EXTENDED SQL PROCESSOR

Fish & Grain:
Zachary Emmanuel Altuna & Emma Millet

STANDARD SQL LIMITATIONS & SOLUTION

Multi-dimensional queries are difficult to express in SQL

- Creates complex queries with multiple joins, group-bys, and sub-queries
- Traditional query optimizers focus on big picture → poor performance
- Large scale data analysis requires sophisticated and precise queries
  - Instead of focusing on refining the queries, why not focus on updating its expression and processing?

Solution? **Create a new syntactic framework that extends the *group-by* and *having* statements, adds a *such that* clause, and introduces a new relational operator Φ to provide an efficient and scalable algorithm to process multi-dimensional queries.**

AGENDA

01 High-Level Architecture & Tech Stack

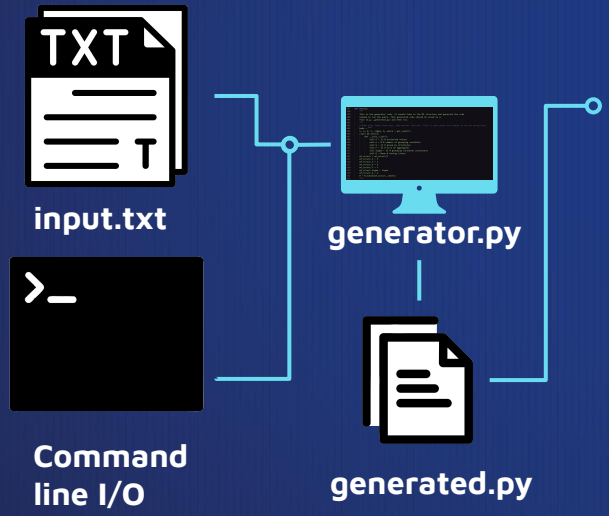
02 Query Structure

03 Technical Limitations

04 Demo & Reflections

HIGH LEVEL ARCHITECTURE

With Emma handling the parsing and overall logic of the program (I/O and MF) and Zachary working on the processing and handling of the query's MF properties, **generator.py** functions as follows:



The input is firstly parsed through to search for MF query properties, after which the MF structure (H table) is constructed and populated according to the where clause and form the results of the initial table scan

cust	f1_sum_quant	...	f2_avg_quant	...
"Dan"	0.0	...	0.0	...
"Claire"	0.0	...	0.0	...

Each consecutive scan computes the aggregate functions and updates the H Table with tuples that match the defining condition in the "SUCH THAT" clause.

cust	f1_sum_quant	...	f2_avg_quant	...
"Dan"	519.093632958802	...	0.0	...
"Claire"	487.033333333336	...	0.0	...

Finally, the last scan reduces the table to only distinct tuples which match the "HAVING" clause.

TECH STACK

Python
Programming Language

PostgreSQL
Database Host

psycopg2
PostgreSQL Database Adapter for Python

pgAdmin
Database GUI and SQL Processor

Tabulate
Library for Output Formatting

GitHub
Code Collaboration and Version Control

QUERY STRUCTURE

Our QPE parses queries and processes them as a class **mf_struct:**

- Projected Values → self.S = []
- Number of Grouping Variables → self.n = 0
- List of Group By Attributes → self.V = []
- List of Aggregates → self.F = []
- List of Predicates → self.sigma = []
- Predicates for the Having Clause → self.G = None

Additionally, our QPE passes the “WHERE” clause to our H_table function in order to properly query the specified tuples.

```
SELECT cust, sum(x.quant), sum(y.quant),
sum(z.quant)
FROM sales
GROUP BY cust : x, y, z
SUCH THAT x.state = ‘NY’ AND y.state = ‘NJ’
AND z.state = ‘CT’
HAVING sum(x.quant) > 2 * sum(y.quant) or
avg(x.quant) > avg(z.quant);
```

→

```
S: ['cust', 'f1_sum_quant', 'f2_sum_quant',
'f3_sum_quant']
n: 3
V: ['cust']
F: ['f1_avg_quant', 'f1_sum_quant',
'f2_sum_quant', 'f3_avg_quant',
'f3_sum_quant']
sigma: [' x.state = NY ', y.state = NJ', z.state = ‘CT']
G: f1_sum_quant > 2 * f2_sum_quant or
f1_avg_quant > f3_avg_quant
```

TECHNICAL LIMITATIONS

Lack of Error Checking

The program lacks error checking for the existence of tables, columns, and other items

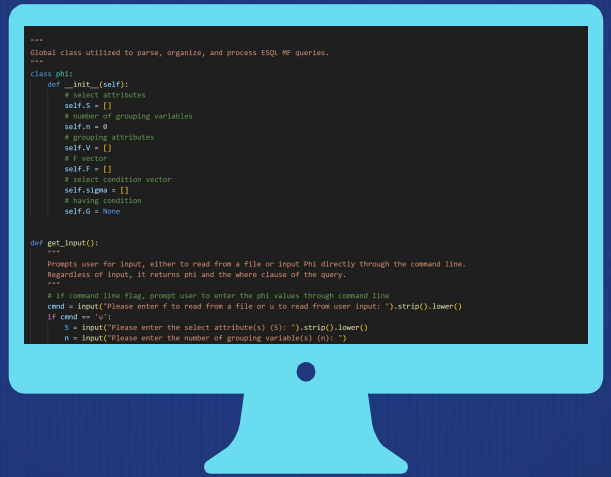
Lacks the Abilities of the EMF Syntax

Due to the nature of the MF syntax being lesser and less exhaustive than that of the EMF syntax, the program

Inability to handle complex SQL and MF query

The program can only handle simple SQL queries (i.e. those without statements such as WITH and JOIN) and MF queries with basic arithmetic operations (i.e. queries with multiple operations in the various clauses where MF properties are present)

PROCESSOR DEMO



PROJECT SIGNIFICANCE & LOOKING FORWARD

Scalability

This algorithmic foundation could be utilized in much larger, robust processors, allowing for computation at a fraction of the original cost.

Multi-Table Functionality

In the future, we would scale to multi-table functionality to boost the computation of much more complex queries in a full database

Efficiency

This QPE can process ESQL in under a second, vastly boosting efficiency for complex SQL queries on standard operating systems

EMF Queries

Additionally, we would add support for EMF (Extended Multi Feature) to allow GROUP BY values in conditions and variables with scope of whole relation rather than a group

THANK YOU!

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution