RASQL: A Graphical Query Language For Recipes

Jolly Chen

Recipes:

visual programs that can be executed in an integrated data management and visualization environment -- Tioga

A recipe is represented by a graph consisting of:

- ingredients
- browsers
- flow edges

main idea of RASQL:

extend recipe definition language so that the queries are themselves recipes with special query ingredients and execution semantics

Ingredients:

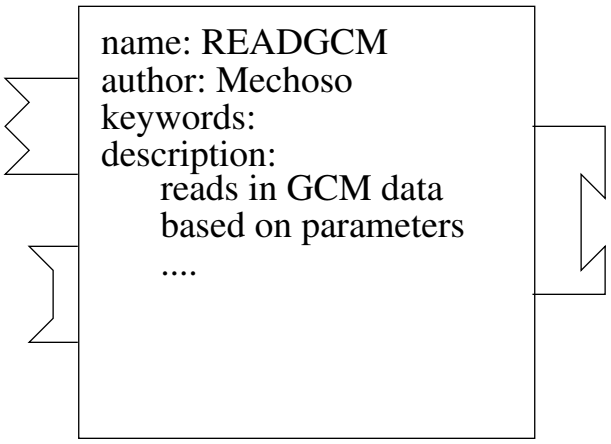
opaque units of data transformation

functions registered with POSTGRES

can be C, POSTQUEL, compiled recipes

typed in-ports and out-port, depicted by icons

Example:



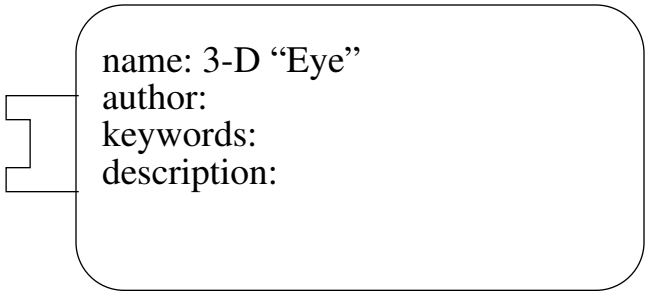
Browsers:

visualization functions that interact with the user

separate user process outside the control of the recipe management system

fetches data from the recipe system with a portal-like protocol

Example:

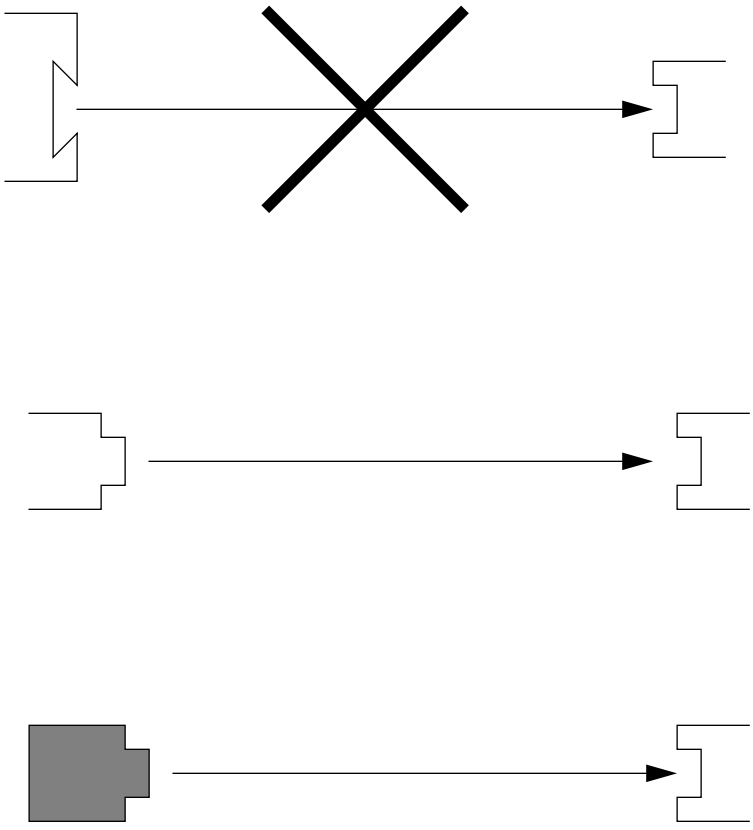


Flow edges:

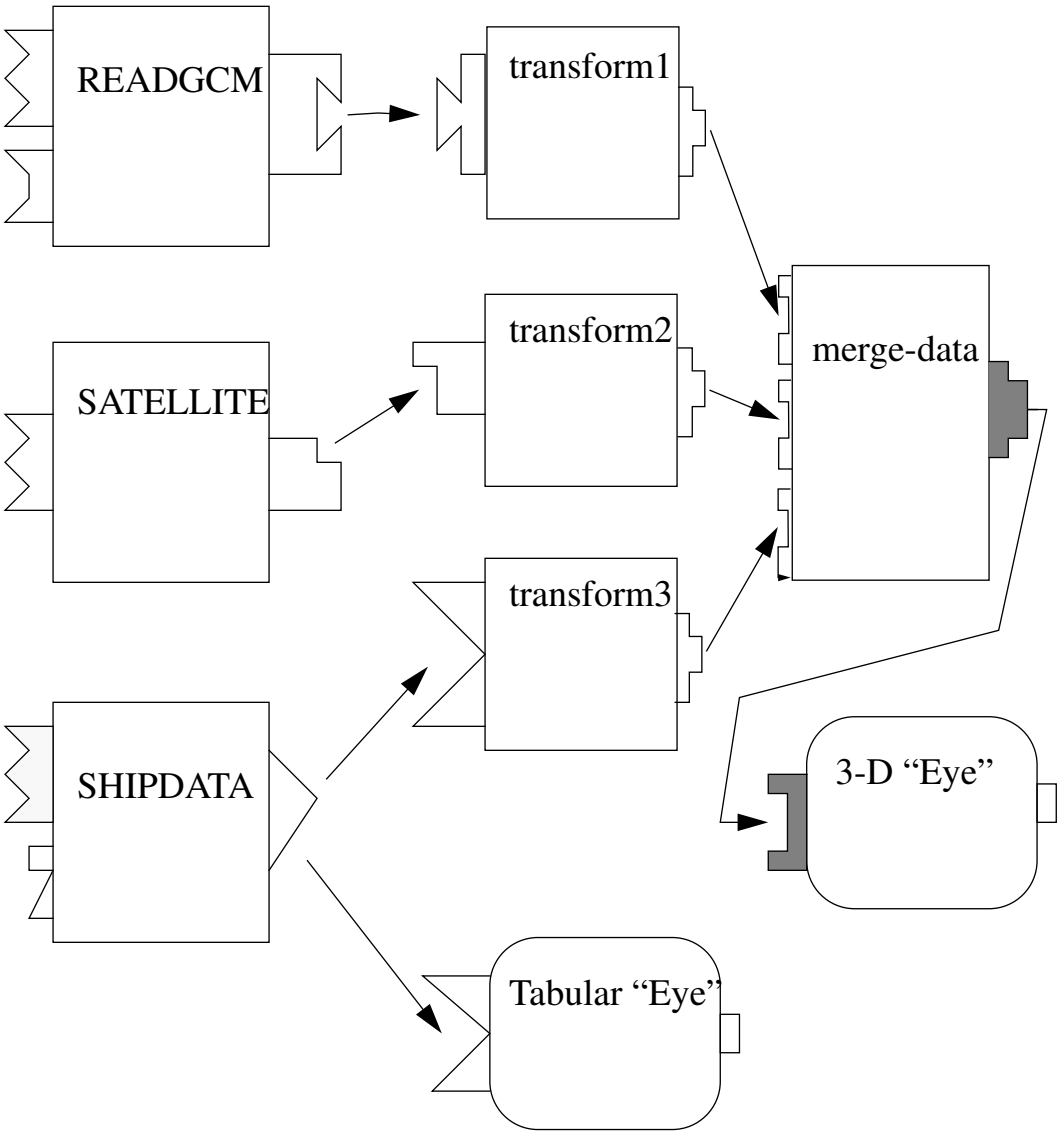
represent directed flow of typed data

fan-out allowed; cycles are not

recipe system performs type-checking



Example of a recipe:



Querying recipes:

cumbersome to specify queries for recipes in textual query language

consider:

```
retrieve R.all from R in recipe where
     contains(R,Func1) and
     contains(R,Func2) and
     contains(R,Func3) and
     childOf(Func1,Func2) and
     childOf(Func3,Func2) and
     childOf(Func1,Func3) and
     outputType(Func1) == "GCM" and
     inputType(Func2) == "DATE" and
     ...
```

