FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Martin Svoboda, Pavel Koupil, Irena Holubová

Categorical Modeling of Multi-Model Data: One Model to Rule Them All

10th International Conference on Model and Data Engineering
Online Conference (Tallinn, Estonia)
June 21-23, 2021

Motivation

- The *variety* feature of Big Data
- The *level of support* of multiple data models in MMDBMS *varies greatly*
  - No unified approaches exist
  - Necessity of multiple *model-specific query constructs*
  - There is *no* solid *formal background*
- We need a representation that would allow us to
  - Capture all the existing data models, preferably in a *unified way*
  - *Query* across multiple *interconnected*, possibly *overlapping* models
  - Perform correct and complete *evolution management*
  - Enable *data migration*
  - Permit *integration* of new data models

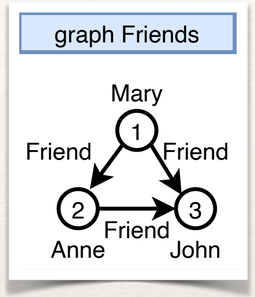


table Customer

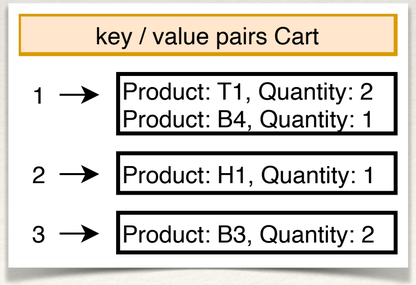
CustomerId	FirstName	Address	Credit
1	Mary	...	3000
2	Anne	...	2000
3	John	...	5000

collection Order

```
{ OrderId : 220,
  Paid: true,
  Items: [
  { ProductId: T1, Name: toy,
    Price: 200, ItemQuantity: 2},
  { ProductId: B4, Name: book,
    Price: 150, ItemQuantity: 1 } ] }
```

column family Orders

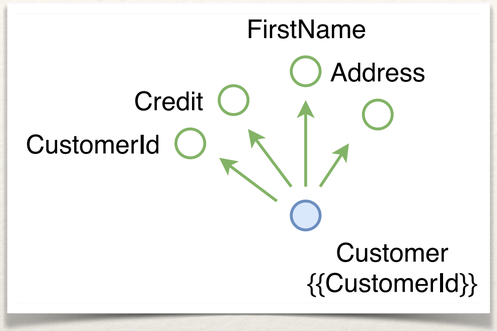
CustomerId	Orders
1	[220, 230, 270, ...]
2	[10, 217]
3	[94, 137, 214, 370]



Relational Schema as a Multigraph

table Customer

CustomerId	FirstName	Address	Credit
1	Mary	...	3000
2	Anne	...	2000
3	John	...	5000

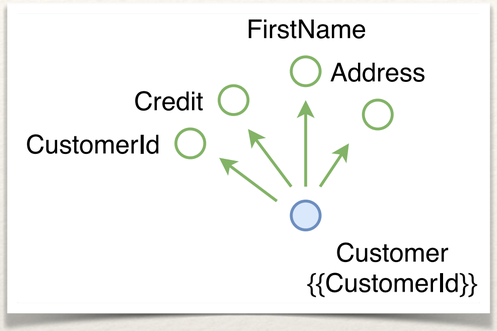


- *Relational schema* is a named set of columns
- Modeling:
  - *Table* modeled as a *named node* with identifier description
  - *Column* modeled as a *named node*
  - Table and its column representation is *connected by an arrow*
  - Two tables are *connected by arrows*

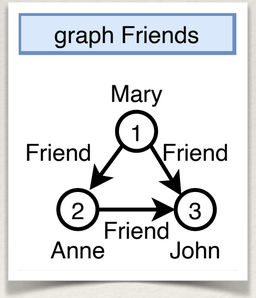
Graph Schema as a Multigraph

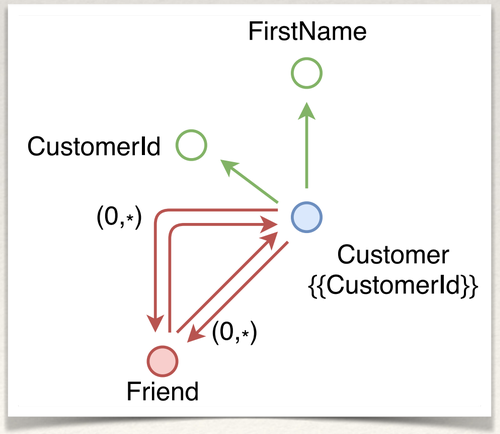
table Customer

CustomerId	FirstName	Address	Credit
1	Mary	...	3000
2	Anne	...	2000
3	John	...	5000



