AN ALGEBRAIC FRAMEWORK FOR MULTI-MODEL DATAMANAGEMENT

Jiaheng Lu
Department of Computer Science
University of Helsinki, Finland

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

OUTLINE

- Multi-Model Databases
- Categorical Algebra and Calculus
- Algebraic Transformation Rules
- UniBench: Multi-Model Database Benchmarking System
- Conclusion

BIG DATA

- Data come from different sources and have different formats

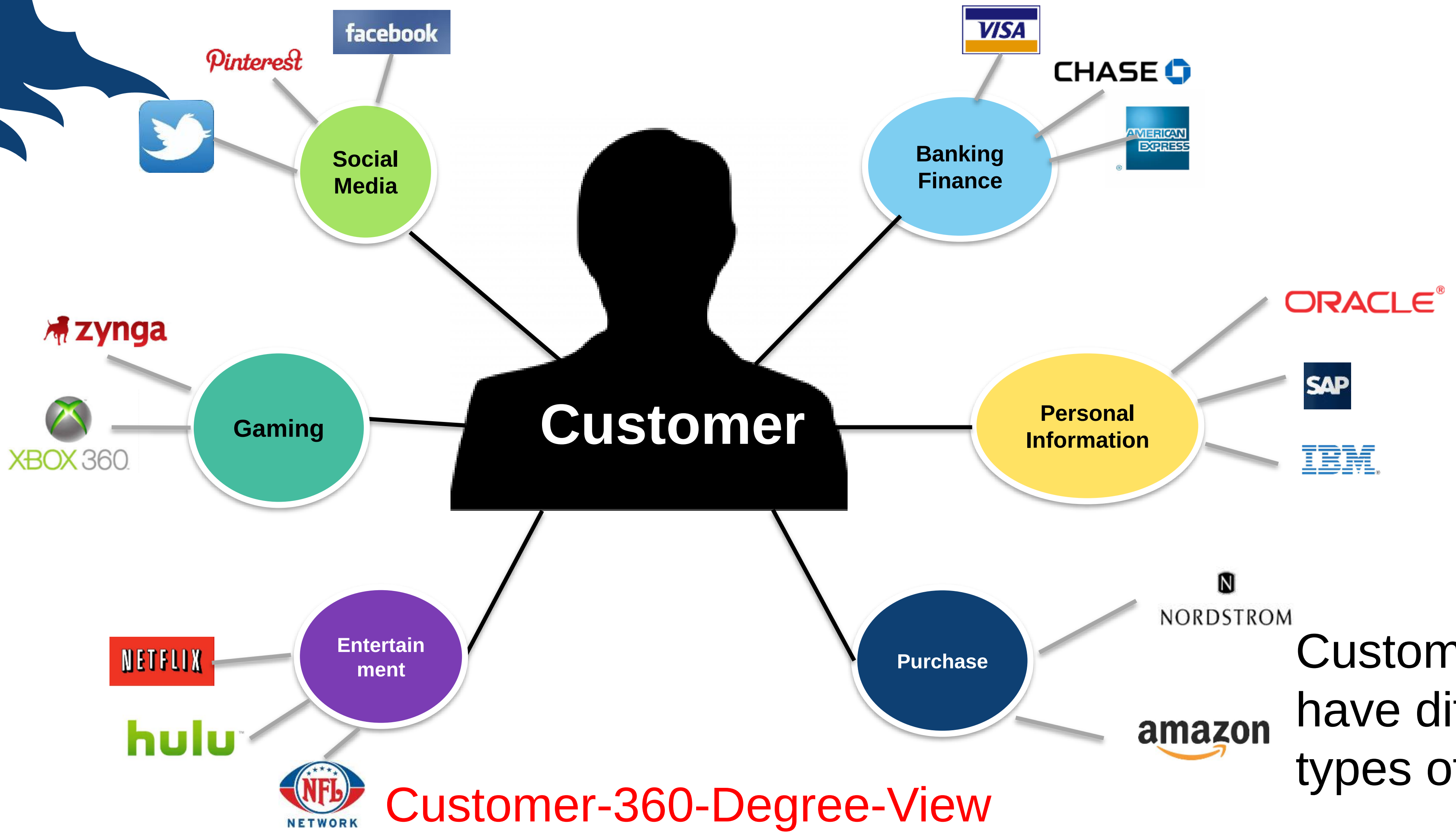
Smart phone

Camera

Social media

Acknowledgement: Icons created by Freepik - Flaticon

Variety challenge of big data

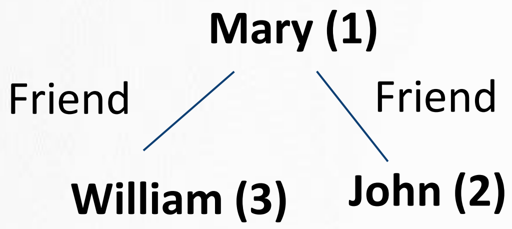


Customer-360-Degree-View

Customers have different types of data.

An example of different data and query in databases

Social network



Table

Customer_ID	Name	Credit_limits
1	Mary	5,000
2	John	3,000
3	William	2,000

Persons made the order:

```
"1" -- > "34e5e759"
"2"-- > "0c6df508"
```

Order information:

```
{"Order_no":"0c6df508",
 “Orderlines": [
   { "Product_no":"2724f”
     “Product_Name":“Toy",
     "Price":66 },
   { "Product_no":“3424g”,
     "Product_Name":“Book",
     "Price":40 } ]
}
```

An example of different data and query

Social network

Mary (1)

Friend — Friend

William (3) **John (2)**

Persons made the order:

"1" -- > "34e5e759"

"2"-- > "0c6df508"

Order information:

```
{"Order_no":"0c6df508",
 "Orderlines": [
   { "Product_no":"2724f"
     "Product_Name":"Toy",
     "Price":66 },
   { "Product_no":"3424g",
     "Product_Name":"Book",
     "Price":40 } ]
}
```

Table

Customer_ID	Name	Credit_limits
1	Mary	5,000
2	John	3,000
3	William	2,000

One application with different models of data

- Relational data: customer databases
- Graph data: social networks
- Hierarchical data: catalog, product
- Key-value data: orders by customers

How to integrate those heterogenous data to provide a unified service?

Multi-Model Databases System

- One unified database system for multi-model data

Multi-model database systems

What is DBMS?

- A Database Management System (DBMS) is software designed to efficiently manage data, with traditional systems storing data in the form of tables (RDBMS).