RASQL - a graphical query language

since recipes already have a graphical definition language, extend it for queries as well

common user interface makes it easy for users who create recipes to also write queries for recipes

RASQL queries are recipes with *query ingredients.*

Three kinds of query ingredients:

- delimiter ingredients
- template ingredients
- operator ingredients

Query ingredients follow recipe syntax but have different semantics

Delimiter Ingredients:

bracket a subgraph of ingredients and indicate that the subgraph is to be used for matching, not execution

beginning delimiter ingredient "{":

- an in-port of type `setof recipe`
- an out-port of type `any`

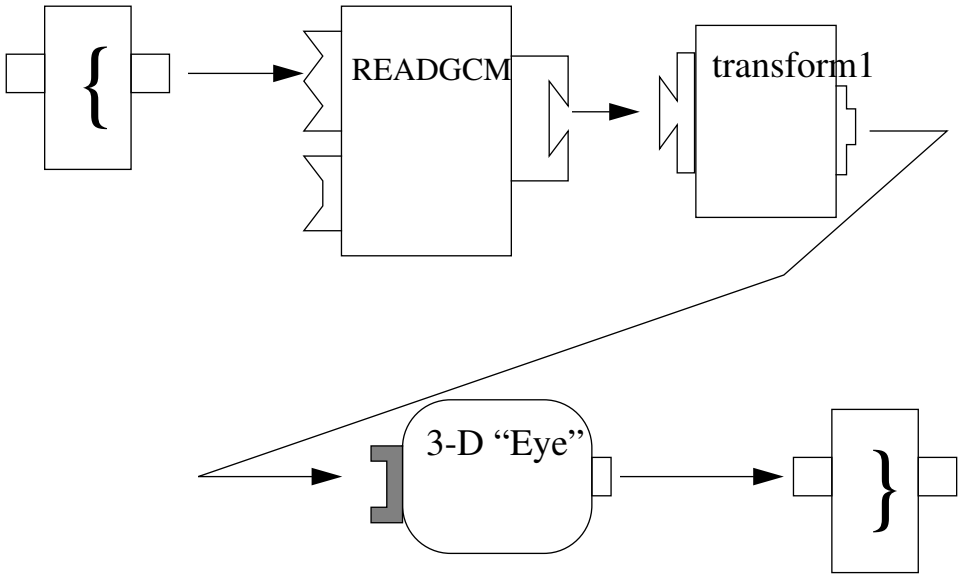
ending delimiter ingredient "}":

- an in-port of type `any`
- an out-port of type `setof recipe`

semantics:

result of ending delimiter is a set of all recipes whose ingredients and edges exactly match the ingredients and edges inside the delimiting ingredients.

Example of use of delimiter ingredients:



Template Ingredients:

with delimiter ingredients, we can query recipes by doing exact matching

template ingredients allow for queries where the exact ingredient is unknown but some attributes of the desired ingredient are known.

use regular expressions for textual ingredient attributes

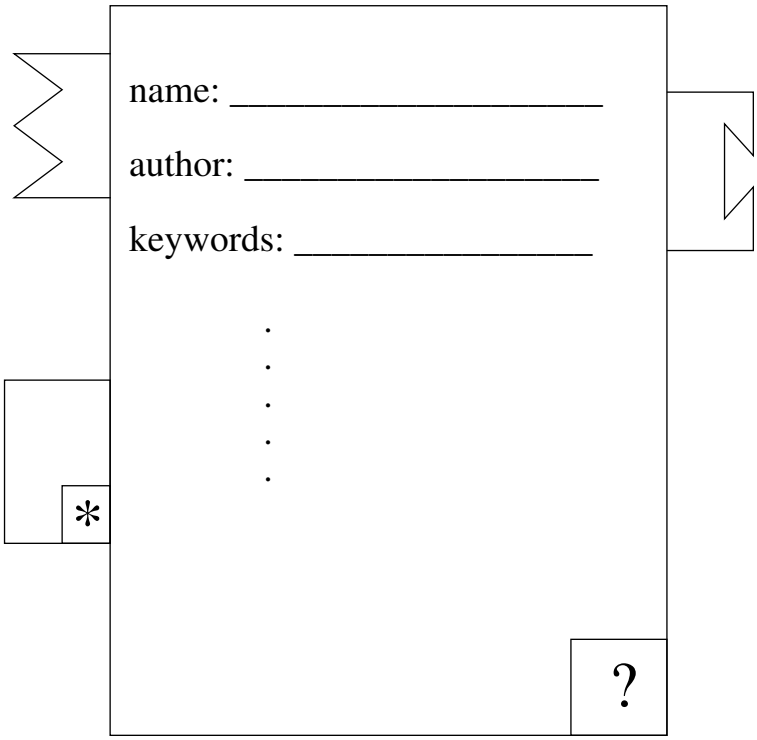
use visual type-matching for in-ports and out-ports

