Java Database Queries

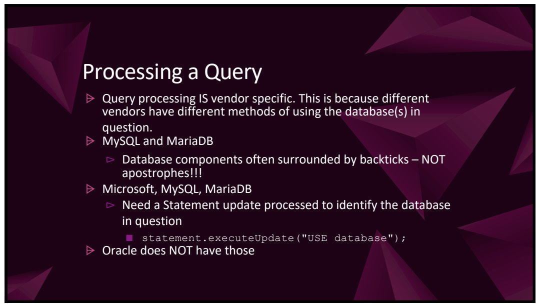
Advanced Java Spring 2025
Cody Henrichsen

1

Vocabulary

- Connection
- Indices
- ResultSet
- ResultSetMetaData
- Statement

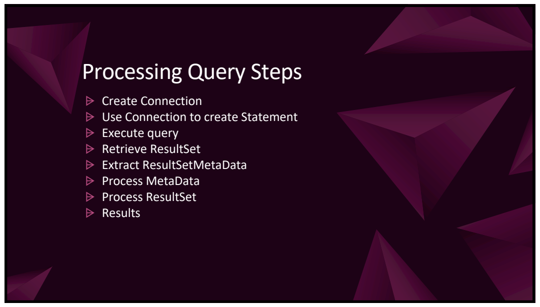
2



Processing a Query

- Query processing IS vendor specific. This is because different vendors have different methods of using the database(s) in question.
- MySQL and MariaDB
  - Database components often surrounded by backticks – NOT apostrophes!!!
- Microsoft, MySQL, MariaDB
  - Need a Statement update processed to identify the database in question
    - `statement.executeUpdate("USE database");`
- Oracle does NOT have those

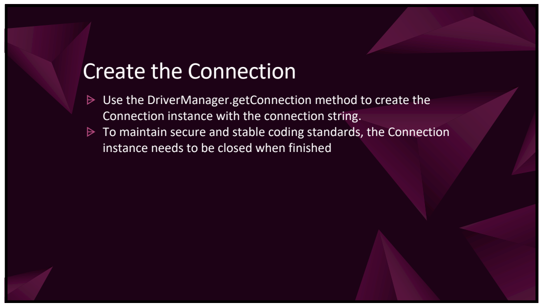
3



Processing Query Steps

- Create Connection
- Use Connection to create Statement
- Execute query
- Retrieve ResultSet
- Extract ResultSetMetaData
- Process MetaData
- Process ResultSet
- Results

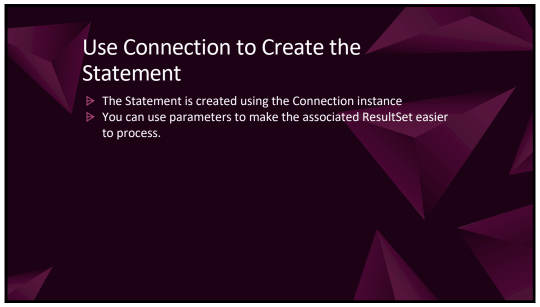
4



Create the Connection

- Use the DriverManager.getConnection method to create the Connection instance with the connection string.
- To maintain secure and stable coding standards, the Connection instance needs to be closed when finished

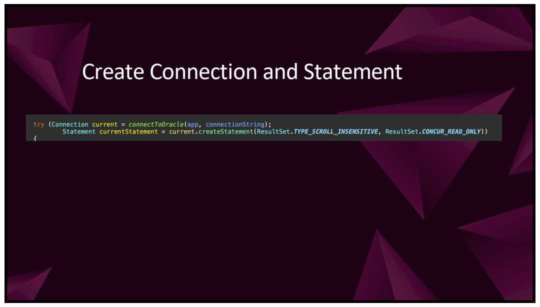
5



Use Connection to Create the Statement

- The Statement is created using the Connection instance
- You can use parameters to make the associated ResultSet easier to process.

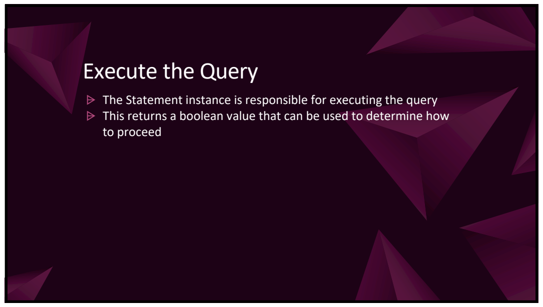
6



Create Connection and Statement

```
try (Connection current = connectToOracle(app, connectionString);
        Statement currentStatement = current.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY))
{
```

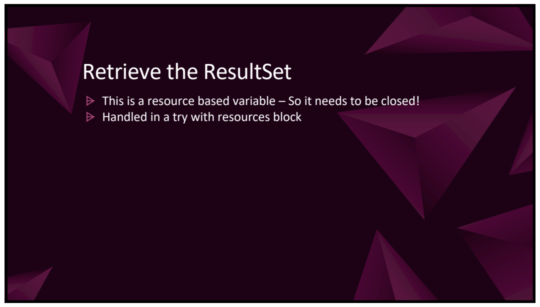
7



Execute the Query

- The Statement instance is responsible for executing the query
- This returns a boolean value that can be used to determine how to proceed