Student ID	First name	Last name	Department
001	John	Smith	Biology
002	Emily	Johnson	Physics
003	Michael	Brown	History
004	Sarah	Davis	English

Students Relational Table

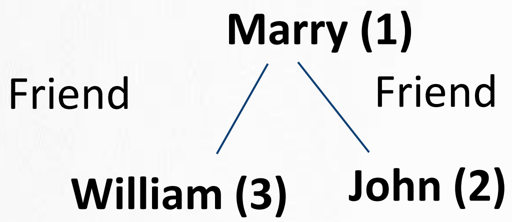
What is multi-model database management system

- A multi-model database management system (MMDBMS) is designed to support multiple data models against a single, integrated backend.

- Document, graph, relational and key-value models are examples of data models that may be supported by a multi-model database.

Four models of data

Graph:



Relation:

Customer_ID	Name	Credit_limits
1	Mary	5,000
2	John	3,000
3	William	2,000

Key-value:

"1" -- > "34e5e759"

"2"-- > "0c6df508"

Document:

```
{"Order_no":"0c6df508",
 “Orderlines": [
   { "Product_no":"2724f”
     “Product_Name":“Toy",
     "Price":66 },
   { "Product_no":“3424g”,
     "Product_Name":“Book",
     "Price":40 } ]
}
```

Advantages of MMDBMS over the traditional relational database

- Handling diverse data types
  - Handle various types of data, such as graph, relation, document and key-value data and more models
- Enhanced query capabilities
  - Support content-based search for multi-model data or spatial queries for geospatial data.
- Flexible schema
  - Greater flexibility in schema design and evolution. Relational DBMS has the fixed database schema definitions.

Multi-model databases products

ORACLE

OrientDB

ArangoDB

Azure Cosmos DB

mongoDB

APACHE DRILL

FOUNDATIONDB

MarkLogic

InterSystems CACHÉ

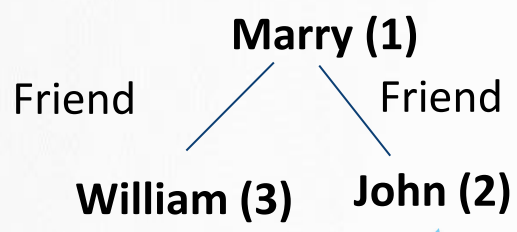
AsterixDB

...

• Supporting graph, document, key/value and object models.

Query: Return all products which are ordered by a friend of a customer whose credit_limit>4000

Answer: John is a friend of Mary (the credit_limit of Mary > 4000)




```
"1" -- > "34e5e759"
"2"-- > "0c6df508"
```

```
{"Order_no":"0c6df508",
 "Orderlines": [
   { "Product_no":"2724f"
     "Product_Name":"Toy",
     "Price":66 },
   { "Product_no":"3424g",
     "Product_Name":"Book",
     "Price":40 } ]
}
```

Customer_ID	Name	Credit_limits
1	Mary	5,000
2	John	3,000
3	William	2,000

Query language of OrientDB:

```
SELECT
expand(out("Knows").Orders.orderlines.Product_no)
FROM Customers
WHERE CreditLimit > 4000
```

Recommendation query:

Return all products which are ordered by any friend of a customer whose credit_limit>4000

Challenge: a new theory foundation

Research goal:

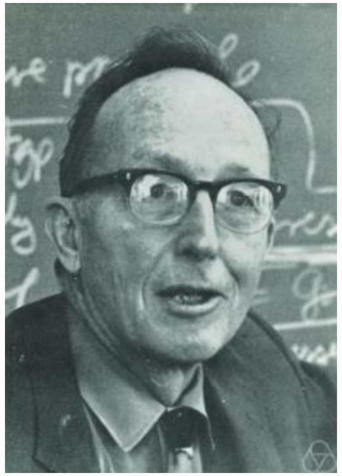
Call for a unified model and theory for multi-model data!

The theory of traditional relations is not adequate to mathematically describe modern database systems.

One possible theory foundation: Category Theory

- Introduced to mathematics world by Samuel Eilenberg and Sauders MacLane in 1944
- Developed for a unified language of topology and algebra

Samuel Eilenberg



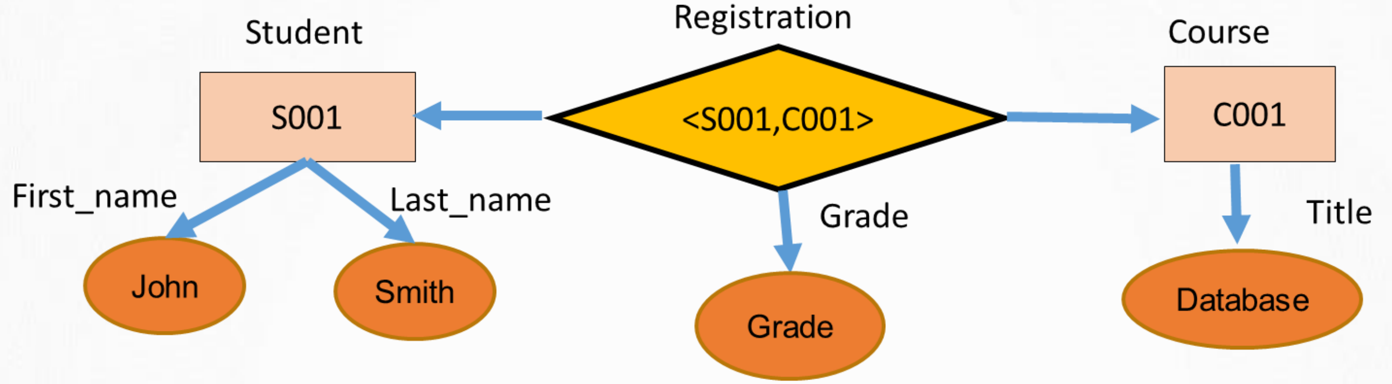
Sauders MacLane

Set Category

- Databases are set categories:
  - Objects are sets and morphisms are functions

- We assume that it is a thin Category (or Posetal Category)
  - Given a pair of objects X and Y in a category C, and any two morphisms f , g: X → Y , we say that C is a thin category if and only if the morphisms f and g are equal.

An example of Categorical Unification



Unified Category for three types of data

Student

Key	FN	LN
S001	John	Smith

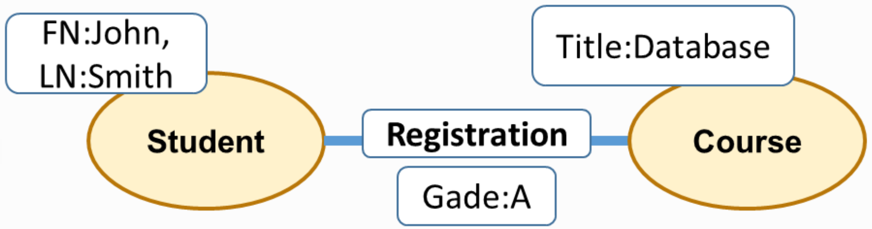
Course

Key	Title
C001	Database

Registration

SKey	CKey	Grade
S001	C001	A

Relation



Property graph

```
<Student key="S001">
  <First_name> John </First_name>
  <Last_name> Smith </Last_name>
</Student>
<Course key="C001">
  <Title> Database </Title>
</Course>
<Registration skey="S001"
ckey="C001">
  <Grade> A </Grade>
</Registration>
```

XML

Relational algebra and relational calculus

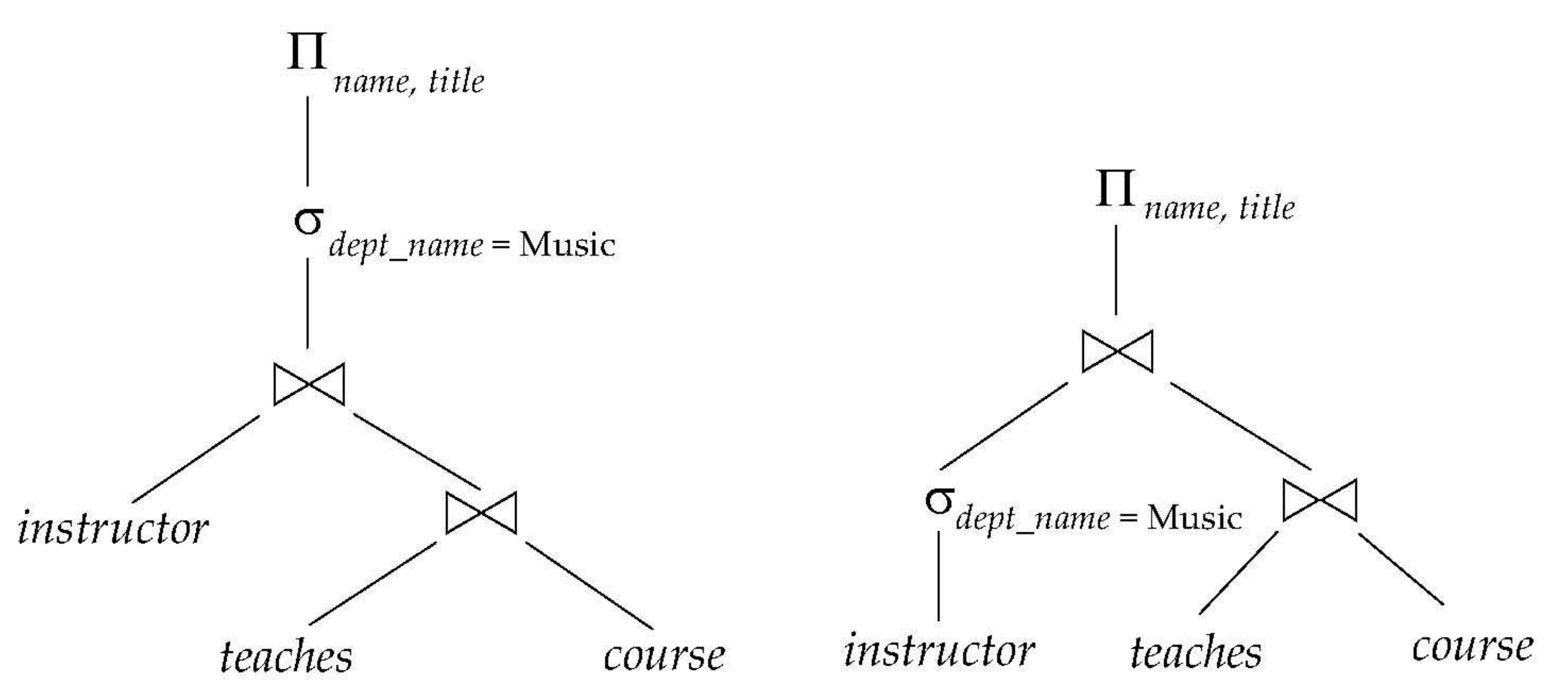
- In the field of relational databases, relational algebra and relational calculus are developed as two formal languages for query databases.

- Similarly, categorical algebra and categorical calculus are developed to query category databases.

Relational algebra

- Operators:
  - Selection: σ (sigma)
  - Projection: Π
  - Union: $\cup$
  - Intersection : $\cap$
  - Difference: -
  - Cartesian Product: $\times$
- Derived operators:
  - Joins (equi-join) $\bowtie$

- Examples of query trees:



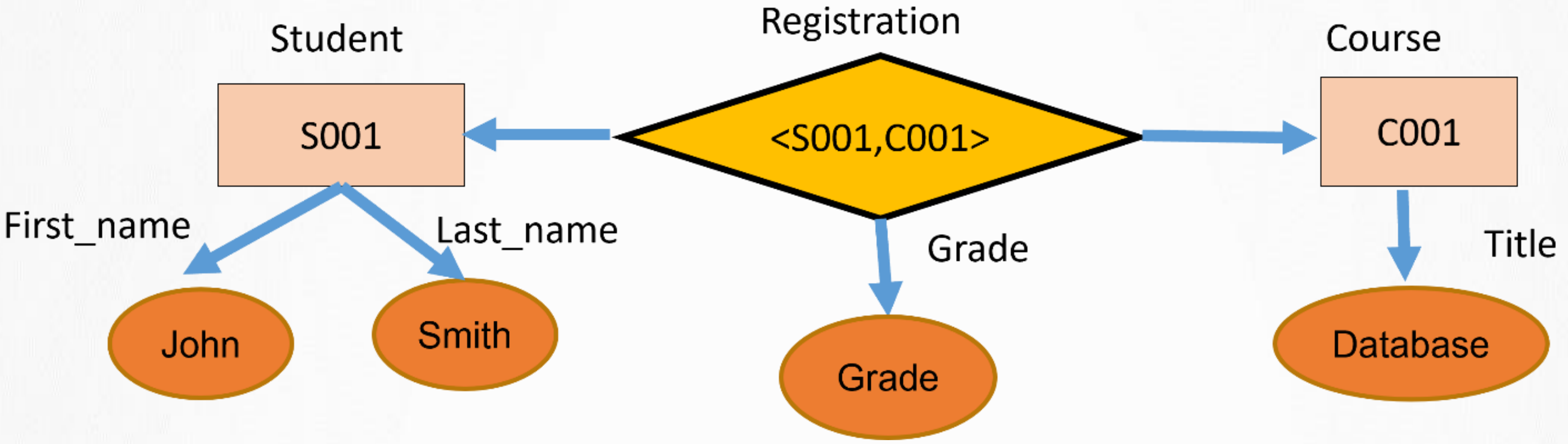
Categorical algebra

- Set operators:
  - Unary operator:
    - Map: f
    - Selection: σ
    - Projection: Π
  - Binary operator:
    - Division: $\div$
    - getParent(D_1,D_2) (tree data)
    - getAncestor(D_1,D_2) (tree data)
  - Tenary operator:
    - getReach(S,T,E) (graph data)
    - getNHop(S,T,E) (graph data)