- *Graph* is a set of vertices possibly interlinked by a set of edges
- *Named vertex* is associated with an *identifier* and *set of properties*
- The same applies for a *named edge*



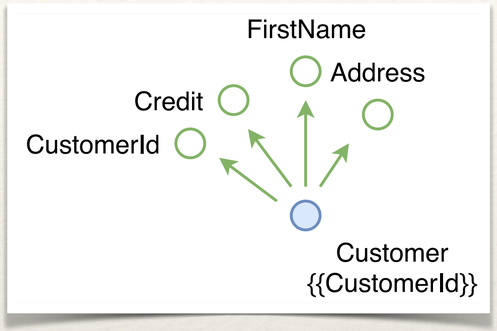


- Modeling:
  - *Set of vertices* of the same type represented as a *named node*
  - Corresponding *properties* modeled as a *named node*
  - *Set of edges* of the same type modeled as a *named node*
  - Property and particular vertex/edge is *connected by an arrow*
  - If applicable, *vertex and edge* are *connected by arrows*

Document Schema as a Multigraph

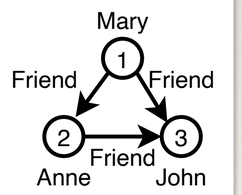
table Customer

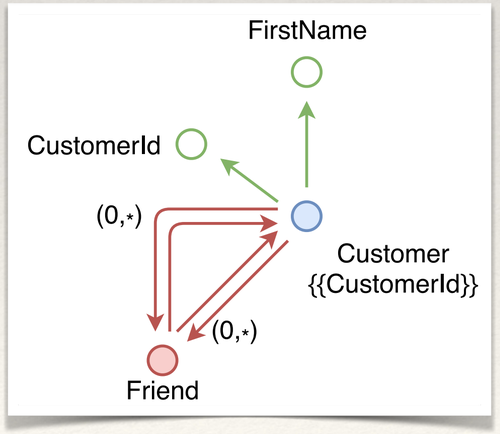
CustomerId	FirstName	Address	Credit
1	Mary	...	3000
2	Anne	...	2000
3	John	...	5000



- *Document* is a set of properties, possibly structured ones
- Modeling:
  - *Document* is represented as a *named node*
  - Each *property*, simple or complex, is represented as a *named node*
  - *Hierarchy* between properties (parent, child) represented as an *arrow*

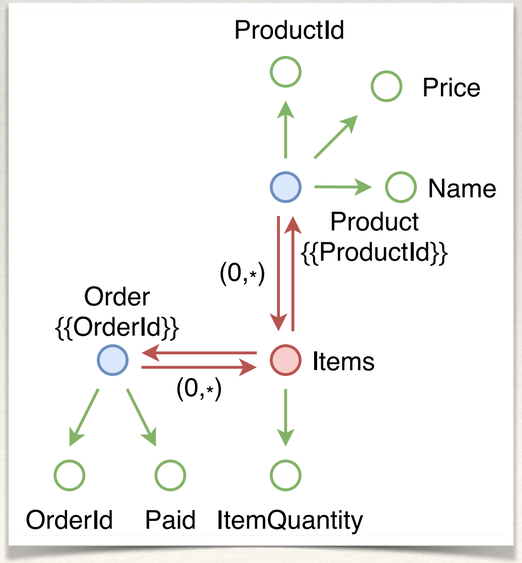
graph Friends





collection Order

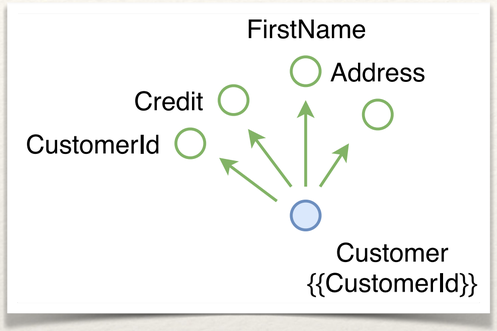
```
{ OrderId : 220,
  Paid: true,
  Items: [
  { ProductId: T1, Name: toy,
    Price: 200, ItemQuantity: 2},
  { ProductId: B4, Name: book,
    Price: 150, ItemQuantity: 1 } ] }
```



