## ResultSet

Resultset é uma classe da API JAVA que permite percorrermos um DataTable de alguma consulta em um banco de dados.

Ao ser inicializado, o Resultset coloca seu cursor na primeira linha do DataTable, o método next() permite que o ponteiro seja direcionado para a próxima linha caso exista.

Além de conseguirmos carregar dados de uma consulta a alguma tabela de um banco de dados, conseguimos armazenar vários DataTable's que podem ser por exemplo vários retornos de um procedure.

Em resumo :

ResultSet é uma interface utilizada para guardar dados vindos de um banco de dados.

Basicamente, ela guarda o resultado de uma pesquisa numa estrutura de dados que pode ser percorrida, de forma que você possa ler os dados do banco.

Exemplo:

```
Conection con = //de alguma forma vc pega conexao com o banco
Statement stmt = con.createStatement();
//aqui voce recebe um objeto ResultSet com todos os elementos
//da tabela cliente:
ResultSet rs = stmt.executeQuery("SELECT * FROM Clientes");
//para percorrer o resultset, faca:
while(rs.next()) {
  //pega o valor da coluna nome, de cada linha:
  String nome = rs.getString("Nome");
  //imprime no console:
  System.out.println("Nome do Cliente: " + nome);
}
```

Espero ter ajudado!

## Statement

Statement é uma interface utilizada para executar instruções SQL.

O método CreateStatement() cria uma instância de Statement para execução de SQL.

Em outras palavras, o Statement é o responsável por executar os teus códigos sql no banco de dados

A diferença vai além da simples adição de parâmetros.

A maioria dos bancos de dados relacionais lida com uma consulta (*query*) JDBC / SQL em quatro passos:

1. Interpretar (*parse*) a consulta SQL;
2. Compilar a consulta SQL;
3. Planejar e otimizar o caminho de busca dos dados;
4. Executar a consulta otimizada, buscando e retornando os dados.

Um *Statement* irá sempre passar pelos quatro passos acima para cada consulta SQL enviada para o banco.

Já um *Prepared Statement* pré-executa os passos (1) a (3). Então, ao criar um *Prepared Statement*alguma pré-otimização é feita de imediato. O efeito disso é que, se você pretende executar a mesma consulta repetidas vezes mudando apenas os parâmetros de cada uma, a execução usando *Prepared Statements* será mais rápida e com menos carga sobre o banco.

Outra vantagem dos *Prepared Statements* é que, se utilizados corretamente, ajudam a evitar ataques de Injeção de SQL. Note que para isso é preciso que os parâmetros da consulta sejam atribuídos através dos métodos `setInt()`, `setString()`, etc. presentes na interface `PreparedStatement` e não por concatenação de strings.

Para uma consulta que vai ser executada poucas vezes e não requer nenhum parâmetro, `Statement` basta. Para os demais casos, prefira `PreparedStatement`.