Categorical algebra

- Category operators:
  - Sets and Functions to Category:
    - $Cat(S_1,\ldots,S_n, f_1 : S_{i1} \rightarrow S_{j1} ,\ldots, f_m : S_{im} \rightarrow S_{jm})$
    - This operator, called **Categorification**, constructs a category using a given set of objects and morphisms.
  - Category to Set
    - Limit which converts a category into a relational object (set).
    - $Lim(Cat(S_1,\ldots,S_n, f_1 : S_{i1} \rightarrow S_{j1} ,\ldots, f_m : S_{im} \rightarrow S_{jm}))$

Example of categorical algebra: Selection



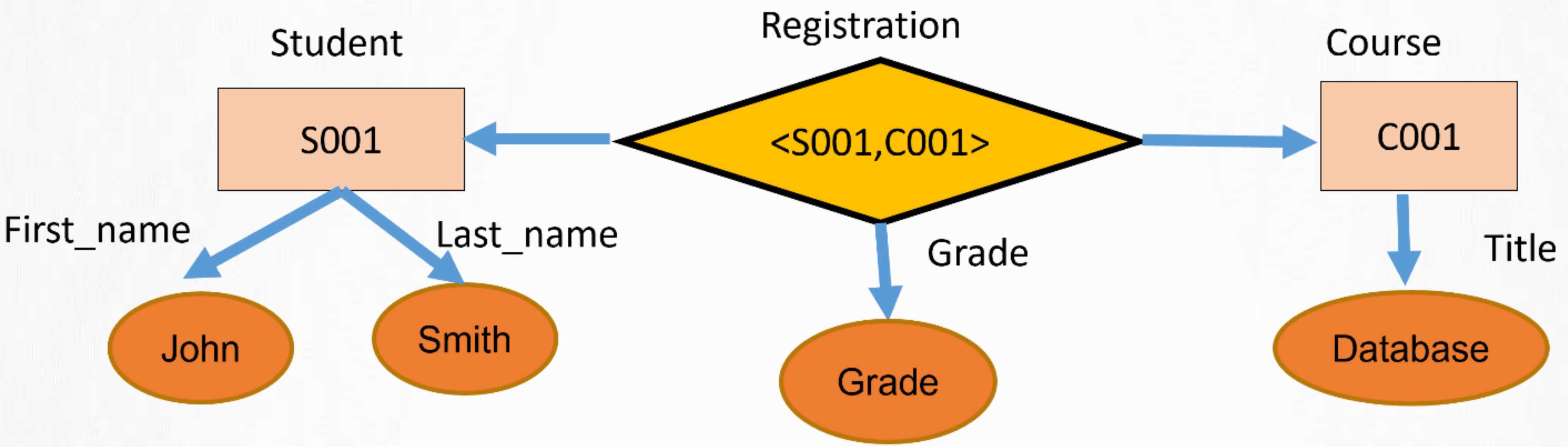
Query: Find all courses taken by ”Smith”

S1:= $\sigma_{\text{student} \cdot \text{Last\_name}=\text{“Smith”}}$ (Registration)

S2:= S1 ·Course ·Title

Return S2

Example of categorical algebra: Division



Query: Find the titles of courses taken by all students

S1:= Registration[Student] ÷ Student

S2:= S1 · Course · Title

Return S2

Two examples of query plan with categorical algebra

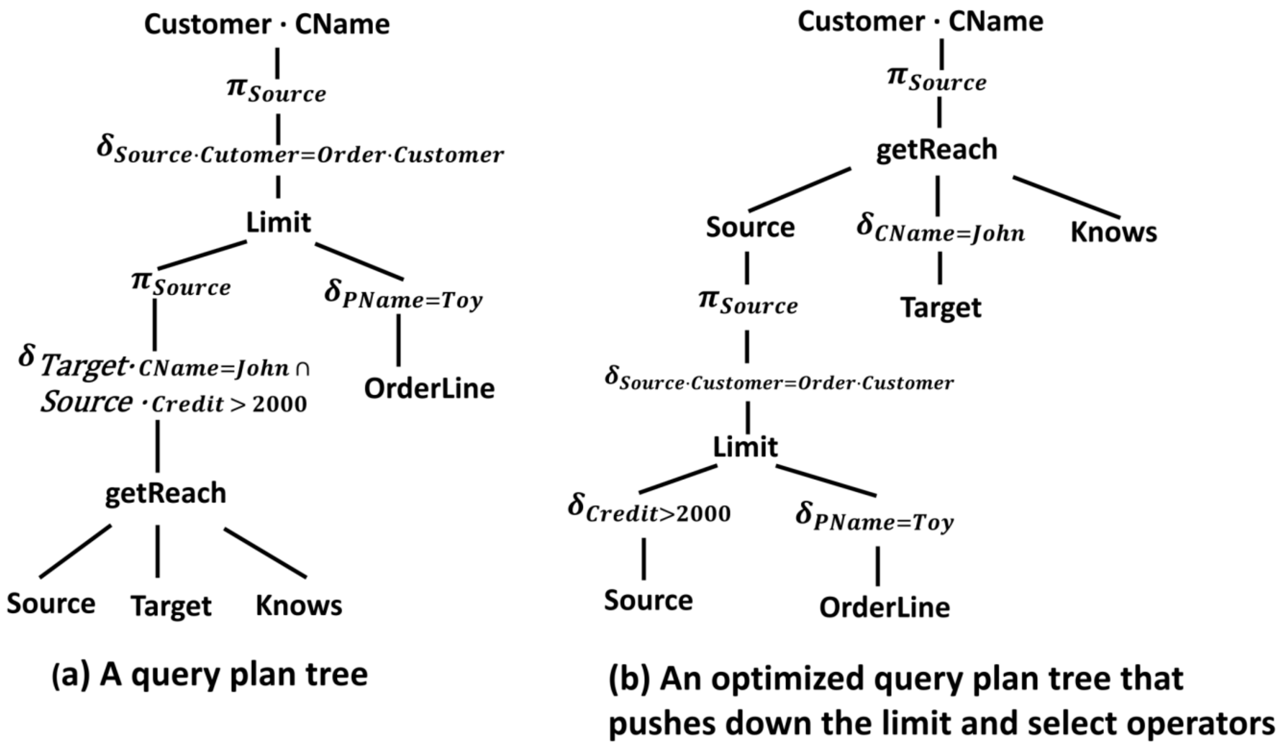


Figure 4: Two holistic query plans involving three types of data

Categorical calculus

- Categorical calculus, a **declarative** language for describing results in the category; Categorical algebra, a **procedural** language for listing operations in the category.
- The formulae of the Categorical Calculus

