DBMS FINAL PROJECT

Team "Blast"
Xin Gao
Qiang Tong
Zhejun Huang
Yang Li

INTRODUCTION

- WHY? MF-Query vs. Normal SQL
- WHAT? QPE(Query Processing Engine)
- “Simple Compiler”
- Transfer source code into another computer language
- Parsed MF-SQL query → JAVA programming Language

summarize the project, compiler, definition of compiler.
It’s too expensive for normal SQL query to accomplish the request. Create handful of views, join them, too many file scans/disk access, too expensive...
While this project has advanced algorithm, it has a significant reduction in file scan / disk access times, much more efficient than normal sql.

PROJECT ARCHITECTURE & TEAMWORK

DBMS.java	The original main function, read a file from certain static directory, call other functions
GenerateMainCode.java	to generate “GeneratedCode.java” file,
GenerateMFStructCode.java	to generate “MFStruct.java file”,
InfoSchema.java	check the information_schema from the database to get all column names and their codes,
MFStructOrig.java	to deal with the original file input, and transfer format and store them into lists...
Pair.java	a little tool to help store information
SelectConVectConverter.java	to deal with “select condition-vect”
SWTWindow.java	manage to make a user interface, support input and
UI_Information.java	to store the DB information, like database URL, username, password.

PROJECT ARCHITECTURE & TEAMWORK (CONT.)

Input:
MF Queries Parameters:

```
SELECT ATTRIBUTE(S):
cust, prod, 1_sum_quant, 2_sum_quant, 3_sum_quant
NUMBER OF GROUPING VARIABLES(n):
3
GROUPING ATTRIBUTES(V);
cust, prod
F-VECT([F]):
1_sum_quant, 2_sum_quant, 3_sum_quant
SELECT CONDITION-VECT([]):
1.state='NY'
2.state='NJ'
3.state='CT'
```

MFStructOrig

ArrayList<String>
lst_Select_Attr

cust	prod	1_sum_quant	2_sum_quant	3_sum_quant

lst_Grouping_Attr

cust	prod

lst_FV

1_sum_quant	2_sum_quant	3_sum_quant

lst_Conditions

1.state = 'NY'	2.state = 'NJ'	3.state = 'CT'

int num_Grouping_Vari

3

MFStructOrig.java <= Xin Gao

PROJECT ARCHITECTURE & TEAMWORK (CONT.)

ArrayList<String>
lst_Select_Attr

cust	prod	1_sum_quant	2_sum_quant	3_sum_quant

lst_Grouping_Attr

cust	prod

lst_FV

1_sum_quant	2_sum_quant	3_sum_quant

lst_Conditions

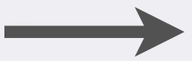
1.state = 'NY'	2.state = 'NJ'	3.state = 'CT'

int num_Grouping_Vari

3

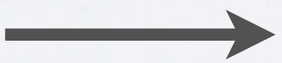
InfoSchema

table name "sales"



Type	Name
character vary => String	cust
character vary => String	prod
integer => int	day
integer => int	month
integer => int	year
integer => int	state
integer => int	quant

Pair



ArrayList<Pair>
lst_NumCondition

SelectConVectConverter

Type	1	2	3
Conditions	state = 'NY'	state = 'NJ'	state = 'CT'

SelectConVectConverter <= Gao
Pair <= Huang
InfoSchema <= Huang

PROJECT ARCHITECTURE & TEAMWORK (CONT.)

cust	prod	1_sum_quant	...

cust	prod

1_sum_quant	...

1.state = 'NY'	2.state = 'NJ'	...

3

GenerateMFStructCode

```
//MFStruct.java
public class MFStruct
{
      String cust;
      int sum_quant_1;
      int max_quant_2;
      int sum_quant_2;
      int count_1;
      int count_2;
      int count_3;

      public void initialization_1(String custTmp,int quant)
      public boolean equals(String custTmp)
      public void set_sum_quant_1(int quantTmp)
      ...
      //Other methods, avg, max, min...
      ...
}
```


GenerateMFStructCode.java <= Tong

PROJECT ARCHITECTURE & TEAMWORK (CONT.)

Static Code

```
//Sample pseudo code... Static Code
import ...

public class SampleCode
{
	static final String DB_URL = "jdbc:postgresql:...";
	static final String user = "postgres";
	//...
	static public void main(String arg[])
	{
		ArrayList<MFStruct>...
		try
		{
			conn = DriverManager.getConnection(...);
			System.out.println("[Results of the query]");
			String queryStr = "select * from sales";
			Statement st = conn.createStatement();
			ResultSet rs = st.executeQuery(queryStr);
			while(rs.next())
			{
				//...
			}
			...
```

GenerateMainCode

Dynamic Code

```
//while() loop need to be dynamically generated.

while(rs.next())
{
	String custTmp = rs.getString("cust");
	String prodTmp = rs.getString("prod");
	int yearTmp = rs.getInt("year");
	int quantTmp = rs.getInt("quant");

	if(lstFMFStruct.size() == 0)
	{
		//Initialization the list
	}
	for(int i = 0; i != lstFMFStruct.size(); i++)
	{
		if(lstFMFStruct.get(i).exists())
		{
			//update the list
		}
		if(i == lstFMFStruct.size() - 1)
		{
			//add to list
		}
	}
}
```


GenerateMainCode.java <= Li

TECHNIQUES

- Database: Postgresql
- Programming language & output language: JAVA
- Database connection: JDBC.
- IDE: Eclipse
- User Interface: Java SWT.

Java Standard Widget Toolkit.

ABOUT DEMO

- Input: both file input and user input
- Set input directory & output directory
- Generate two files: GeneratedCode.java, MFStruct.java
- Extra credits? “Independent from databases”, “Enhanced UI”...

LIMITATIONS & FORWARD LOOKING

- Error checking
- To-do: Generate C codes? Support having clauses? EMF-Queries? SQL Queries as input?

The technical limitations of the project,

SUMMATION

- Query processing engine.
- Input: MF-Query parameters.
- Output: Java codes.
- More efficient, lower cost.

summarize the project.
MFQuery is SOOOOOOO good.

THE END!
THANK YOU!

QR graph of our project URL on github.com.