

Eg: Jdbc\_Example\_With\_Prepared\_Statement

//go through the code

Eg: Jdbc\_Example\_With\_Prepared\_Statement\_Eg2

// go through the code

Eg: Jdbc\_Example\_With\_Prepared\_Statement\_Eg3

// go through the code

Advantages of PreparedStatement:

1. Performance is very high compared to Statement approach because query will be compiled only once.

2. Since we don't send the query multiple times b/w java application and database traffic will be reduced.

3. inputs to the query need not be supplied at the beginning dynamically we can supply the inputs.

4. inputs to the query can be supplied just like java style, no need to perform formatting as per the DB specification.

5. Best suitable for inserting Date values.

6. Best suitable for insertion of BLOB's and CLOB's (image and pdf files).

7. It prevents SQLInjection Attack.

Limitation of PreparedStatement :

Statement stmt = connection.createStatement();

stmt.executeUpdate("insert into student values () ... ");

stmt.executeQuery("select \* from student");

stmt.executeUpdate("delete from student where .... ");

One statement object can be used to execute mulitple query but with no change in inputs.

PreparedStatement pstmt = connection.prepareStatement("select \* from student");

pstmt.executeQuery();

One PreparedStatement object is restricted to only one query, that query can be executed multiple times with change in input.

Which of the following represent valid statements ?

1. delete from employee where ename = ?(valid)

2. delete from employee ? ename = ? (invalid)

3. delete from ? where ename = ? (invalid)

4. delete ? employees where ename = ?(invalid)

Note: we can use ? only in the place of input values and we cannot use in the place of sql keywords, tablenames and column names.

Static query vs Dynamic query

The sql query without positional parameter(?) is called static query.

eg: delete from employee where ename = 'sachin';

The sql query with positional parameter(?) is called dynamic query

eg: delete from employee where ename = ?

select eid,ename,esal from employee where esal > ?

Note:

Simple Statement object can be used for static queries, where as Preparedstatement object can be used for static queries and dynamic queries also.

Handling Date Values For Database Operations:

Sometimes as the part of programing requirement, we have to insert and retrieve Date like

DOB, DOJ, DOM, DOP ... wrt database.

It is not recommended to maintain date values in the form of String, because comparisons will become difficult.

In Java we have two Date classes

1. java.util.Date

2. java.sql.Date

java.sql.Date is the child class of java.util.Date.

java.sql.Date is specially designed class for handling Date values wrt database.

Otherthan database operations, if we want to represent Date in our java program then we should go for java.util.Date.

java.util.Date can represent both Date and Time where as java.sql.Date represents only Date but not time.

Eg: Jdbc\_Date

util Date:Mon Mar 20 19:07:29 IST 2017

sql Date:2017-03-20

Differences between java.util.Date and java.sql.Date

java.util.Date

1) It is general Utility Class to handle Dates in our Java Program.

2) It represents both Data and Time.

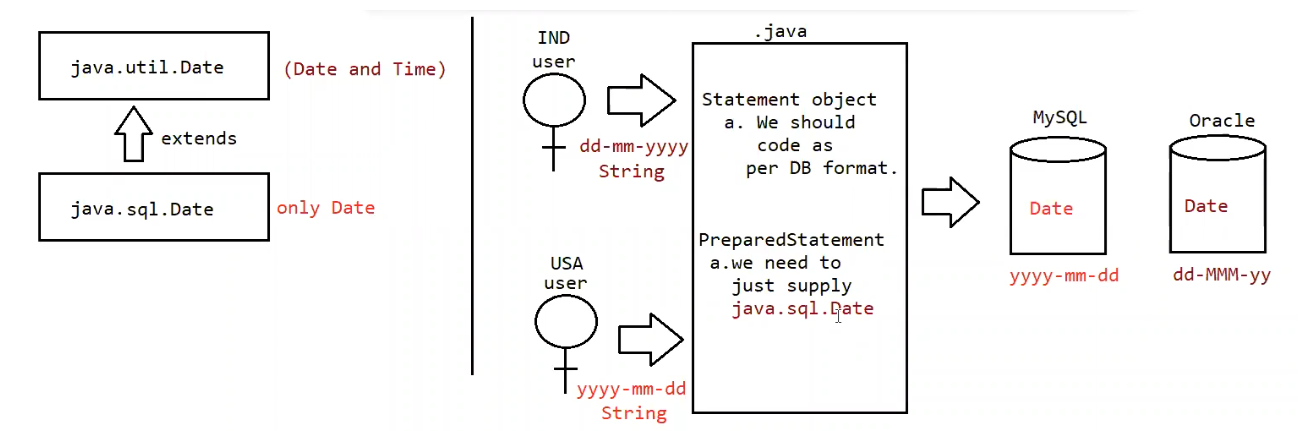
java.sql.Date

1) It is specially designed Class to handle Datesw.r.t DB Operations.

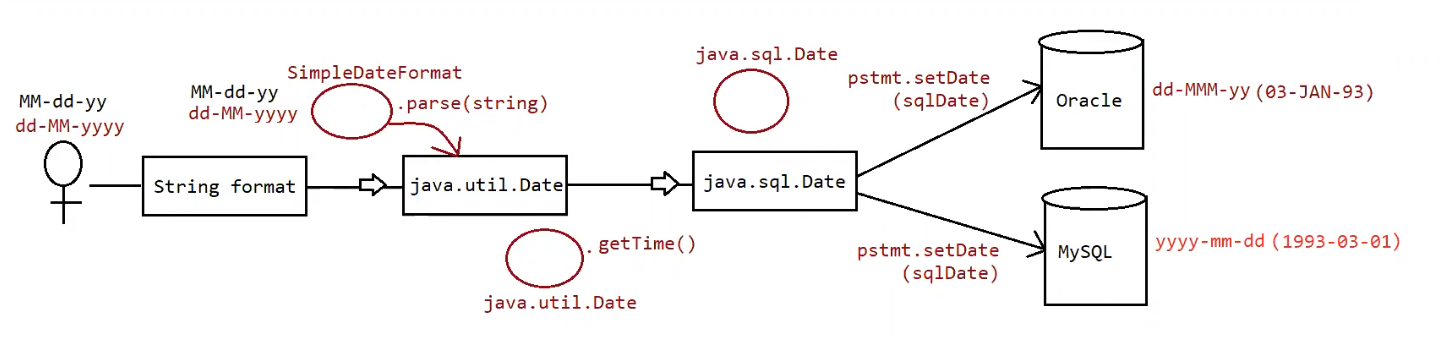
2) It represents only Date but not Time.

Note:

In sql package Time class is available to represent Time values and TimeStamp class is available to represent both Date and Time.



Eg: Convert\_String\_To\_Sql\_Date



-> Inserting Date Values into Database:

Various databases follow various styles to represent Date.

Oracle: dd-MMM-yy eg: 28-May-90

MySQL: yyyy-mm-dd eg: 1990-05-28

=> If we use simple Statement object to insert Date values then we should provide Date value in the database supported form.

=> If we use PreparedStatement, then we are not required to worry about database supported form, just we have to call

pst.setDate (2, java.sql.Date);

This method internally converts date value into the database supported format.

Hence it is highly recommended to use PreparedStatement to insert Date values into database.

Steps to insert Date value into Database:

=> DB: create table users (name varchar2(10), dobdate);

1. Read Date from the end user (in String form)

System.out.println("Enter DOP (dd-mm-yyyy):");

String dop=sc.next();

2. Convert date from String form to java.util.Date form by using SimpleDateFormat object.

SDF sdf = new SDF("dd-MM-yyyy");

java.util.Date udate = sdf.parse(dop);

3. convert date from java.util.Date to java.sql.Date

long | = udate.getTime();

java.sql.Date sdate=new java.sql.Date(I);

4. set sdate to query

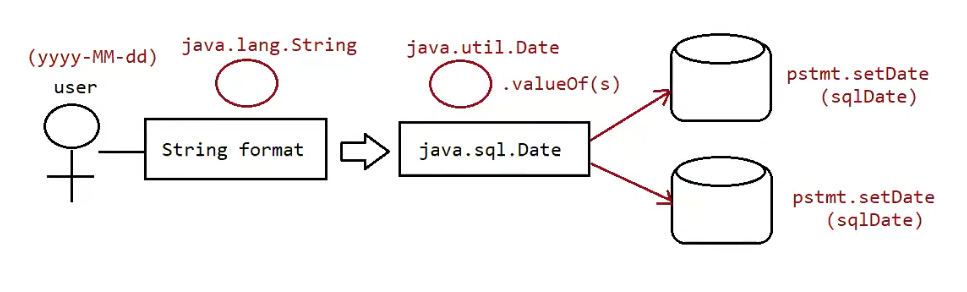
pst.setDate(2,sdate);

5. int rowAffected= pst.executeUpdate();//Execute the query.

Note: If end user provides Date in the form of "yyyy-MM-dd", then we can convert directly that String into java.sql.Date form as follows...

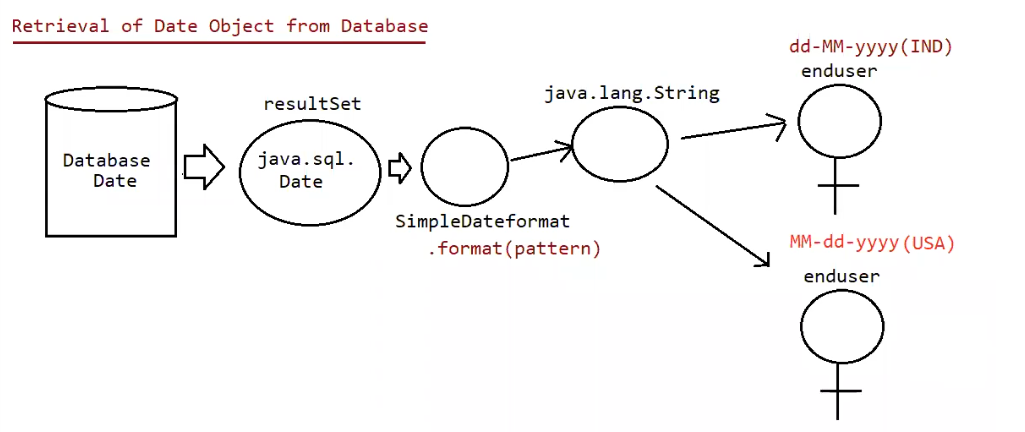
String s = "1980-05-27"; //Mysql

java.sql.Date sdate = java.sql.Date.valueOf(s);



