

Steps to Retrieve BLOB type from Database

=========================================================

1. Prepare ResultSet object with BLOB type

ResultSet rs = st.executeQuery("select \* from persons");

2. Read Normal data from ResultSet

String name=rs.getString(1);

3. Get InputStream to read binary data from ResultSet

InputStream is = rs.getBinaryStream(2);

4. Prepare target resource to hold BLOB data by using FileOutputStream

FileOutputStream fos = new FOS("katrina\_new.jpg");

5. Read Binary Data from InputStream and write that Binary data to output Stream.

int i=is.read();

while(i !=- 1){

fos.write(i);

s.read();

}

(or)

byte[] b= new byte[2048];

while(is.read(b) > 0){

fos.write(b);

}

Eg: Jdbc\_Image\_Retrival

// go through the code

Eg: Jdbc\_Image\_Retrival\_Apache\_Lib

// go through the code

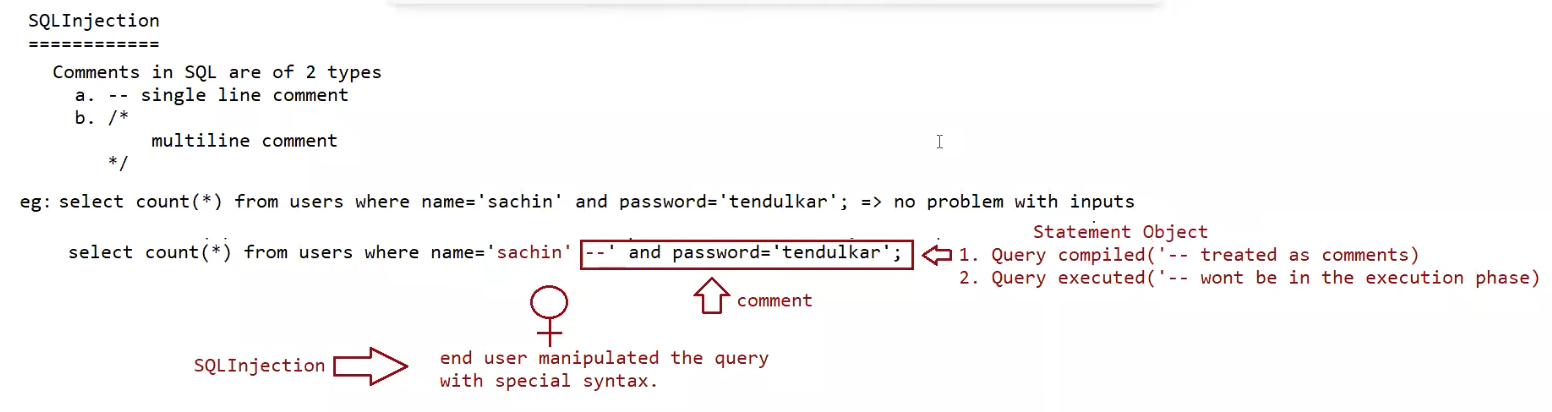


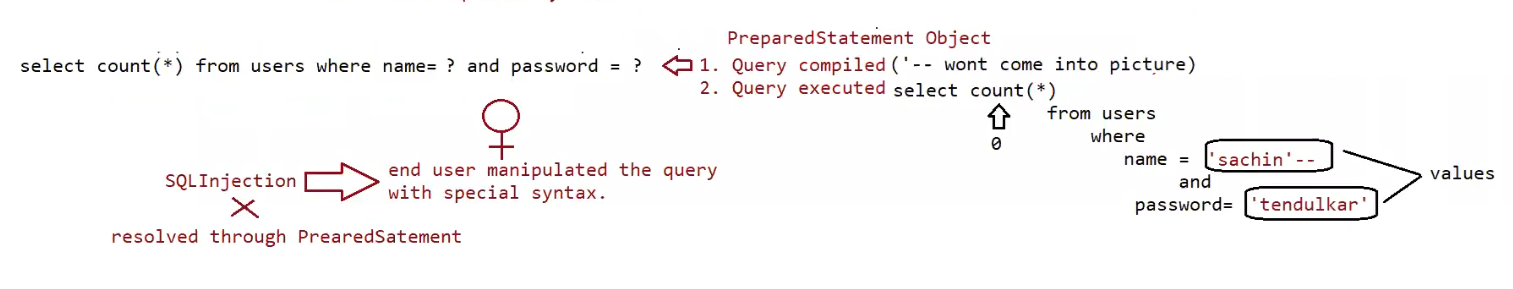
Eg: Jdbc\_Pdf\_Insertion\_To\_Database

// go through the code

Eg: Jdbc\_Pdf\_Retrival\_From\_Database

// go through the code





Eg: Sql\_Injection\_Statement\_Approach

// go through the code

Eg: Sql\_Injection\_PreparedStatement\_Approach

//go through the code

CallableStatement(I):