Columnar Schema as a Multigraph

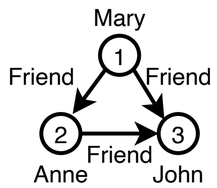
table Customer

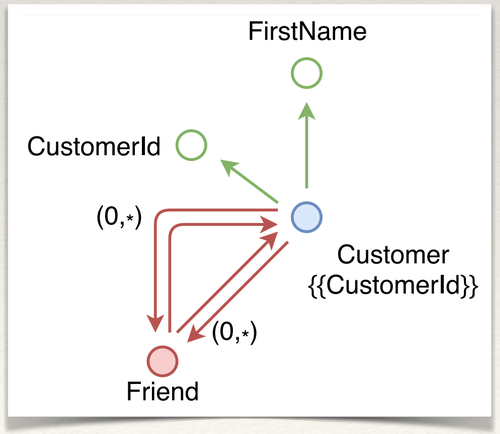
CustomerId	FirstName	Address	Credit
1	Mary	...	3000
2	Anne	...	2000
3	John	...	5000



- *Column family* is just a special kind of a document...

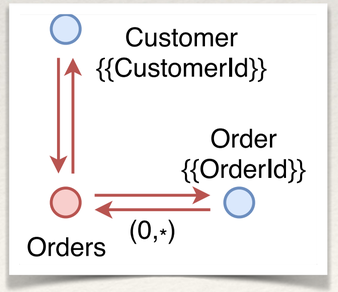
graph Friends





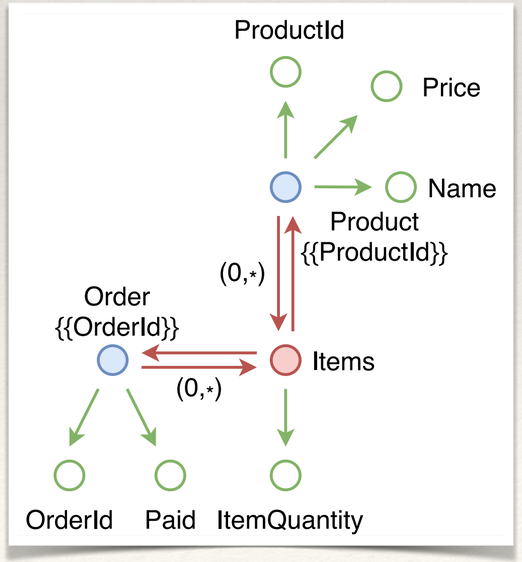
column family Orders

CustomerId	Orders
1	[220, 230, 270, ...]
CustomerId	Orders
2	[10, 217]
CustomerId	Orders
3	[94, 137, 214, 370]



collection Order

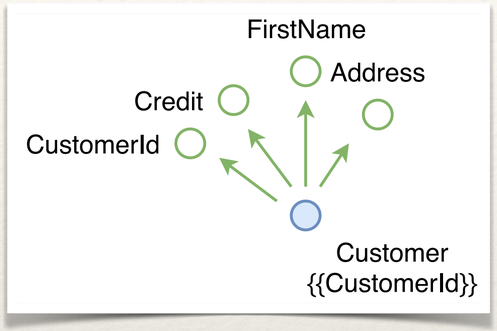
```
{ OrderId : 220,
  Paid: true,
  Items: [
  { ProductId: T1, Name: toy,
    Price: 200, ItemQuantity: 2},
  { ProductId: B4, Name: book,
    Price: 150, ItemQuantity: 1 } ] }
```



Key/Value Schema as a Multigraph

table Customer

CustomerId	FirstName	Address	Credit
1	Mary	...	3000
2	Anne	...	2000
3	John	...	5000



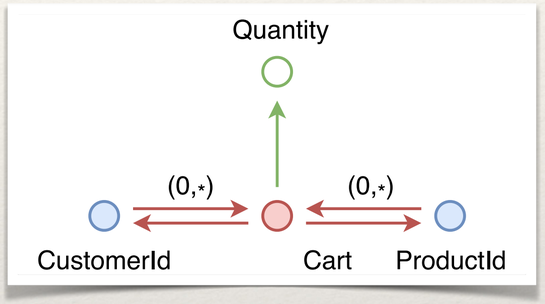
- The same applies for *key/value*

key / value pairs Cart

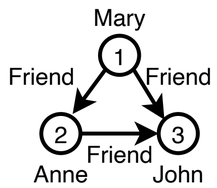
1 → Product: T1, Quantity: 2
Product: B4, Quantity: 1

