE\_SCROLL\_INSENSITIVE** : a scrollable result set does not reflects changes that are made to it while it is open

          **TYPE\_SCROLL\_SENSITIVE**: a scrollable result set  reflects changes that are made to it while it is open

Read more: <http://javarevisited.blogspot.com/2012/12/top-10-jdbc-interview-questions-answers.html#ixzz3OCJUkFwP>

**Question 3: What is the mean of “dirty read“ in database?**

Answer : This kind of JDBC interview question is asked on 2 to 4 years experience Java programmer, they are expected to familiar with [database transaction](http://javarevisited.blogspot.ca/2011/11/database-transaction-tutorial-example.html) and isolation level etc. As the name it self convey the meaning of dirty read *“read the value which may or may not be correct”.* in database when one transaction is executing and changing some field value same time some another transaction comes and read the change field value before first transaction **commit or rollback** the value ,which cause invalid value for that field, this scenario is known as **dirty read**.

**Question 4: What is 2 phase commit?**

Answer : This is one of the most popular JDBC Interview question and asked at advanced level, mostly to senior Java developers on J2EE interviews. Two phase commit is used in distributed environment where multiple process take part in distributed transaction process. In simple word we can understand like if any transaction is executing and it will effect multiple database then two phase commit will be used to make all database synchronized with each other.

In two phase commit, commit or rollback is done by two phases:

1.       **Commit request phase**: in this phase main process or coordinator process take vote of all other process that they are complete their process successfully and ready to commit if all the votes are “**yes**” then they go ahead for next phase. And if “No “then rollback is performed.

**2.**       **Commit phase:** according to vote if all the votes are yes then commit is done.

Similarly when any transaction changes multiple database after execution of transaction it will issue pre commit  command on each database and all database send acknowledgement and according to acknowledgement if all are positive transaction will issue the commit command otherwise rollback is done .

Read more: <http://javarevisited.blogspot.com/2012/12/top-10-jdbc-interview-questions-answers.html#ixzz3OCJai0hC>

**Question 9: What are the locking system in JDBC**

Answer : One more tough JDBC question to understand and prepare. There are 2 types of locking in JDBC by which we can handle multiple user issue using the record. if two user are reading the same record then there is no issue but what if users are updating the record , in this case changes done by first user is gone by second user if he also update the same record .so we need some type of locking so no lost update.

**Optimistic Locking:**optimistic locking lock the record only when update take place. Optimistic locking does not use exclusive locks when reading

**Pessimistic locking:** in this record are locked as it selects the row to update

Read more: <http://javarevisited.blogspot.com/2012/12/top-10-jdbc-interview-questions-answers.html#ixzz3OCK6zojq>

**Question 10: Does the JDBC-ODBC Bridge support multiple concurrent open statements per connection?**

Answer: No, we can open only one statement object when using JDBC-ODBC Bridge.

Read more: <http://javarevisited.blogspot.com/2012/12/top-10-jdbc-interview-questions-answers.html#ixzz3OCKTOGjd>