

# DBMS Final Project

Authors: Sonali Chavan, Abhinav Garg

# Introduction

- What is this project about?
- Why MF query vs Standard SQL?
  - Pivoting, making queries succinct
- MF/EMF input -> Project.java -> Query.java
- Dynamic Project.java (i.e. works with any table, any attributes)



# Tools Used

- Database: Postgresql
- DB connector JDBC
- IDE - Eclipse
- Programming language & output language: JAVA



# How does it work

- Change username password and database name at two places before running the project
  - qp.ConnectDB.java (at line number 18, 19, 20)
  - qp.CodeGenerator.java (at line number 41, 42, 43)
- Provide input as MF/EMF query (Input) using a text file
- Run Project.java
- Files generated in qp.output package
- Query.java -> (Output generation file)
- Run Query.java to see the output on the console



# Project Architecture

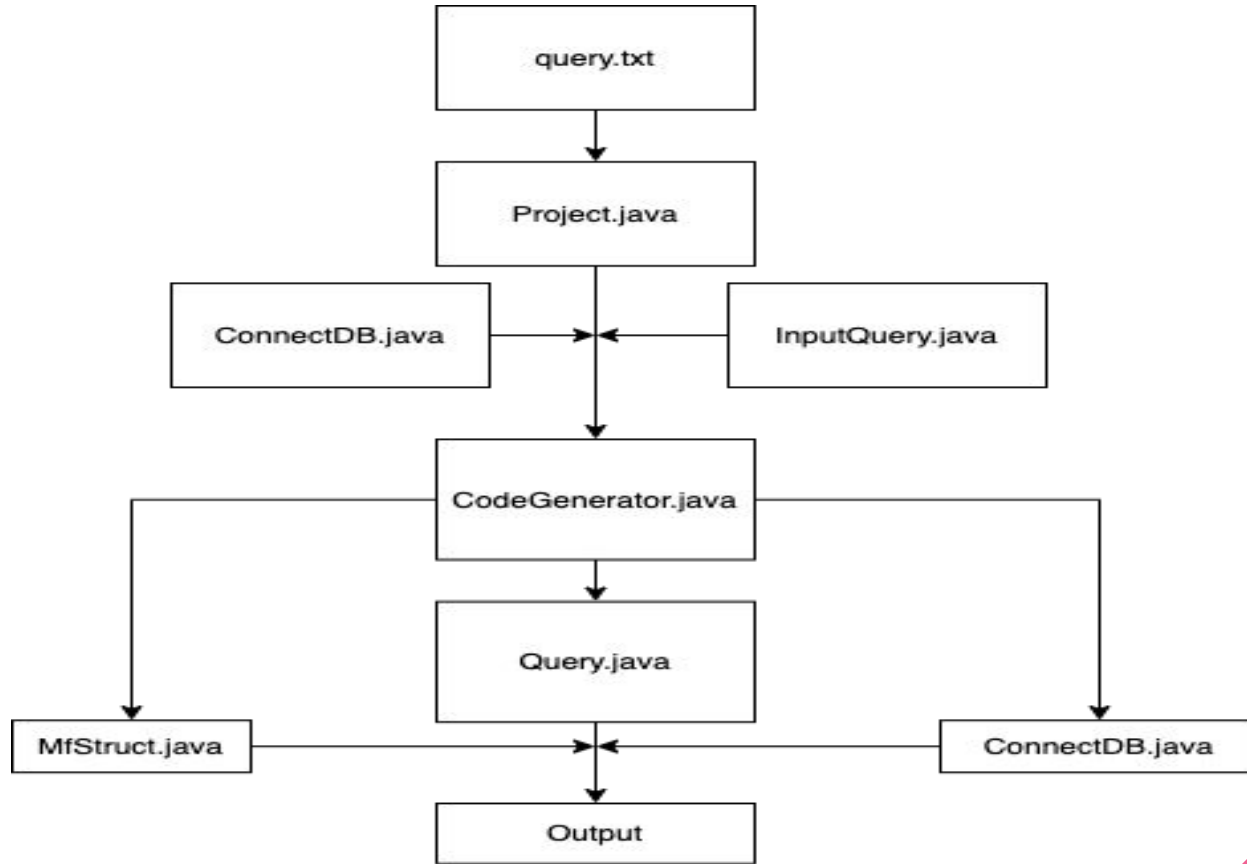
- ***qp package, in this package we have following classes,***
  - Project.java - main class that runs the whole project
  - ConnectDB.java - responsible for making the DB connection for fetching infoSchema
  - InputQuery.java - responsible for fetching input through text file and parsing it accordingly
  - CodeGenerator.java - file responsible for generating Query.java

## ***Code Generator***

- This file generates the output files in the qp.output package.
- It first makes the connection, then generates the MfStruct, and finally generates Query.java file, that runs the query and generates output



# Project Architecture(Cont)



# Project Architecture (Cont.)

## Input Query

### **SELECT ATTRIBUTE(S):**

cust, prod, count\_1\_quant, min\_2\_quant, sum\_2\_quant  
,max\_2\_quant

### **NUMBER OF GROUPING VARIABLES(n):**

2

### **GROUPING ATTRIBUTES(V):**

cust, prod

### **F-VECT([F]):**

count\_1\_quant, min\_2\_quant, sum\_2\_quant  
,max\_2\_quant

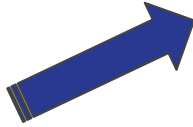
### **SELECT CONDITION-VECT([σ]):**

1.state="NY" and 1.quant<100

2.state="NJ"