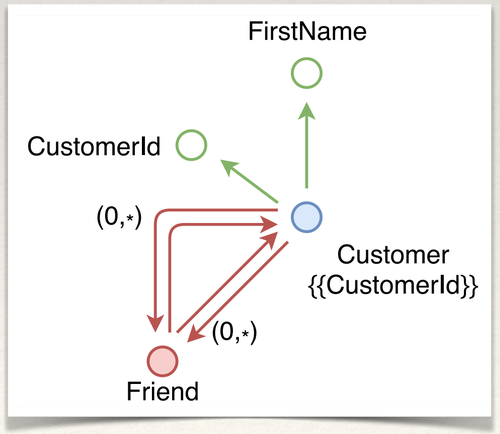
2 → Product: H1, Quantity: 1

3 → Product: B3, Quantity: 2



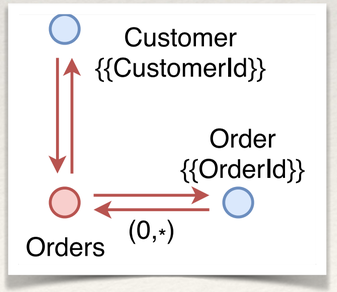
graph Friends





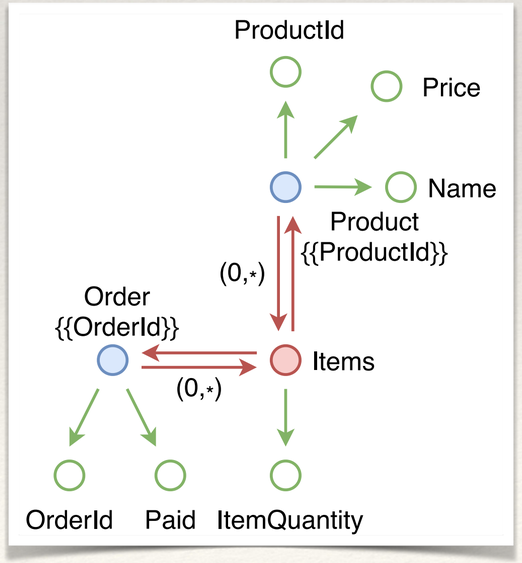
column family Orders

CustomerId	Orders
1	[220, 230, 270, ...]
CustomerId	Orders
2	[10, 217]
CustomerId	Orders
3	[94, 137, 214, 370]