==================

Statement(I) => Executing the query (query should be written by java developer)

PreparedStatement(I) => Executing the query (query should be written by java developer) CallableStatement(I) => Executing the query (Query will be coded by DBA and present inside database)

If we want to execute only StoredProcedure can we use Statement/PreparedStatement?

Answer: No, we need to use "CallableStatement"

In programming if any code is repeatedly required, we can define the code inside the method and we can call that method

multiple times based on our requirement.

hence forth methods are best reusable component in programming.

Similarly in Database programming, if any group of sql statements is repeatedly required then we define those sql statements

in a single group and we call that group repeatedly based on our requirement.

This group of sql statements that perform a particular task is nothing but "Stored procedure".

StoredProcedure is the best reusable component at database level.

StoredProcedure

It refers to group of sql statements that perform particular task.

These stored procedures are stored permanently in database for future usage and hence the name "Stored procedure".

Usually StoredProcedures are created by DatabaseAdmin(DBA)

a. Oracle -> PL/SQL

b. MySQL -> Stored Procedure Language

c. MicrosoftSQLServer -> Transact SQL(TSQL)

Similar to methods stored procedures has its own parameters.

Stored Procedures has 3 parameters

a. IN parameters (to provide input values)

b. OUT parameters (to provide output values)

c. INOUT parameters (to provide and to collect output)

note: Generally procedures are written by the dba admin and given to the java developer

procedure:

DELIMITER $$

CREATE

PROCEDURE `pavan-workspace`.`P\_GET\_PRODUCT\_DETAILS\_BY\_ID`(IN id INT, OUT NAME VARCHAR (20),

OUT rate INT, OUT qnt INT)

BEGIN

SELECT pname, price, qty INTO NAME, rate, qnt FROM products WHERE pid = id;

END$$

DELIMITER;

DBA will run the query as shown below:

CALL `P\_GET\_PRODUCT\_DETAILS\_BY\_ID`(2,@name,@rate,@qty);

SELECT @name,@rate,@qty;

Syntax for calling stored procedure in java program

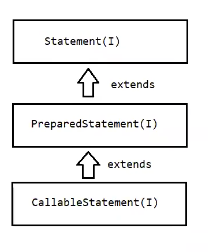
String storedProcedureCall = “{CALL P\_GET\_PRODUCT\_DETAILS\_BY\_ID(?,?,?,?)}”

CallableStatement cstmt = connection.prepareCall(storedProcedureCall);

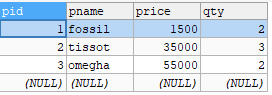
When jvm encounters the above line, jvm will send the call to database.

DB engine will check whether the sepcified stored procedure is available or not.

if it is available then it will return CallableStatement object representing that procedure.



Eg: Callable\_Statement



Note:

eg: CALL P\_GET\_PRODUCT\_DETAILS\_BY\_ID(?,?,?,?)

setXXXX() -> availabe to take care of setting the input values as per the DBengine datatypes.

eg: setInt(1,id) ----------- > int

setString(2,name) -- > varchar

Before getting the value from the storedprocedure, we need to register the out variables with java specific datatypes.

-> Types.INTEGER

-> Types.VARCHAR

Registering the output variables of StoredProcedure

a. To map the Java datatype and Database specific datatypes we need to use some mechanism.

b. The mechanism used is "JDBC Types" which is also known as "Bridge Types".

eg: java datatype -> int

JDBC type -> Types.INTEGER

DB datatype -> number

eg: java datatype -> String

JDBC type -> Types.VARCHAR2

DB datatype -> varchar, varchar2

eg: java datatype -> java.util.Date

JDBC type-> Types.DATE

DB datatype -> Date

getXXXX() -> available to get the value from the DBA as per the DB specific datatype

DB(varchar) ------ >JAVA(String)

Note

To execute stored procedure we use execute().