Eg : Jdbc\_Convert\_String\_To\_Sql\_Date

// go through the code



Retrieving Date value from the database

=> For this we can use either simple Statement or PreparedStatement.

=> The retrieved Date values are Stored in ResultSet in the form of "java.sql.Date" and we can get this value by using getDate() method.

=> Once we got java.sql.Date object, we can format into our required form by using

SimpleDateFormat object.

Sequence

1. Database

(java.sql.Date)sqldate = rs.getDate(2);

2. Our required String Form

String s = sdf.format(sqldate);

3. String s holds the date.

Eg: Jdbc\_Convert\_Sql\_Date\_To\_String

Working with Large Objects (BLOB And CLOB)

Sometimes as the part of programming requirement, we have to insert and retrieve large files like images, video files, audio files, resume etc w.r.t database.

Eg: upload image in matrinomial web sites

I upload resume in job related web sites

To store and retrieve large information we should go for Large Objects (LOBs).

There are 2 types of Large Objects.

1. Binary Large Object (BLOB)

2. Character Large Object (CLOB)

1) Binary Large Object (BLOB)

A BLOB is a collection of binary data stored as a single entity in the database.

BLOB type objects can be images, video files, audio files etc.

BLOB datatype can store maximum of "4GB" binary data.

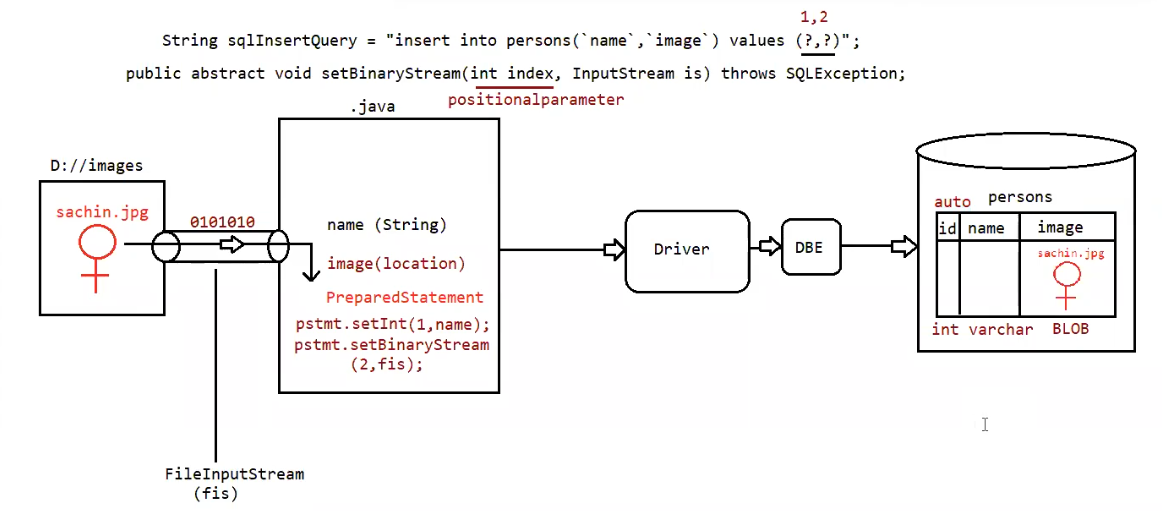
2) CLOB (Character Large Objects):

A CLOB is a collection of Character data stored as a single entity in the database.

CLOB can be used to store large text documents (mainly plain text or xml documents)

CLOB Type can store maximum of 4GB data.

Eg: resume.txt



Eg: Jdbc\_Image\_Insertion

Steps to insert BLOB type into database:

1. create a table in the database which can accept BLOB type data.

create table persons (name varchar2(10), image BLOB);

2. Represent image file in the form of Java File object.

File f = new File("sachin.jpg");

3. Create FilelnputStream to read binary data represented by image file

FilelnputStream fis = new FilelnputStream(f)

4. Create PreparedStatement with insert query.

PreparedStatement pst = con.prepareStatement("insert into persons values(?,?)");

5. Set values to positional parameters.

pst.setString(1,"katrina");

To set values to BLOB datatype, we can use the following method: setBinaryStream()

public void setBinaryStream(int index,InputStream is)

public void setBinaryStream(int index,InputStream is,int length)

public void setBinaryStream(int index,InputStream is,long length)

6. execute sql query

pst.executeUpdate();