### **HAVING\_CONDITION(G):**

sum\_1\_quant > 2 \* min\_2\_quant



## Parsed Input Query

**ArrayList Selected:** [cust, prod, count\_1\_quant, min\_2\_quant, sum\_2\_quant ,max\_2\_quant]

**int GV:** 2

**ArrayList Grouping Attrs:** [cust, prod]

**HashSet Function Vector:** [count\_1\_quant, min\_2\_quant, sum\_2\_quant ,max\_2\_quant]

**ArrayList Condition Vector:** [1.state="NY" and 1.quant<100, 2.state="NJ"]

**String Having:** sum\_1\_quant > 2 \* min\_2\_quant

# Generate MfStruct

Generating MfStruct from  
Code Generator, based on  
Input query.

```
package qp.output;
//MfStruct created using Grouping Attributes and F vectors

class MfStruct {
    String cust;
    String prod;
    int count_1_quant;
    int min_2_quant;
    int sum_2_quant;
    int max_2_quant;
    //Constructor
    MfStruct(String cust, String prod) {
        this.cust = cust;
        this.prod = prod;
        this.min_2_quant = Integer.MAX_VALUE;
    }
}
```



# Generate Main Code

## STEP 1:

Generate MF Struct (Previous Slide)

## STEP 2:

Iterate over Grouping Attributes and populate fields accordingly in the mfStruct

## STEP 3:

Iterate over each Grouping variable and calculate aggregate functions based on condition

## STEP 4:

Loop over mfStruct list and generate the code to print the output

```
while(rs.next()) {
    cust = rs.getString("cust");
    prod = rs.getString("prod");
    uniqueKey = (cust + prod).toLowerCase();
    if(!uniqueGAttr.contains(uniqueKey)) {
        uniqueGAttr.add(uniqueKey);
        newRow = new MfStruct(cust , prod);
        mfStruct.add(newRow);
    }
}

rs = st.executeQuery(queryStr);
while(rs.next()) {
    if(rs.getString("state").equals("NY") && rs.getInt("quant")<10) {
        cust = rs.getString("cust");
        prod = rs.getString("prod");
        for(MfStruct row: mfStruct) {
            if(row.cust.equals(cust) && row.prod.equals(prod)){
                row.count_1_quant++;
                row.sum_1_quant += rs.getInt("quant");
            }
        }
    }
}

rs = st.executeQuery(queryStr);
while(rs.next()) {
    if(rs.getString("state").equals("NJ")) {
        cust = rs.getString("cust");
        prod = rs.getString("prod");
        for(MfStruct row: mfStruct) {
            if(row.cust.equals(cust) && row.prod.equals(prod) && row.cust.equals(cust) && row.prod
```

# Output from Generated Code

Customer	Product	sum_2_quant	avg_1_quant	min_0_quant	max_1_quant	avg_0_quant
Emily	Fruits	8112	2463.0	111	3864	2696.85
Helen	Milk	1839	1589.4	77	4249	2103.25
Bloom	Eggs	5531	2164.0	559	3621	2504.82
Helen	Pepsi	16335	2507.67	1407	3002	2662.54
Emily	Yogurt	2336	2575.33	557	3822	2358.57
Sam	Milk	4522	1600.75	385	3120	1814.54
Helen	Yogurt	2096	2048.0	787	4909	2655.0
Knuth	Pepsi	9578	4252.33	277	4942	2646.23
Knuth	Eggs	3692	4050.5	326	4313	2538.1
Knuth	Yogurt	2167	3575.0	301	3705	2644.63
Emily	Cookies	8874	2463.5	173	3669	2672.81
Knuth	Soap	657	2790.67	436	4307	1856.0
Emily	Eggs	15380	2787.0	128	4754	2756.67
Bloom	Milk	9520	3306.33	553	4583	2764.69
Sam	Fruits	9847	3573.5	1311	4738	3387.9
Sam	Butter	12846	2697.67	673	4893	2701.5
Bloom	Fruits	6702	2332.75	26	3798	2471.24
Sam	Pepsi	18518	3993.0	160	4797	2988.31
Helen	Butter	15376	2685.29	163	4969	1908.77
Helen	Bread	4200	3381.67	389	4901	2781.33
Sam	Eggs	2205	2174.67	964	3363	2372.13
Emily	Coke	6807	2501.67	81	4258	2409.0

# Features

- Database table independent
- Grouping variables independent
- Modular design
- Usage of environment variables for DB username, password and table name
- Takes care of AND and OR conditions in condition vector
- Supports having clause




# Limitations and Prerequisites

## Limitations:

- Where clause is not supported
- If records are not present for any group, default Integer.MIN and Integer.MAX value is displayed in the output table for min, max conditions.
- Simple sql query without group by is not supported
- No Syntax checking in query file and presence of column in table

## Prerequisite:

- Condition vectors should be specified in numeric order
  - Grouping variables needs to numbers from 1 to n
  - Condition vector shouldn't have space between operator and operand e.g 1.quant > 30 is not allowed. It should be 1.quant>30 in the input query file
- 

*Thank you*

*Questions?*