StoredProcedure to retrive all the records based on the product name

==================================================================

DELIMITER $$

CREATE

PROCEDURE `pavan-workspace`.`P\_GET\_PRODUCT\_DETAILS\_BY\_NAME`(IN name1 VARCHAR(20), IN name2 VARCHAR(20))

BEGIN

SELECT pid,pname,price,qty FROM products WHERE pname IN (name1,name2);

END$$

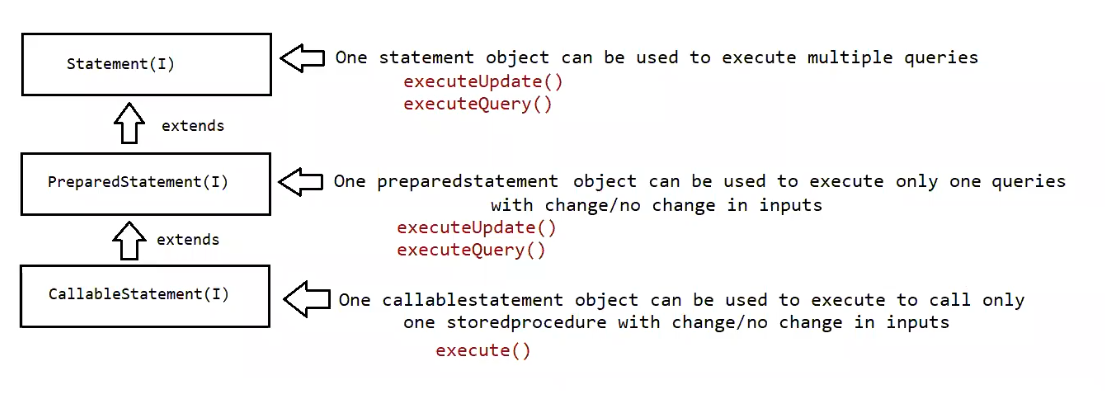
DELIMITER ;

To call storedprocedure we use the following syntax

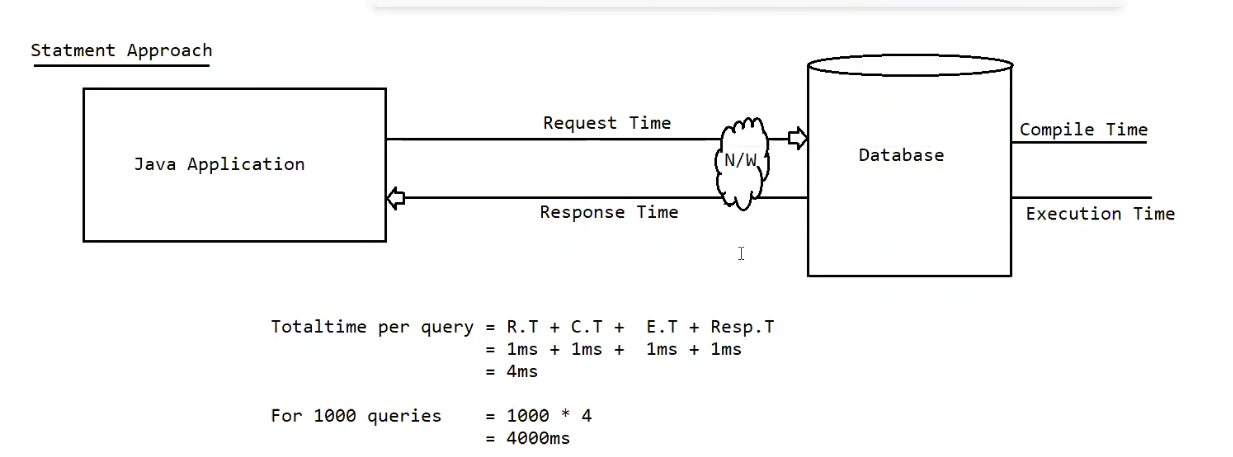
CALL `P\_GET\_PRODUCT\_DETAILS\_BY\_NAME` ('fossil','tissot');