Formulae with range terms	Safe variables
$x_1 \in O_1$	x_1
$x_1 \in O_1 \wedge \neg(x_1 \in O_2)$	x_1
Formulae with function and range terms	**Safe variables**
$((f_1 : x_1 \to x_2) = f_2 \circ g_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
$(\pi_1 : (x_1, x_2) \to x_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
Formulae with predicate, range and function terms	**Data model**
$(x_1 \in S_1) \wedge (x_2 \in S_2) \wedge (x_1 \rightsquigarrow^E x_2) \wedge (x_1 \cdot \text{Name} = \text{"John"})$	Graph
$(x_1 \in D_1) \wedge (x_2 \in D_2) \wedge (x_1\ isAncestor\ x_2)$	Tree
Formulae with unsafe terms	**Unsafe variable**
$x_2 \in S_2, x_3 \in S_3, \exists x_1 (x_1 > x_3 \wedge x_2 = 6)$	x_1
$\forall x_1 \exists x_2 \in S_2 (x_1 > x_2)$	x_1
$(x_1 \in S_1) \vee f(x_1) = a_1$	x_1

Categorical calculus

- The formulae of the Categorical Calculus

Reachable predicate from node x_1 to x_2

Formulae with range terms	Safe variables
$x_1 \in O_1$	x_1
$x_1 \in O_1 \wedge \neg(x_1 \in O_2)$	x_1
Formulae with function and range terms	**Safe variables**
$((f_1 : x_1 \to x_2) = f_2 \circ g_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
$(\pi_1 : (x_1, x_2) \to x_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
Formulae with predicate, range and function terms	**Data model**
$(x_1 \in S_1) \wedge (x_2 \in S_2) \wedge (x_1 \rightsquigarrow^E x_2) \wedge (x_1 \cdot \text{Name} = \text{"John"})$	Graph
$(x_1 \in D_1) \wedge (x_2 \in D_2) \wedge (x_1\ isAncestor\ x_2)$	Tree
Formulae with unsafe terms	**Unsafe variable**
$x_2 \in S_2, x_3 \in S_3, \exists x_1 (x_1 > x_3 \wedge x_2 = 6)$	x_1
$\forall x_1 \exists x_2 \in S_2 (x_1 > x_2)$	x_1
$(x_1 \in S_1) \vee f(x_1) = a_1$	x_1

Categorical calculus

- The formulae of the Categorical Calculus

Ancestor predicate to determine the relationship between two nodes in trees.

Formulae with range terms	Safe variables
$x_1 \in O_1$	x_1
$x_1 \in O_1 \wedge \neg(x_1 \in O_2)$	x_1
Formulae with function and range terms	**Safe variables**
$((f_1 : x_1 \to x_2) = f_2 \circ g_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
$(\pi_1 : (x_1, x_2) \to x_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
Formulae with predicate, range and function terms	**Data model**
$(x_1 \in S_1) \wedge (x_2 \in S_2) \wedge (x_1 \rightsquigarrow^E x_2) \wedge (x_1 \cdot \text{Name} = \text{"John"})$	Graph
$(x_1 \in D_1) \wedge (x_2 \in D_2) \wedge (x_1\ \mathit{isAncestor}\ x_2)$	Tree
Formulae with unsafe terms	**Unsafe variable**
$x_2 \in S_2, x_3 \in S_3, \exists x_1 (x_1 > x_3 \wedge x_2 = 6)$	x_1
$\forall x_1 \exists x_2 \in S_2 (x_1 > x_2)$	x_1
$(x_1 \in S_1) \vee f(x_1) = a_1$	x_1

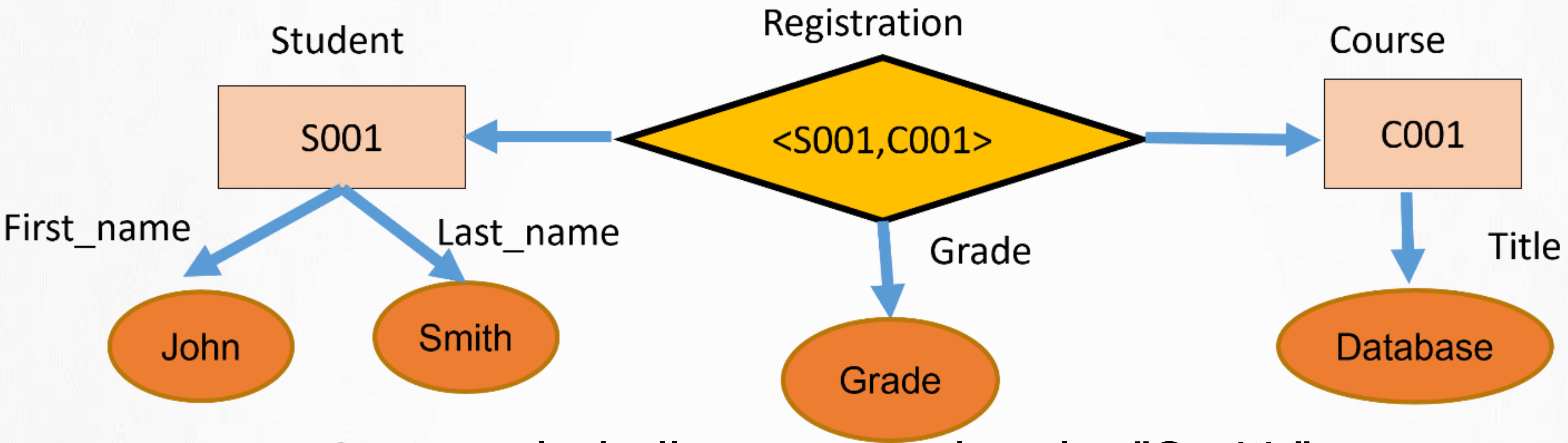
Categorical calculus

- The formulae of the Categorical Calculus

Formulae with range terms	Safe variables
$x_1 \in O_1$	x_1
$x_1 \in O_1 \wedge \neg(x_1 \in O_2)$	x_1
Formulae with function and range terms	**Safe variables**
$((f_1 : x_1 \to x_2) = f_2 \circ g_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
$(\pi_1 : (x_1, x_2) \to x_1) \wedge (x_1 \in S_1) \wedge (x_2 \in S_2)$	x_1, x_2
Formulae with predicate, range and function terms	**Data model**
$(x_1 \in S_1) \wedge (x_2 \in S_2) \wedge (x_1 \rightsquigarrow^E x_2) \wedge (x_1 \cdot \text{Name} = \text{"John"})$	Graph
$(x_1 \in D_1) \wedge (x_2 \in D_2) \wedge (x_1 \; isAncestor \; x_2)$	Tree
Formulae with unsafe terms	**Unsafe variable**
$x_2 \in S_2, x_3 \in S_3, \exists x_1 (x_1 > x_3 \wedge x_2 = 6)$	x_1
$\forall x_1 \exists x_2 \in S_2 (x_1 > x_2)$	x_1
$(x_1 \in S_1) \vee f(x_1) = a_1$	x_1

Unsafe variables refer to a variable that has possibly infinite number of values or is unbounded.

Categorical calculus and categorical algebra are equivalent (I)



Query: Find all courses taken by ”Smith”

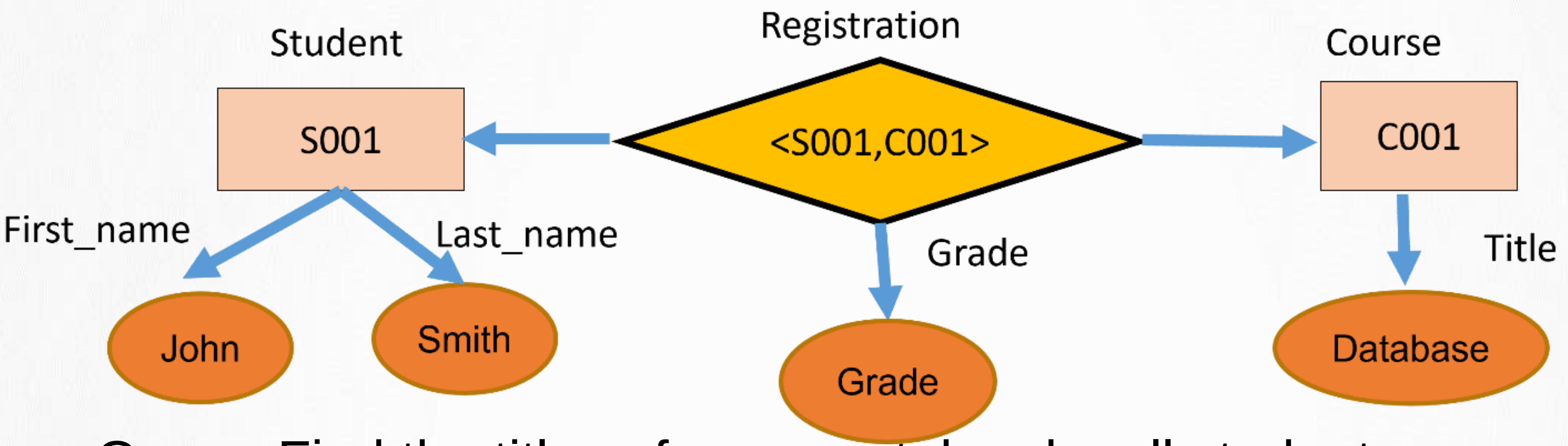
S1:= $\sigma_{\text{student} \cdot \text{Last\_name}=\text{“Smith”}}$ (Registration)

S2:= S1 ·Course ·Title

Equivalent calculus:

$\{\, x \mid x \in Title, \exists y \in Registration, y \cdot Student \cdot Last\_Name = \text{"Smith"} \wedge y \cdot \text{Course} \cdot \text{Title} = x \}$

Categorical calculus and categorical algebra are equivalent (II)



Query: Find the titles of courses taken by all students

S1:= Registration[Student] ÷ Student

S2:= S1 · Course · Title

Equivalent calculus:

$\{\ x \mid x \ \in Title,\ \forall y \in Student, \exists r \ \in Registration, r \cdot Student = y \wedge\ r \cdot$ Course · Title$=\ x\}$

OUTLINE

- Multi-model Databases
- Categorical Algebra and Calculus
- Algebraic Transformation Rules (Query optimization)
- Conclusion

Algebraic transformation rules (I)

- Rewrite the algebraic operators for query optimization.
- Limit and Projection:

$$\pi_{S_1}(Lim(Cat(S_1, S_2, f : S_1 \rightarrow S_2))) \equiv S_1$$

$$\pi_{S_2}(Lim(Cat(S_1, S_2, f : S_1 \rightarrow S_2))) \subseteq S_2$$

- Pushing σ to one or multiple objects in lim

$$\sigma_C(Lim(S_1, S_2)) \equiv Lim(\sigma_C(S_1), S_2)$$

$$\sigma_C(Lim(S_1, S_2)) \equiv Lim(\sigma_{C_1}(S_1), \sigma_{C_2}(S_2))$$

Algebraic transformation rules (II)

- Pushing σ to one or multiple objects in getReach:

$$getReach(\sigma_{C_1}(S), \sigma_{C_2}(T), E) \equiv \sigma_{C_1 \wedge C_2}(getReach(S, T, E))$$

- Commuting function mapping with the product operator.

$$(f \otimes g)(S_1 \times S_2) \equiv f(S_1) \times g(S_2)$$

- Commuting π with the Lim operation.

$$\pi_L(Lim(Cat(R_1, R_2, f_1 : R_1 \to R_2))) \equiv Lim(Cat(\pi_{L_1}(R_1), \pi_{L_2}(R_2), f_2 : \pi_{L_1}(R_1) \to \pi_{L_2}(R_2)))$$

Algebraic transformation rules (III)

- Commuting *g* with the Lim operation..
- The following diagram holds:

$$
\begin{array}{ccc}
S_1 & \xrightarrow{f_1} & S_2 \\
{\scriptstyle g_1}\downarrow & & \downarrow{\scriptstyle g_2} \\
S_1' & \xrightarrow[f_2]{} & S_2'
\end{array}
$$

then the two operators *g* and Lim can be commuted as follows:

$$g(Lim(Cat(S_1,S_2,f_1:S_1\to S_2)))\equiv Lim(Cat(g_1(S_1),g_2(S_2),f_2:g_1(S_1)\to g_2(S_2)))$$