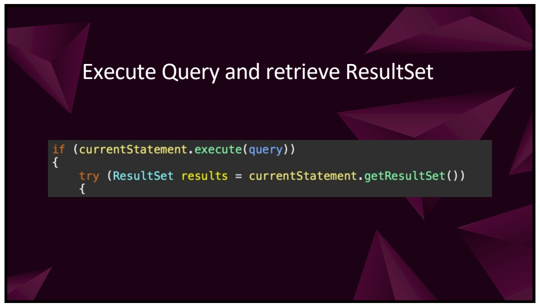
8



Retrieve the ResultSet

- This is a resource based variable – So it needs to be closed!
- Handled in a try with resources block

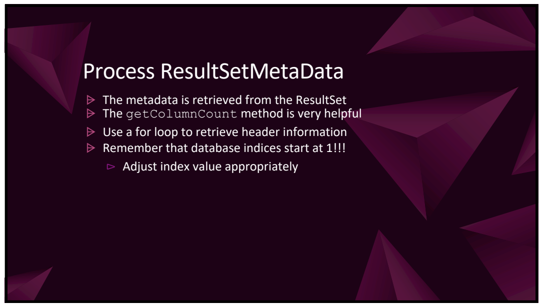
9



Execute Query and retrieve ResultSet

```
if (currentStatement.execute(query))
{
    try (ResultSet results = currentStatement.getResultSet())
    {
```

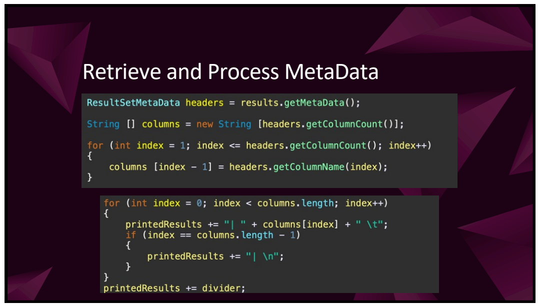
10



Process ResultSetMetaData

- The metadata is retrieved from the ResultSet
- The `getColumnCount` method is very helpful
- Use a for loop to retrieve header information
- Remember that database indices start at 1!!!
  - Adjust index value appropriately

11



Retrieve and Process MetaData

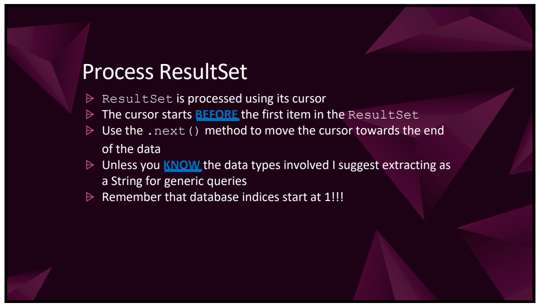
```
ResultSetMetaData headers = results.getMetaData();

String [] columns = new String [headers.getColumnCount()];

for (int index = 1; index <= headers.getColumnCount(); index++)
{
    columns [index - 1] = headers.getColumnName(index);
}
```

```
for (int index = 0; index < columns.length; index++)
{
    printedResults += "| " + columns[index] + " \t";
    if (index == columns.length - 1)
    {
        printedResults += "| \n";
    }
}
printedResults += divider;
```

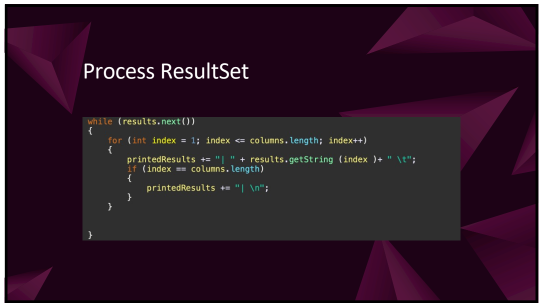
12



Process ResultSet

- `ResultSet` is processed using its cursor
- The cursor starts **BEFORE** the first item in the `ResultSet`
- Use the `.next()` method to move the cursor towards the end of the data
- Unless you **KNOW** the data types involved I suggest extracting as a String for generic queries
- Remember that database indices start at 1!!!

13

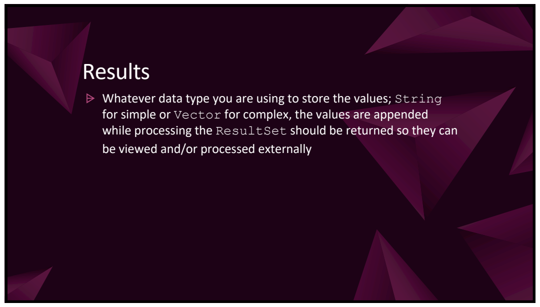


Process ResultSet

```java
while (results.next())
{
    for (int index = 1; index <= columns.length; index++)
    {
        printedResults += "| " + results.getString (index )+ " \t";
        if (index == columns.length)
        {
            printedResults += "| \n";
        }
    }


}
```

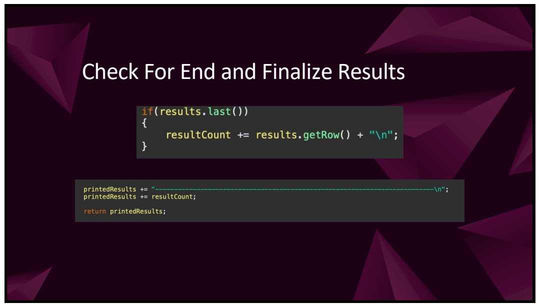
14



Results

- Whatever data type you are using to store the values; `String` for simple or `Vector` for complex, the values are appended while processing the `ResultSet` should be returned so they can be viewed and/or processed externally

15



Check For End and Finalize Results

```java
if(results.last())
{
    resultCount += results.getRow() + "\n";
}
```

```java
printedResults += "------------------------------------------------------------------------------\n";
printedResults += resultCount;

return printedResults;
```

16