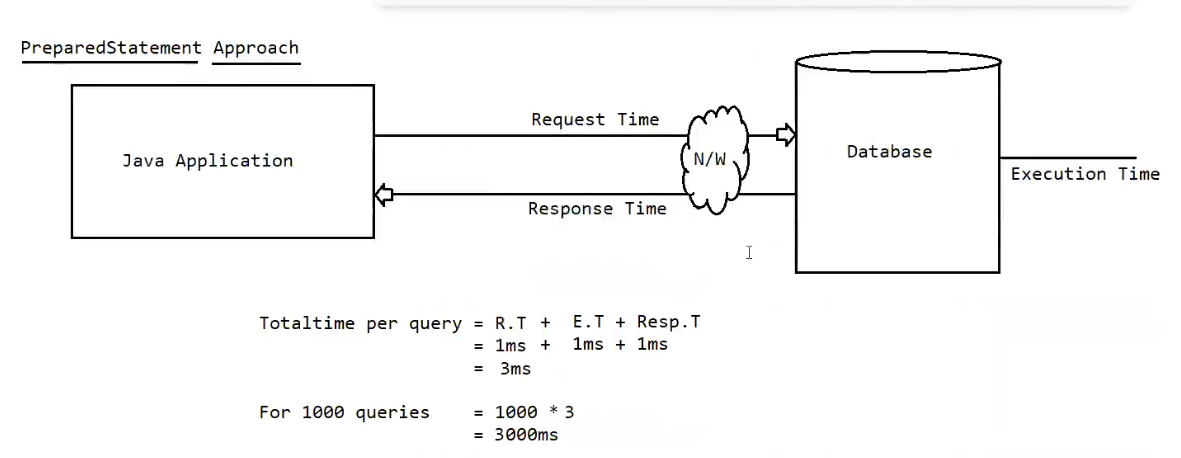
ResultSet is required since we need to get multiple records.

Eg: Callable\_Statement\_Eg2



Statement Approach:



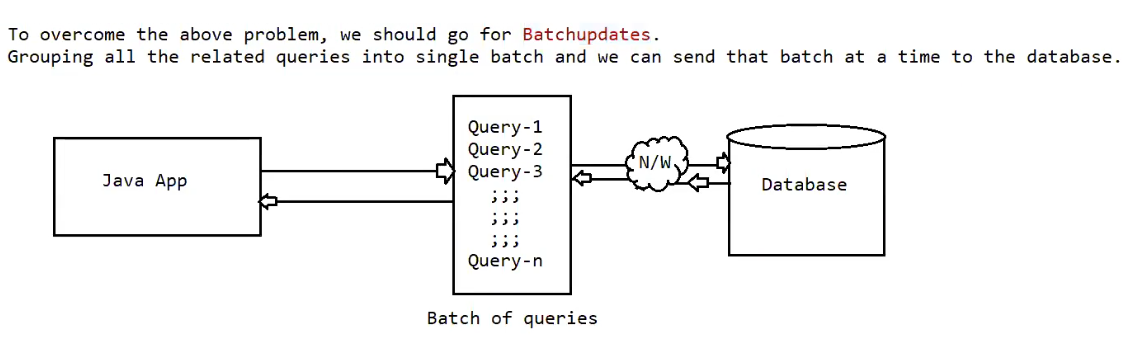


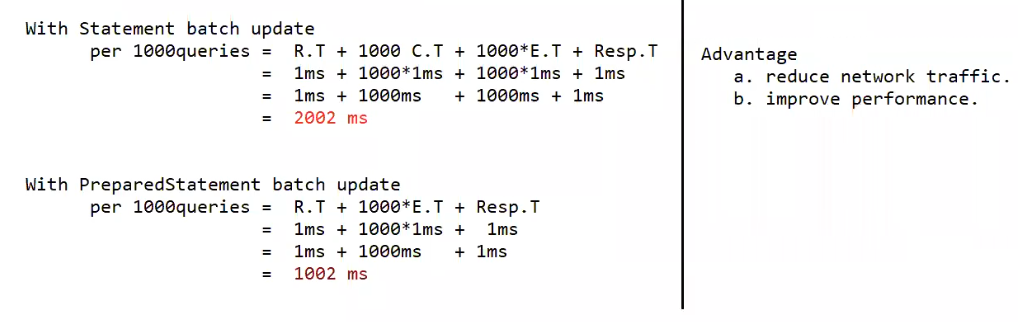
In the above 2 cases, we are trying to submit 1000 queries to the database one by one.