An optimization query plan with algebraic operators transformation

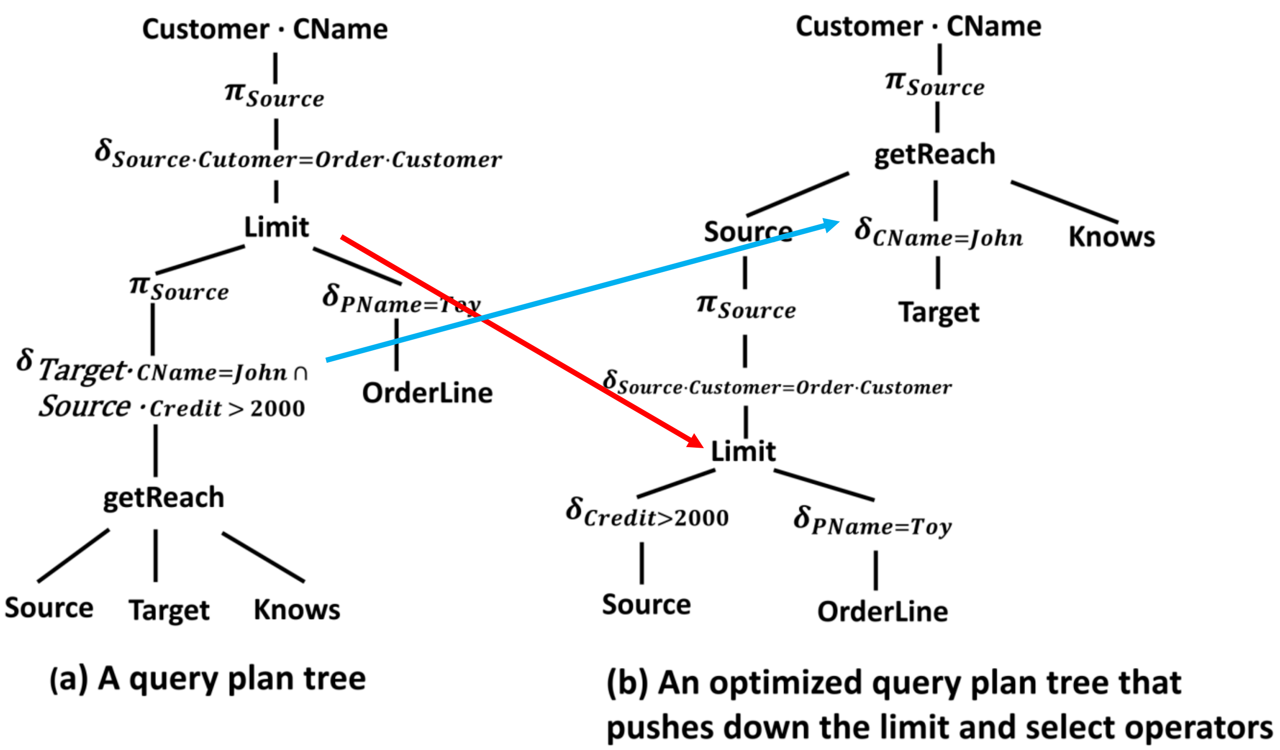


Figure 4: Two holistic query plans involving three types of data

$$\sigma_C(Lim(S_1, S_2)) \equiv Lim(\sigma_{C_1}(S_1), \sigma_{C_2}(S_2))$$

EXPRESSIBILITY POWER

- Categorical calculus and categorical algebra can express all of the following:
  - Relational calculus and algebra queries;
  - Graph pattern matching and graph reachability queries;
  - XML twig pattern queries.

RELATED WORK

- Previous works use category theory on relational databases, but our work focuses on multi-model databases.
  - Libkin and Wong (1997) showcase the connection between database operations and the categorical notion of a monad.
  - Schultz and Spivak (2016) introduce a categorical query language that serves as a data integration scripting language
  - …..
- There are existing algebra and calculus for relational data, graph data, and object-oriented data, but not multi-model data.

UniBench: Multi-model data bencharking system

- How to compare the performance of different multi-model database?