template ingredients can be

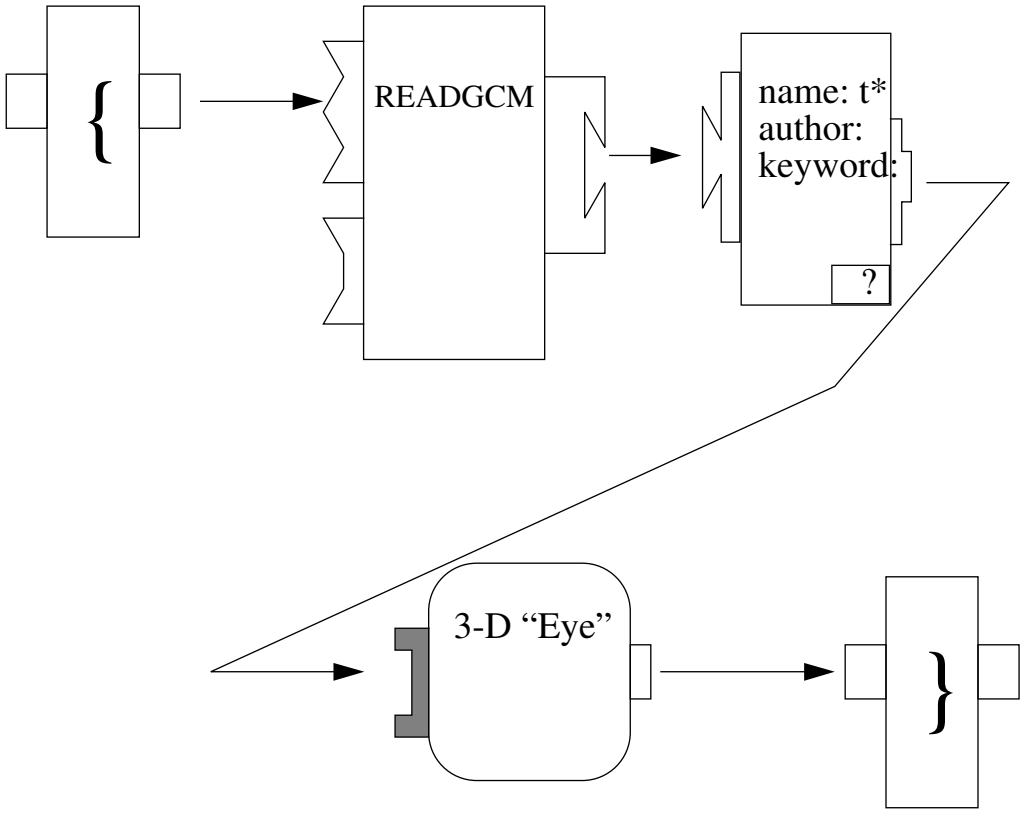
- single-matching (“?”)
- multi-matching (“*”)

similarly, in/out-ports on a template ingredient can be single or multi-matching

Example of a template ingredient:



Example of use of template ingredients:



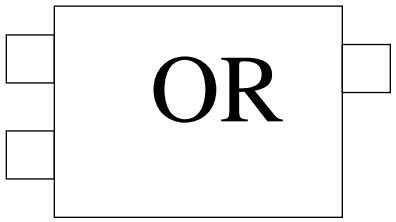
Operator ingredients:

normal ingredients that deal with data of the type `setof recipe`

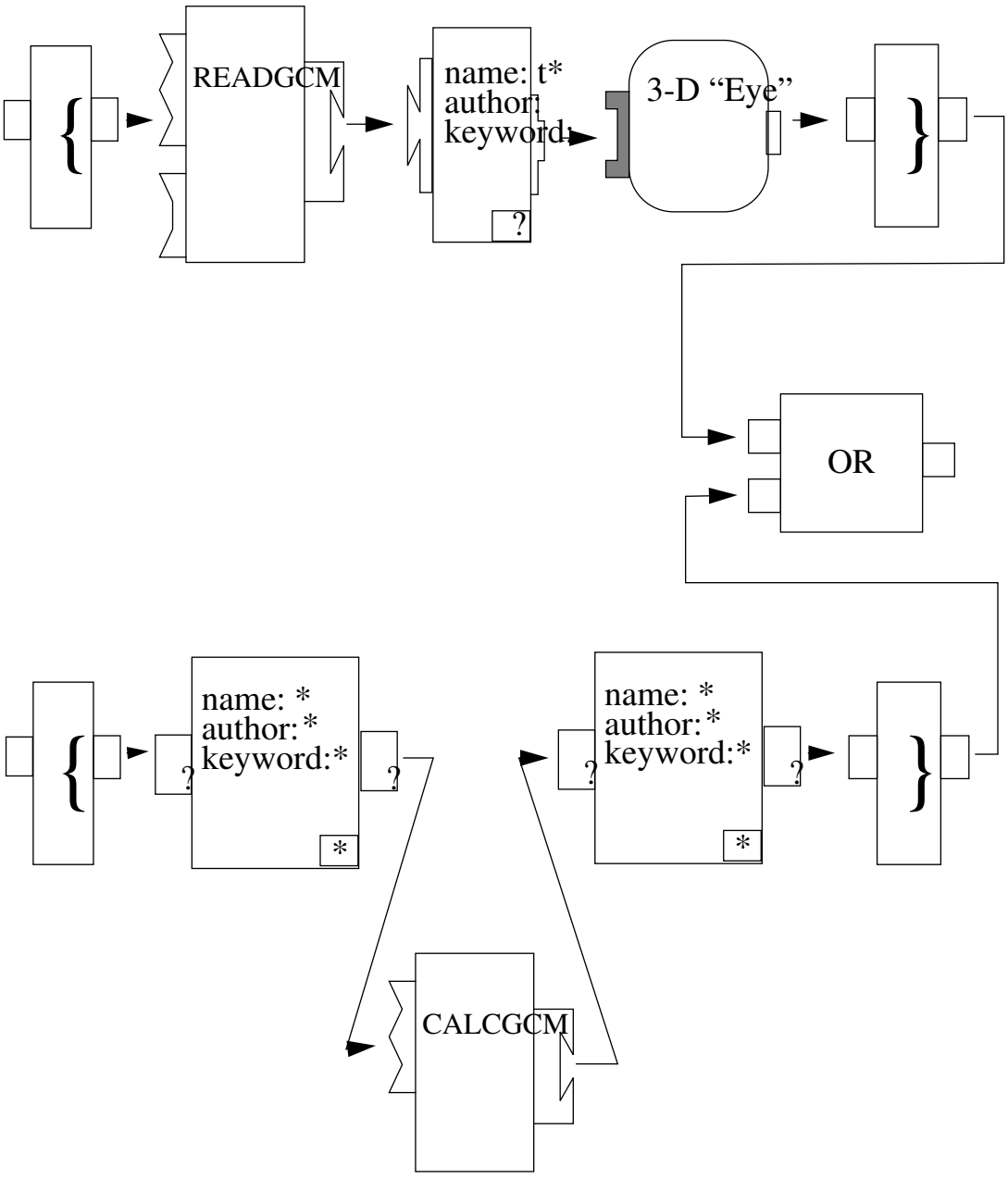
typically used in RASQL queries as relational operators

can be user-defined for extensibility

Examples:



Example of a complex RASQL query:



Implementation plans:

take advantage of and re-use graphical user interface of recipe editor

initially, use brute-force to execute queries

need graph-based access methods and efficient pattern matching algorithms

although graph matching problems are computationally difficult, RASQL queries should be practical since number of nodes in recipe graphs is small (10's)

Conclusions:

RASQL is a natural extension to the recipe definition language -- only a few query ingredients to learn about

easy to use and expressive

RASQL queries can be stored and manipulated as recipes. RASQL can even be used to write queries for RASQL queries.