

Performing Basic CRUD Operations Using JDBC

Introduction



Sekhar Srinivasan

@sekharonline4u | sekhartheguru.net

Overview



- Executing Static SQL Statements
- Understanding ResultSet
 - Scrollable ResultSet
 - Updatable ResultSet
- Understanding Prepared Statements
 - Retrieve(Select)
 - Insert (Create)
 - Update
 - Remove(Delete)

Executing Static SQL Statements



Executing Static SQL Statements



Statements

ResultSet

Statement Interface

Used for executing a static SQL Statement

Methods:

- `ResultSet executeQuery(String SQL)`
- `int executeUpdate (String SQL)`
- `boolean execute(String SQL)`

Steps for Development Process

- Establish Connection to a Database
- Create Statement Object
- Execute SQL Query
- Process the ResultSet

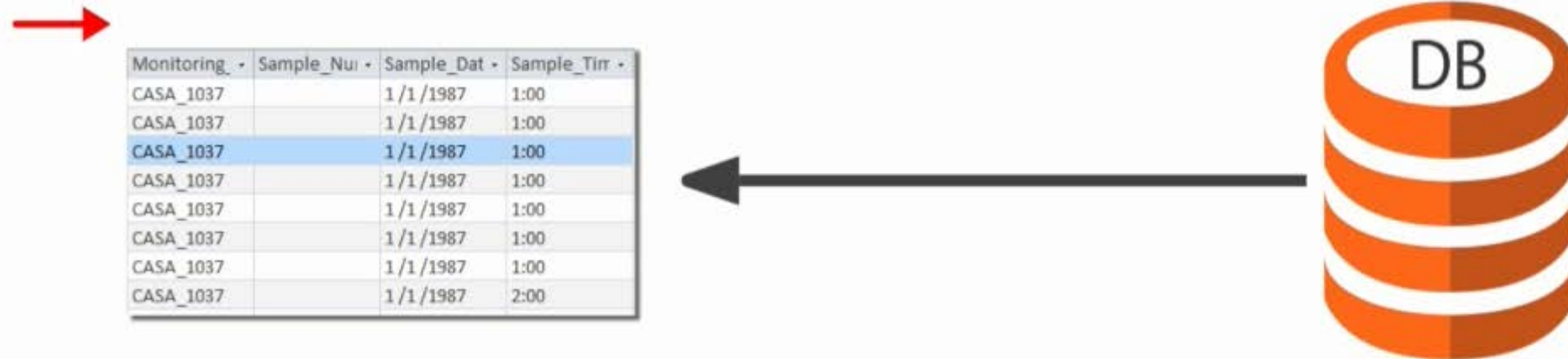
Demo



Executing Static SQL Statements

Iterating Through ResultSets

Java.Sql.ResultSet interface represents the Result Set of a database query



When the ResultSet is first returned, the starting cursor position is before the first row of data

Iterating Through ResultSets

Methods of the ResultSet can be divided into 3 categories

- i. Navigational Methods
- ii. Get Methods
- iii. Update Methods

Iterating Through ResultSets

Common methods of ResultSet interface

Symbol	Volume	ECN	ISLD	ARCA
CIPH	214,689	2,994	2,194	800
ATAR	147,015	21,000	16,200	4,800
UPD	103,299	28,010	22,410	5,600
CPHD	168,203	38,952	28,452	10,500
DDDC	131,871	49,466	46,666	2,800
FMSI	123,574	21,160	7,060	14,100
ARRS	140,511	95,822	17,622	78,200
MEDX	129,907	53,084	34,884	18,200
WIND	227,680	82,040	23,640	58,400
ANAD	112,249	29,210	22,810	6,400
SKIL	255,826	105,917	61,317	44,600
AZPN	203,132	50,925	27,325	23,600
TGAL	203,953	14,660	11,160	3,500
CHRD	104,130	18,100	4,800	13,300
DIGL	323,659	90,110	80,310	9,800

→ 08

beforeFirst()
afterLast()
first()
last()
previous()
next()
getRow()

Iterating Through ResultSets

ResultSet interface get methods

- get method for each data type
- Each get method has two versions
 - i. One that takes in a column name
 - ii. One that takes in a column index

Demo



Iterating through ResultSets

Understanding Scrollable ResultSets



Symbol	Volume	ECN	ISLD	ARCA
CIPH	214,689	2,994	2,194	800
ATAR	147,015	21,000	16,200	4,800
LIFD	103,299	28,010	22,410	5,600
CPHD	168,203	38,952	28,452	10,500
DDDC	131,871	49,466	46,666	2,800
PMSI	123,574	21,160	7,060	14,100
APRS	140,511	95,822	17,622	78,200
MEDX	129,907	53,084	34,884	18,200
WIND	227,680	82,040	23,640	58,400
ANAD	112,249	29,210	22,810	6,400
SKIL	255,826	105,917	61,317	44,600
AZPN	203,132	50,925	27,325	23,600
TGAL	203,953	14,660	11,160	3,500
CHRD	104,130	18,100	4,800	13,300
DIGL	323,659	90,110	80,310	9,800
JIMAR	144,195	22,995	22,195	800
CRAY	297,422	110,113	74,913	35,200
CPWR	231,490	71,211	28,111	43,100
VXGN	125,231	47,339	27,339	20,000
OCPI	102,743	21,387	12,387	9,000
CPST	292,305	116,475	110,675	5,800
INFA	458,817	63,506	44,406	19,100
DTAS	108,518	67,690	19,190	48,500
VRSO	220,475	95,783	23,683	72,100
SIGM	805,089	507,858	300,558	207,300
PMTG	114,964	40,350	11,750	28,600
PLUG	116,934	22,226	15,526	6,700
NTOP	159,386	32,984	13,484	19,500
SYKE	267,718	66,151	48,651	17,500
GMST	337,558	100,998	37,798	63,200

(ResultSet Object)



Understanding Scrollable ResultSets

Symbol	Volume	EQN	IBLD	ARCA
CPH	214.609	2.994	2.194	800
ATAP	147.015	21.000	16.200	4.800
LIFD	103.299	26.010	22.410	5.600
CPHD	168.203	36.952	28.452	10.500
DDOC	131.871	46.466	46.666	2.000
PMSI	123.574	21.180	7.080	14.100
APRS	140.511	95.822	17.622	78.200
MEDX	128.907	53.094	34.894	18.200
WIND	227.680	82.040	23.640	58.400
ANAD	112.249	29.210	22.010	6.400
SKL	295.826	105.917	61.317	44.600
AZPN	203.132	50.925	27.325	23.600
TGAL	203.953	14.600	11.160	3.500
CHRD	104.130	18.100	4.800	13.300
DIGL	323.659	90.110	60.310	8.000
IMAR	144.195	22.985	22.185	800
CRAY	297.402	110.113	74.913	35.200
CPWR	231.490	71.211	28.111	43.100
WGN	125.231	47.338	27.338	20.000
DOR	102.743	21.387	12.387	8.000
CPST	242.305	116.475	110.675	5.000
INFA	458.017	63.526	44.406	19.100
DTAS	108.518	67.690	19.190	48.500
VRGO	228.475	95.783	23.683	72.100
SIGM	895.089	587.058	300.558	207.300
PMTIC	114.964	40.358	11.758	28.600
PLUG	116.934	22.226	15.526	6.700
NTOP	158.388	32.884	13.484	19.500
SYKE	267.718	66.151	40.051	17.500
QMST	337.558	180.998	37.798	63.200

(ResultSet Object)