For 1000 queries to be submitted we need to hit the database 1000 times.

It increases the network traffic between java application and database.

It creates performance problems also.





Eg: Batch\_Update\_Using\_Prepared\_Statement

// go through the code

Transaction Management

Process of combining all the related operations into a single unit and executing on the rule "either all or none" is referred as

"Transaction Management".

Case1: Fund transfer

1. debit funds from sender account (update query with bal = bal-amt)

2. credit funds into receiver account (update query with bal = bal+amt)

All operations should be performed as single unit.

if money from sender account is debited, and if the money doesn't reach to receiver's account then there may be a chance of "Data inconsistency problem".

Case2: Movie Ticket Reservation

1. verify the status

2. Reserve the ticket

3. Payment

4. Issue ticket

All operations should be performed as single unit.

If some operations success and some operations fails then there may be "Data inconsistency problem".

Transaction Properties:

Every Transaction should follow the following four ACID properties

Atomicity:

Either all operations should be done or none

Consistency:

It should ensure bringing the database from one state to another state.

(Eg: where ever you go the performance should be same)

Isolation:

It ensures the transactions are isolated from other transactions

(eg: if you are making the payment with upi in one place and with the same account debit card the transaction is done by your family at the same time, both the transactions should be maintained in such a way that upi payment should have one thread and debit card payment should have another thread, even though they are operated on the same table of same database. Data inconsistency should never happen)

Durability:

It means once the transaction is committed, the results are permanent even in the case of system restart, errors etc.

What are the types of Transactions available?

a. Local Transaction

b. Global Transaction

a. Local Transaction

All operations in transaction are executed over the same database.

eg: Funds transfer from one account to another account where the both the accounts in the same bank.

b. Global Transaction

All operations in transaction are executed over the different database.

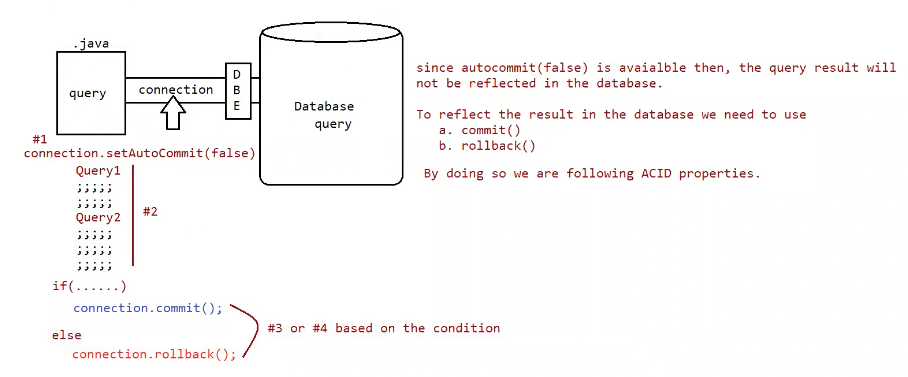
eg: Funds transfer from one account to another account where the accounts are related to different banks.

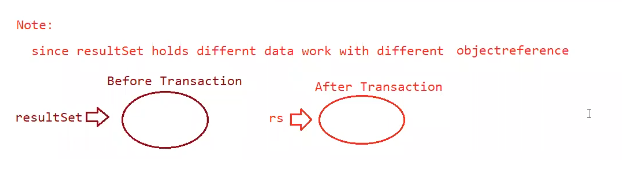
Note:

JDBC supports only local transactions.

If we want global transactions then we need to go for frameworks like Spring/hibernate or EJB's.

Process of transaction management:





1. Disable the autocommit nature of JDBC

connection.setAutoCommit(false);

2. If all operations are completed means then we can commit the transaction using the following method

connection.commit();

3. if any sql query fails, then we need to rollback the operations which are already completed using the following method

connection.rollback();

Eg: Jdbc\_Transaction\_App

create table paymentapp ( name varchar2(10) , amount int );

insert into paymentapp values ( 'ram' , 10000);

insert into paymentapp values ('lakshman' , 8000);