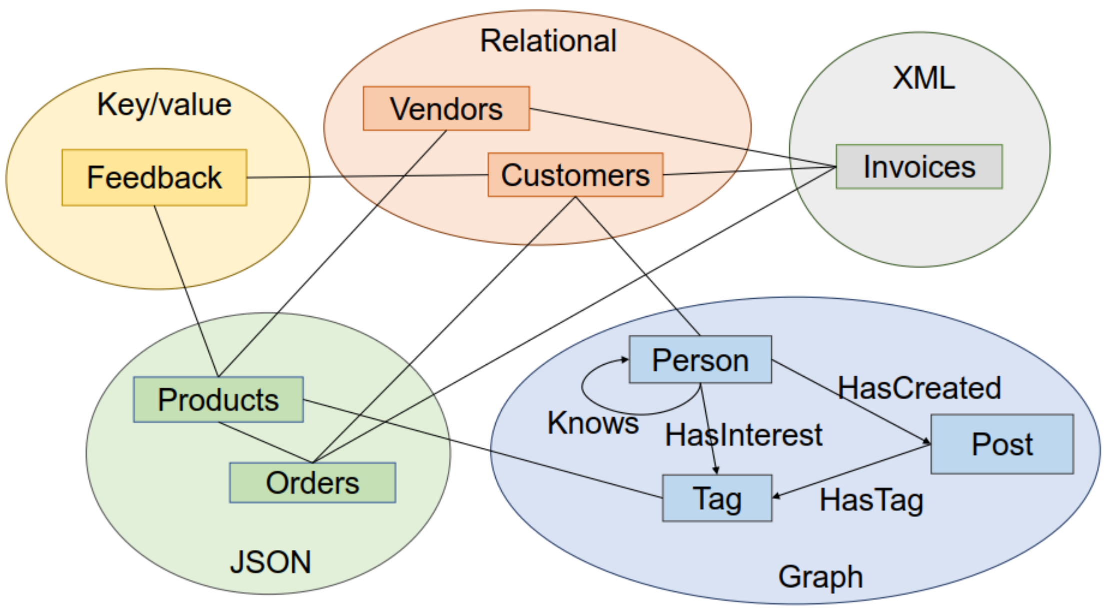


Fig. 1: Unibench Data Model

Table 2: Characteristics of Workload

Label	Business category	Technique dimension	Description
Q1	Individual	Perform point query on a customer's all multi-model data.	For a given *customer*, find her **profile**, **orders**, **feedback**, and **posts**.
Q2	Conversation	Join data from Relation, Graph, and JSON.	For a given *product*, find the **persons** who had bought it and posted on it.
Q3	Conversation	Join data from Relation, Graph, and Key-value, filter structured and unstructured data.	For a given *product*, find **persons** who have commented and posted on it, and detect negative **sentiments** from them.
Q4	Community	Aggregate and sort the JSON order, Perform the 3-hop graph traversal in the subgraph, return the intersection of two sets.	Find the top-2 **persons** who spend the highest amount of money in orders. Then for each person, traverse her knows-graph with 3-hop to find the friends, and finally return the common **friends** of these two persons.
Q5	Community	Join data from Relation, Graph, and Key-value with two predicates, recursive path query for Graph, embedded array operation for JSON, and composited-key lookup for Key-value.	Given a start *customer* and a product *category*, find **persons** who are this customer's friends within 3-hop friendships in knows-graph, and they have bought products in the given category. Finally, return **feedback** with the 5-rating review of those bought products.
Q6	Community	Perform the shortest path calculations between two nodes, find the correlated JSON orders of nodes in the path, aggregation on returned JSON orders.	Given *customer 1* and *customer 2*, find persons in the **shortest path** between them in the subgraph, and return the TOP 3 **best sellers** from all these persons' purchases.
Q7	Commerce	Join data from Relation, JSON and Key-value, compare the aggregation results between two periods, identify the reviews with negative sentiment.	For the *products* of a given *vendor* with declining sales compare to the former quarter, analyze the **reviews** for these items to see if there are any negative sentiments.
Q8	Commerce	Perform the embedded array filtering and aggregation on JSON order, aggregate the correlated graph data for each records.	For all the *products* of a given *category* during a given year, compute its **total sales amount**, and measure its **popularity** in the social media.
Q9	Commerce	Perform the embedded array filtering, aggregation, and sorting on JSON order, then find the correlated graph data.	Find top-3 **companies** who have the largest amount of sales at one *country*, for each company, compare the number of the male and female customers, and return the most recent **posts** of them.
Q10	Commerce	Perform the aggregation and sort on graph data, then find the correlated Key-value and JSON data.	Find the top-10 most active **persons** by aggregating the *posts* during the last year, then calculate their **RFM (Recency, Frequency, Monetary) value** in the same period, and return their recent **reviews** and tags of **interest**
T1	New Order Transaction	Check the ACID properties and evaluate the efficiency on read-heavy multi-model transaction that involves JSON and XML.	(i) create and insert the **order**, (ii) update the quantity of involved **products**, (iii) insert the **invoice**.
T2	Payment Transaction	Check the ACID properties and evaluate the efficiency on write-heavy multi-model transaction that involves Relation, JSON and XML.	(i) retrieve the unpaid **order**, (ii) update the balance of the **seller** and **buyer**, (iii) update the **order** status to paid, (iv) update the related **invoice**.

Workload:

4 business categories,

10 queries,

2 transactions

EXPERIMENTAL RESULTS

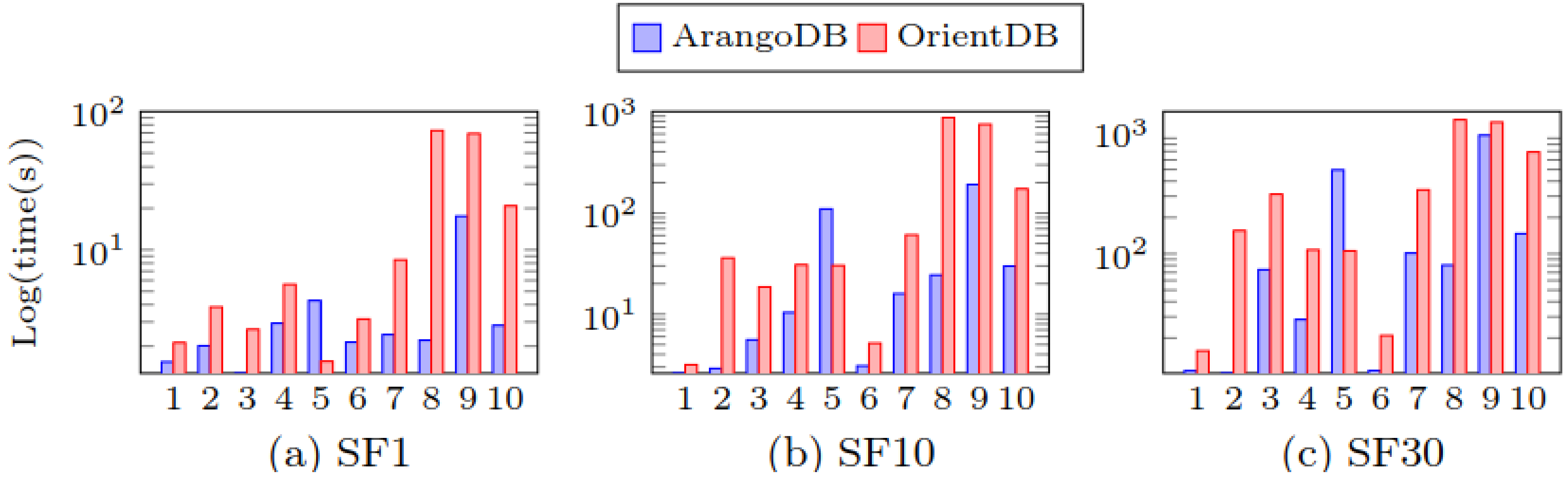


Fig. 5: Processing time on a logarithmic scale for queries, x-axis labels are query ids, i.e., Q1 to Q10.

This open-source benching mark system has been widely used in DataBricks, MongoDB and Amazon.

MAIN CONTRIBUTION AND CONCLUSION

- Define categorical algebra and calculus for multi-model database
- Develop the algebraic transformation rules for query optimization
- Develop a benchmark system for multi-model database

Applied category theory (ACT) here can contribute to **practical query processing and optimization** of multi-model databases.

REFERENCES

- Jeremy Gibbons, Fritz Henglein, Ralf Hinze & Nicolas Wu (2018): Relational algebra by way of adjunctions. Proc. ACM Program. Lang. 2(ICFP), pp. 86:1–86:28.
- Leonid Libkin & Limsoon Wong (1997): Query Languages for Bags and Aggregate Functions. J. Comput. Syst. Sci. 55(2), pp. 241–272.
- Patrick Schultz, David I. Spivak, Christina Vasilakopoulou & Ryan Wisnesky (2016): Algebraic Databases. arXiv:1602.03501.
- Allen Van Gelder & Rodney W. Topor (1991): Safety and translation of relational calculus. ACM Trans Database Syst. 16(2), p. 235–278, doi:10.1145/114325.103712. Available at https://doi.org/10.1145/114325.103712.
- Jiaheng Lu, Irena Holubová: Multi-model Databases: A New Journey to Handle the Variety of Data. ACM Comput. Surv. 52(3): 55:1-55:38 (2019)
- Jiaheng Lu: A Categorical Unification for Multi-Model Data: Part I Categorical Model and Normal Forms. CoRR abs/2502.19131 (2025)

THANK YOU!

- Answer questions in Poll about multi-model database.