ResultSet Types

- Type_Forward_Only
- Type_Scroll_Insensitive
- Type_Scroll_Sensitive

ResultSet Concurrency Types

- Concur_Read_Only
- Concur_Updatable

Demo



Working with Scrollable ResultSets

Understanding Updatable ResultSets

Methods of the ResultSet can be divided into 3 categories

i) Navigational Methods

ii) Get Methods

iii) Update Methods

Understanding Updatable ResultSets

Updatable ResultSet allows modification to data in a table through the ResultSet

➤ Each Update method has two versions

- i) One that takes in a column name
- ii) One that takes in a column index

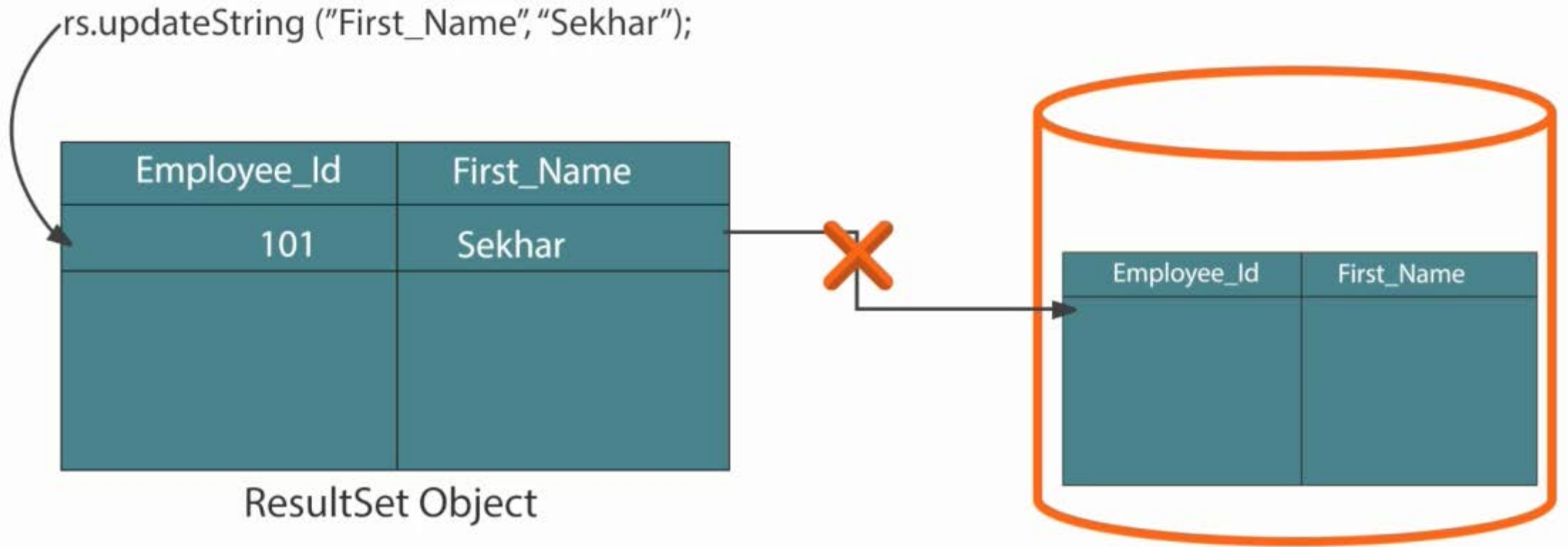
Understanding Updatable ResultSets

To Update a String Column of the current row

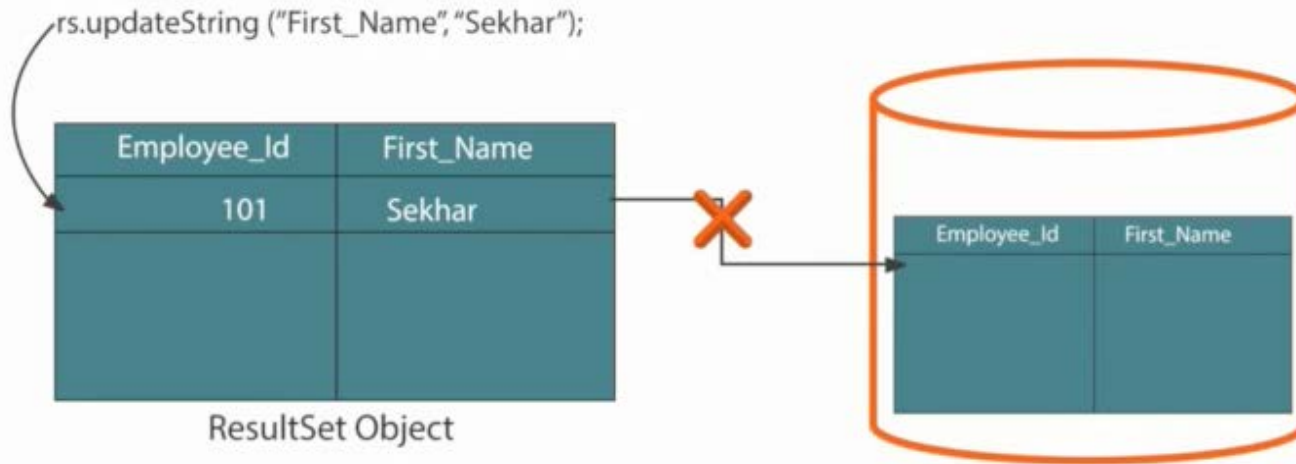
- `public void updateString (int columnIndex,String s)`
- `public void updateString (String columnName,String s)`

All update methods throws SQLException

Understanding Updatable ResultSets



Understanding Updatable ResultSets



`updateRow()`

`deleteRow()`

`refreshRow()`

`cancelRowUpdates()`

`insertRow()`

Demo



Managing Data Using Updatable ResultSets

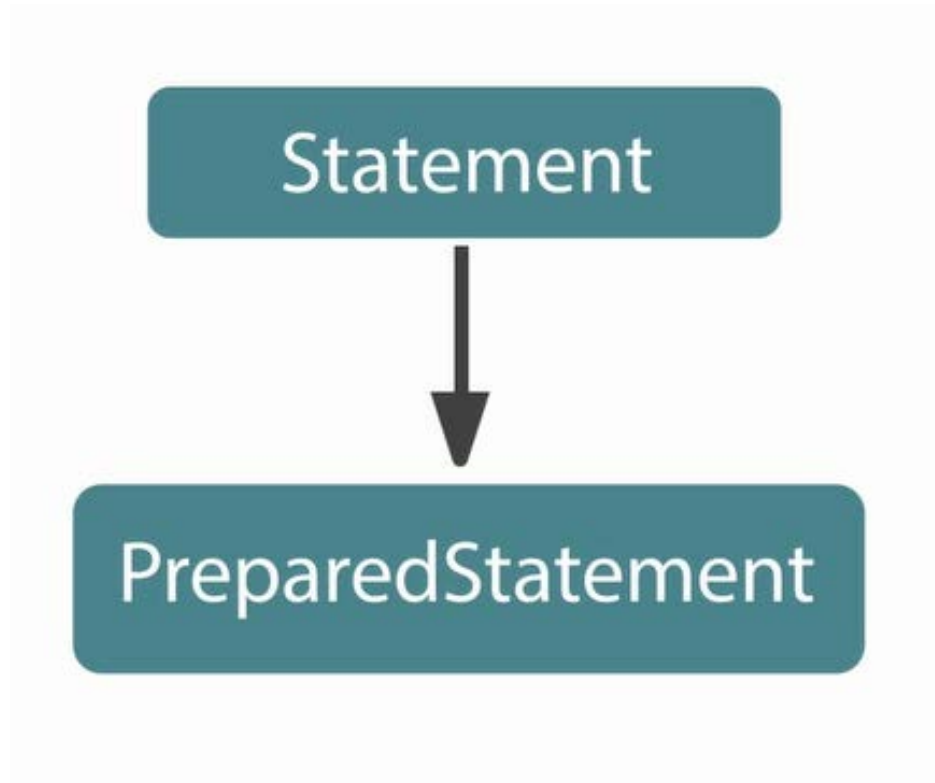
Understanding the PreparedStatement

- What is PreparedStatement
- How to create a PreparedStatement
- Setting Parameter Values
- Executing a PreparedStatement
- Reusing a PreparedStatement

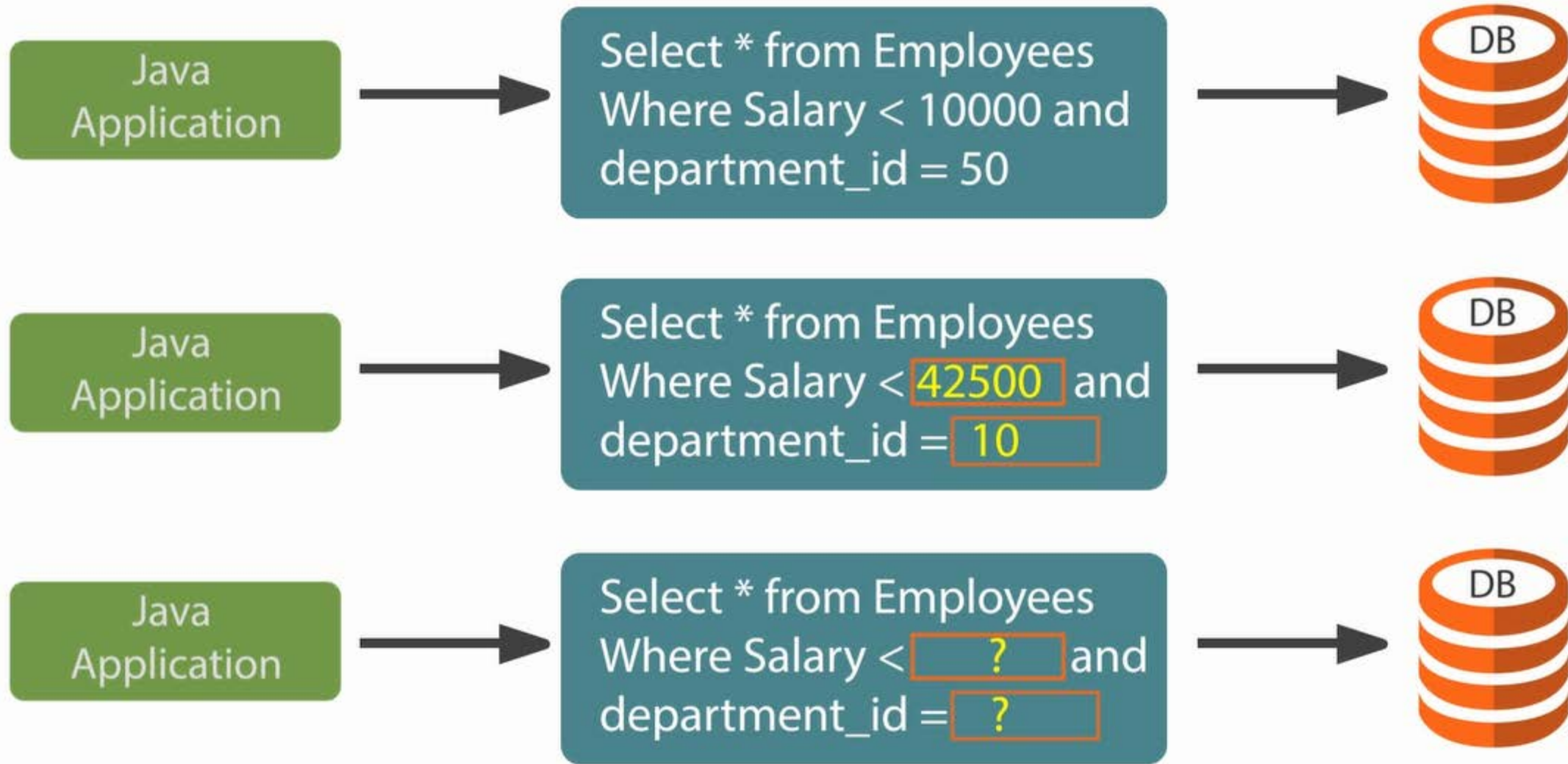
Understanding the PreparedStatement

Benefits

- Improve Performance of an Application
- Easy to Set SQL parameter Value
- Prevents SQL Dependency Injection Attacks



Understanding the PreparedStatement



Understanding the PreparedStatement

```
PreparedStatement pstmt = myConn.createStatement( "select *  
from employees where salary < ? and department_id= ?" );
```

To Bind :set Xxx() Method

For Example:setInt(P1,P2)

setString (P1,P2)

setDouble(P1,P2)

To Set:

```
pstmt.setDouble(1, 10000 );  
pstmt.setInt(2, 50 );
```

P1: Position(1 Based)

P2: Value

To Do



Step 1: Establish a Connection with the Database

Step 2: Prepare the Statement using Parameter Placeholder(?)

Step 3: Set the values for Parameters

Step 4: Executes the Statement

Demo



Retrieving Data Using PreparedStatement

Demo



Inserting the Record

```
SQL> conn hr/hr;  
Connected.
```

```
SQL> desc NewEmployees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
EMPLOYEE_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
HIRE_DATE	NOT NULL	DATE
SALARY		NUMBER(8,2)

```
SQL> _
```

```
Insert into TableName values ( value1, value2 , .... )
```

```
Insert into NewEmployees values ( 786,'sekhar',  
'sekharonline4u@gmail.com',2016-01-14',1000)
```

```
PreparedStatement pstmt = conn.prepareStatement  
(insert into NewEmployees values ( ?,?,?,?,? ))
```

Demo



Updating the Record

Demo: Updating the Record

NewEmployees

```
EMPLOYEE_ID  
EMPLOYEE_NAME  
EMAIL  
HIRE_DATE  
SALARY
```

Update NewEmployees

Set Salary = New Salary

where Employee_id= id

Demo: Updating
the Record

NewEmployees

```
EMPLOYEE_ID  
EMPLOYEE_NAME  
EMAIL  
HIRE_DATE  
SALARY
```

Update NewEmployees

Set Salary = ?

where Employee_id= ?

Demo



Removing the Record

Demo: Removing the Record

NewEmployees

```
EMPLOYEE_ID  
EMPLOYEE_NAME  
EMAIL  
HIRE_DATE  
SALARY
```

Delete from NewEmployees
where Employee_Id=

Demo: Removing
the Record

NewEmployees

```
EMPLOYEE_ID  
EMPLOYEE_NAME  
EMAIL  
HIRE_DATE  
SALARY
```

Delete from NewEmployees
where Employee_Id=?

Summary



- Statement
- ResultSet
- Scrollable ResultSet
- PreparedStatement
- CRUD Operations Using PreparedStatement