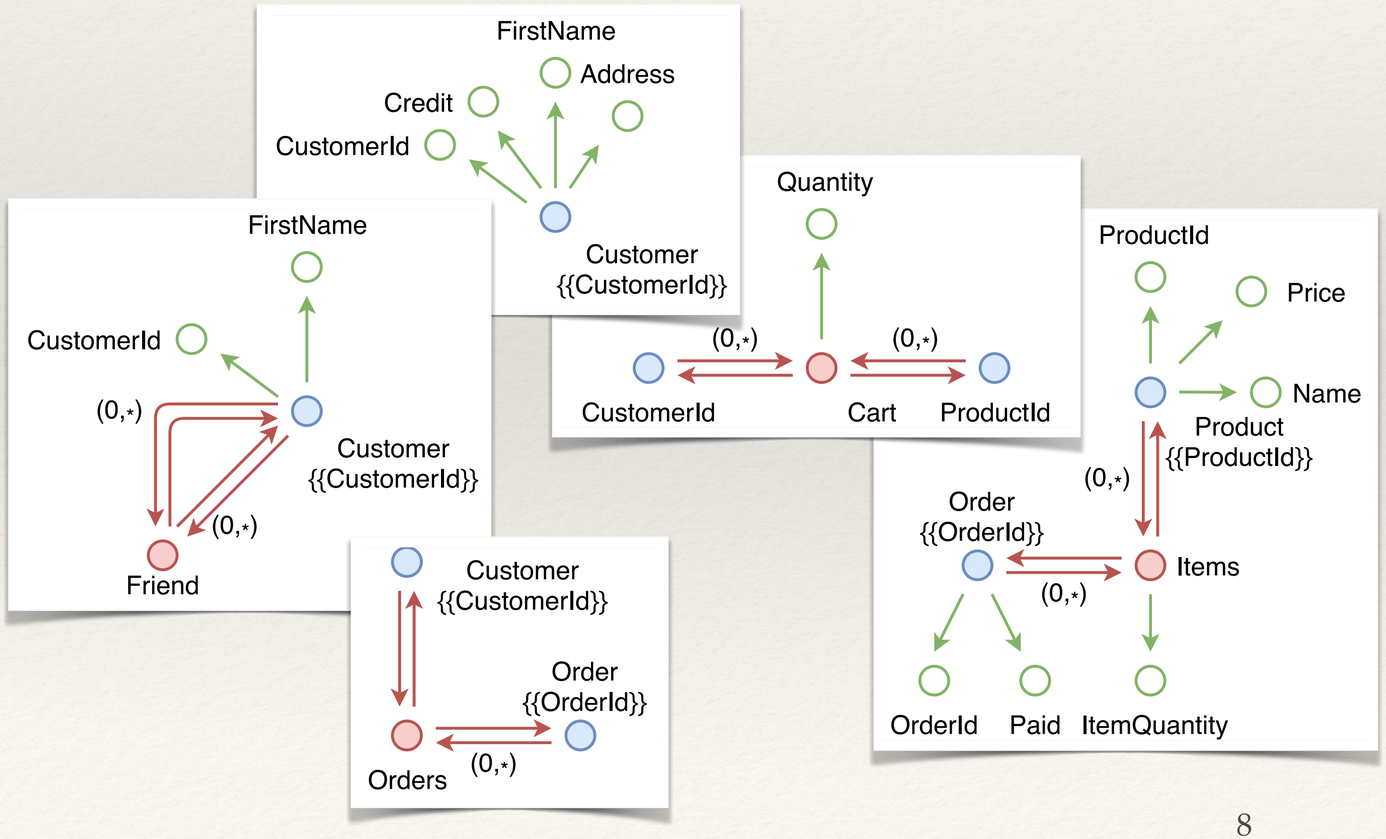
collection Order

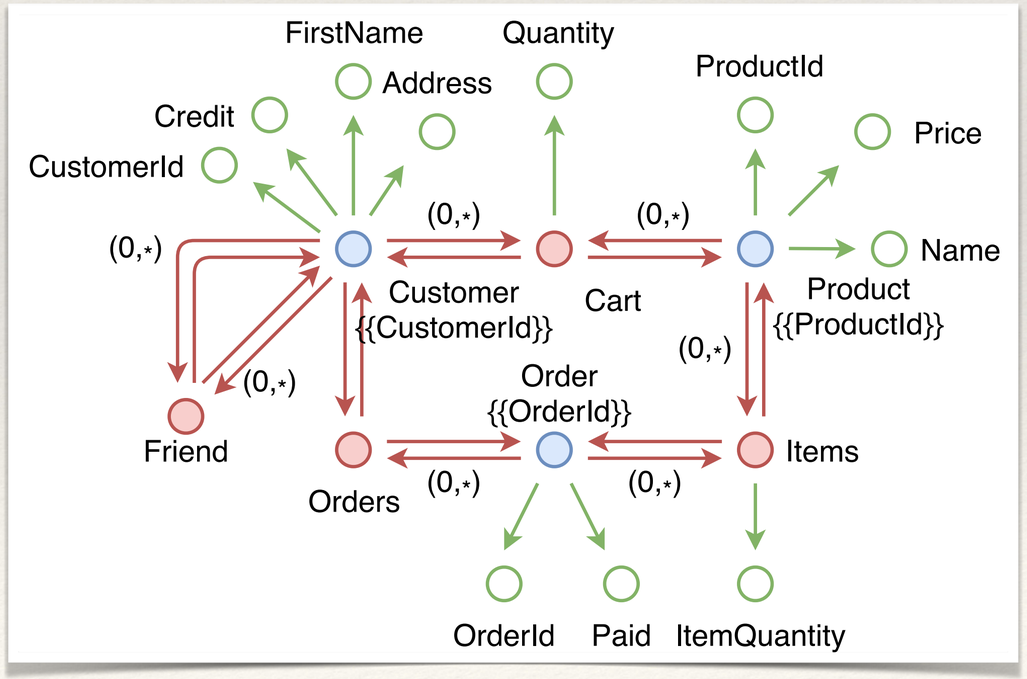
```
{ OrderId : 220,
  Paid: true,
  Items: [
  { ProductId: T1, Name: toy,
    Price: 200, ItemQuantity: 2},
  { ProductId: B4, Name: book,
    Price: 150, ItemQuantity: 1 } ] }
```



Multi-Model Schema as a Multigraph

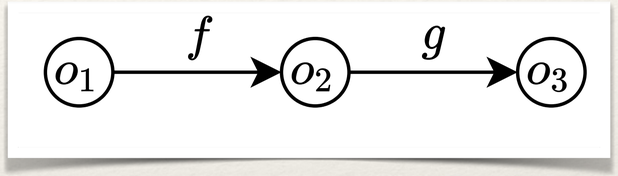
- Everything is put together to get the resulting multi-model schema of the involved interconnected and possibly overlapping particular models

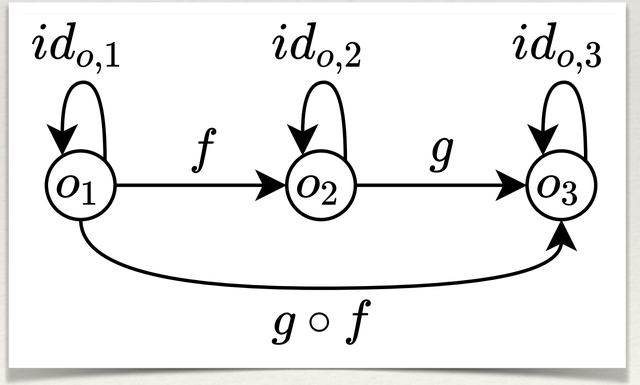




(Small) Category

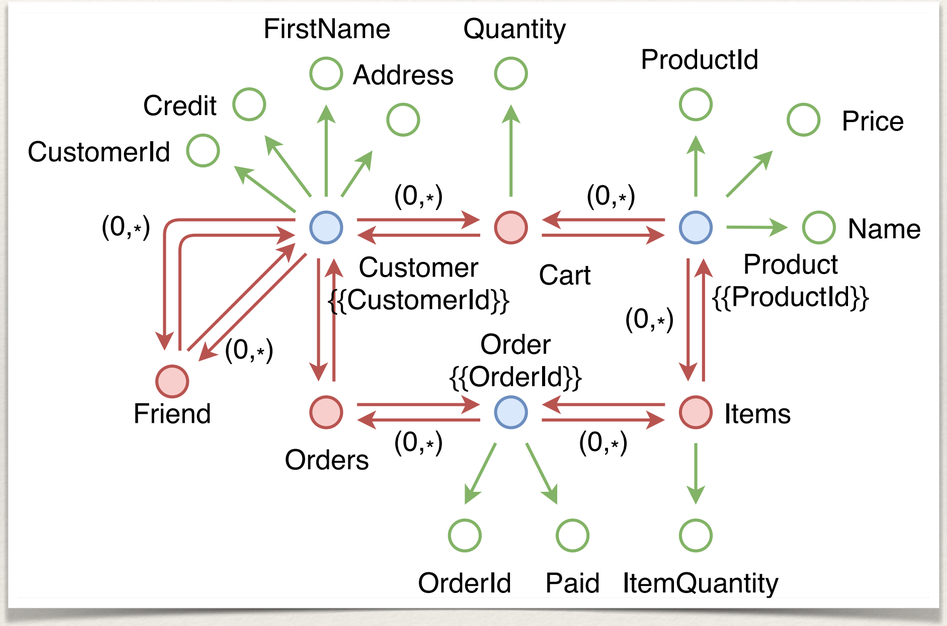
- Special kind of a *multigraph* $C = (\mathcal{O}, \mathcal{M}, \circ)$ conforming to identity and composition laws
- *Object* $o_i \in \mathcal{O}$ is depicted as a graph node
- Each *morphism* is an arrow $f : o_i \to o_j$ between $o_i, o_j \in \mathcal{O}$
- *Composition* $\circ$ is associative
  - For any morphisms $f, g, h \in \mathcal{M}$ such that $f : o_1 \to o_2$, $g : o_2 \to o_3$ and $h : o_3 \to o_4$ it holds that $h \circ (g \circ f) = (h \circ g) \circ f$
- When $f, g \in \mathcal{M}$ such that $f : o_1 \to o_2$ and $g : o_2 \to o_3$ also $g \circ f \in \mathcal{M}$ (composition law)
- For every object $o_i \in \mathcal{O}$ there exists $id_{o,i} \in \mathcal{M}$ such that $f_{i,j} \circ id_{o,i} = f_{i,j} = id_{o,j} \circ f_{i,j}$ (identity law)





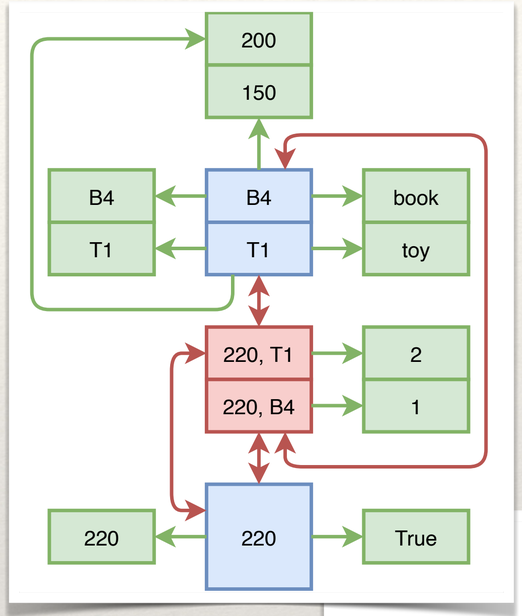
Schema Category $\mathcal{S}$

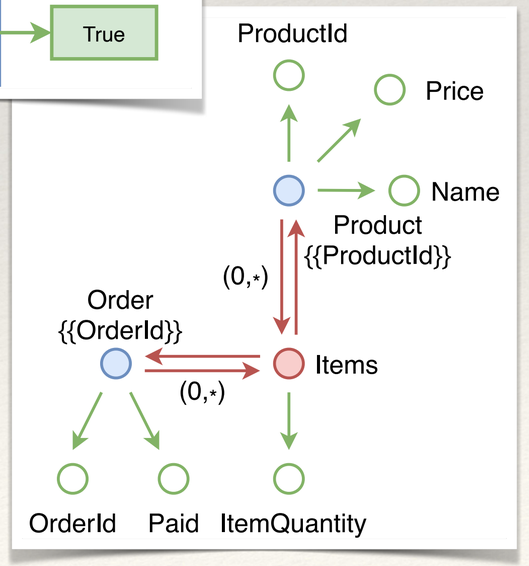
- $\mathcal{S} = (\mathcal{O}_{\mathcal{S}}, \mathcal{M}_{\mathcal{S}}, \circ_{\mathcal{S}})$ grasps the *structure* of the data and basic *integrity constraints*
  - *Objects* represent individual *entity types*, *relationship types* and *attributes*
    - Internally modeled as a pair (*superid*, *ids*)
      - *superid* is a set of attributes using which the respective instances can be uniquely identified, i.e., *superidentifier*
      - *ids* is a set of *individual identifiers*, allowing simple, composed and overlapping
  - *Morphism* is a pair (*min*, *max*) modeling the concept of *cardinalities*



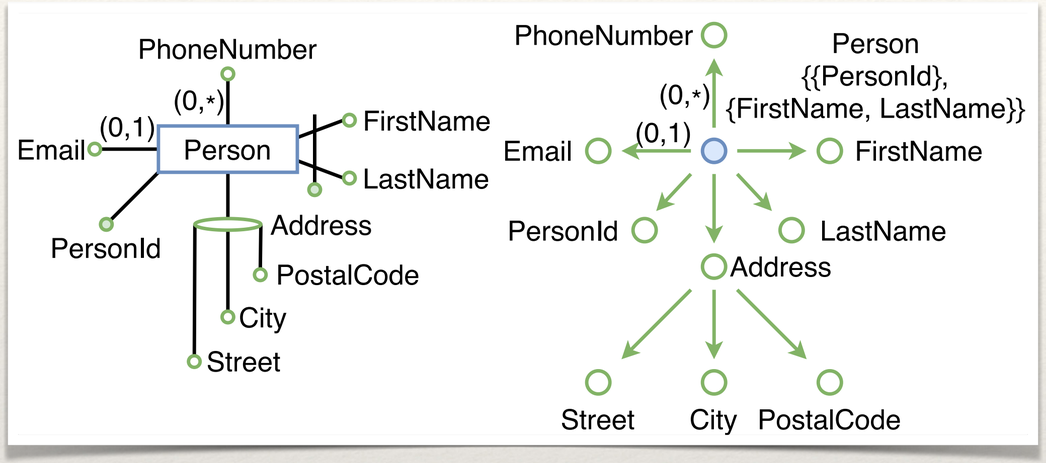
Instance Category $\mathcal{I}$

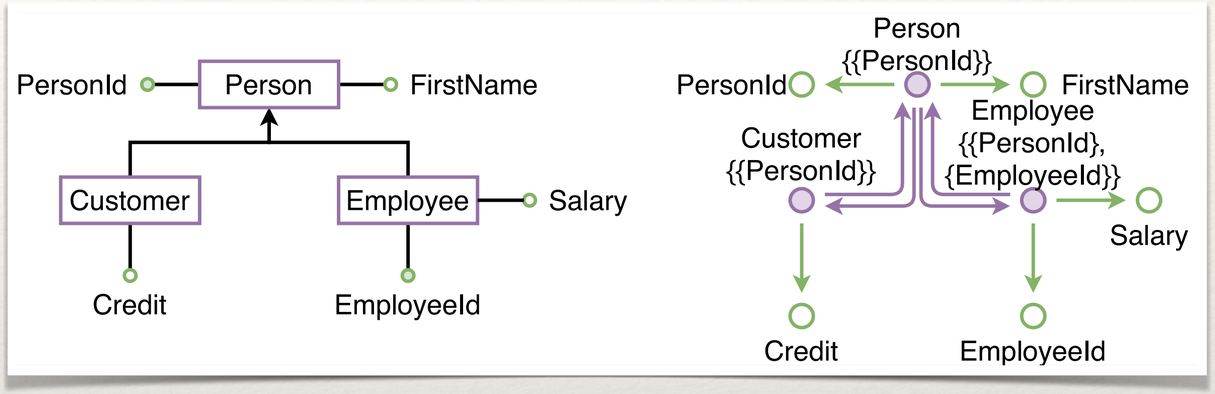
- *Unifying data structure* $\mathcal{I} = (\mathcal{O}_{\mathcal{I}}, \mathcal{M}_{\mathcal{I}}, \circ_{\mathcal{I}})$
  - Based on category of relations, i.e., *Rel*
  - Represents *data instance* of schema $\mathcal{S}$
  - *Object* internally modeled as *set of tuples* $t_1, \ldots, t_n$
    - Tuple t_i represents mapping of a given *superid* to particular values, i.e., function $t_i : \mathit{superid} \to \mathbb{V}$
  - *Morphism* is a *binary relation* between sets of tuples

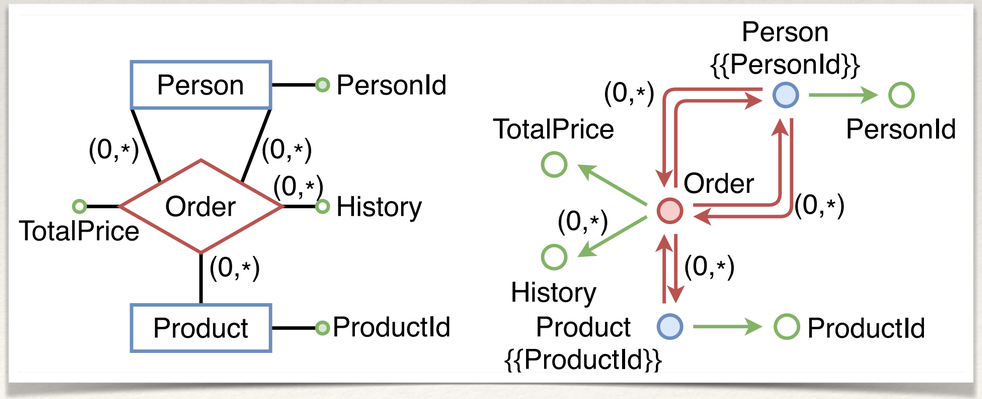


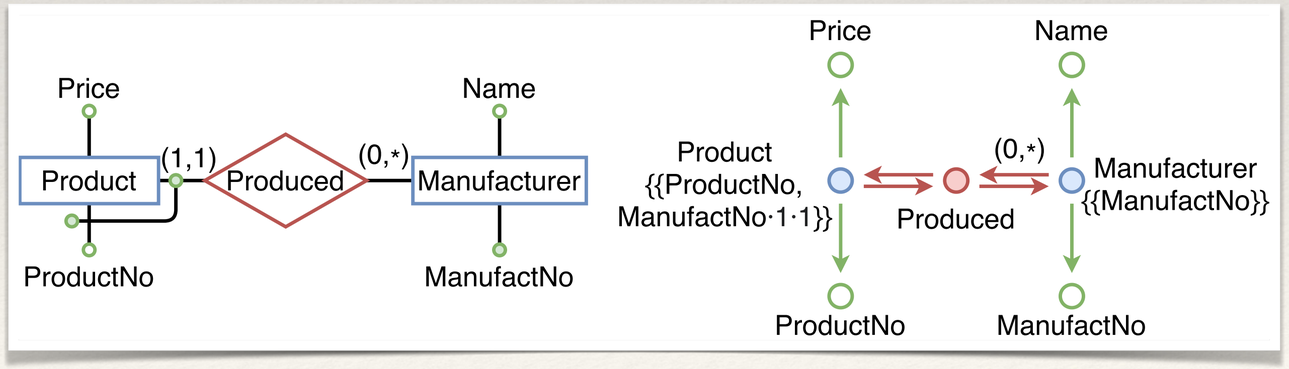


ER-to-Category Translation





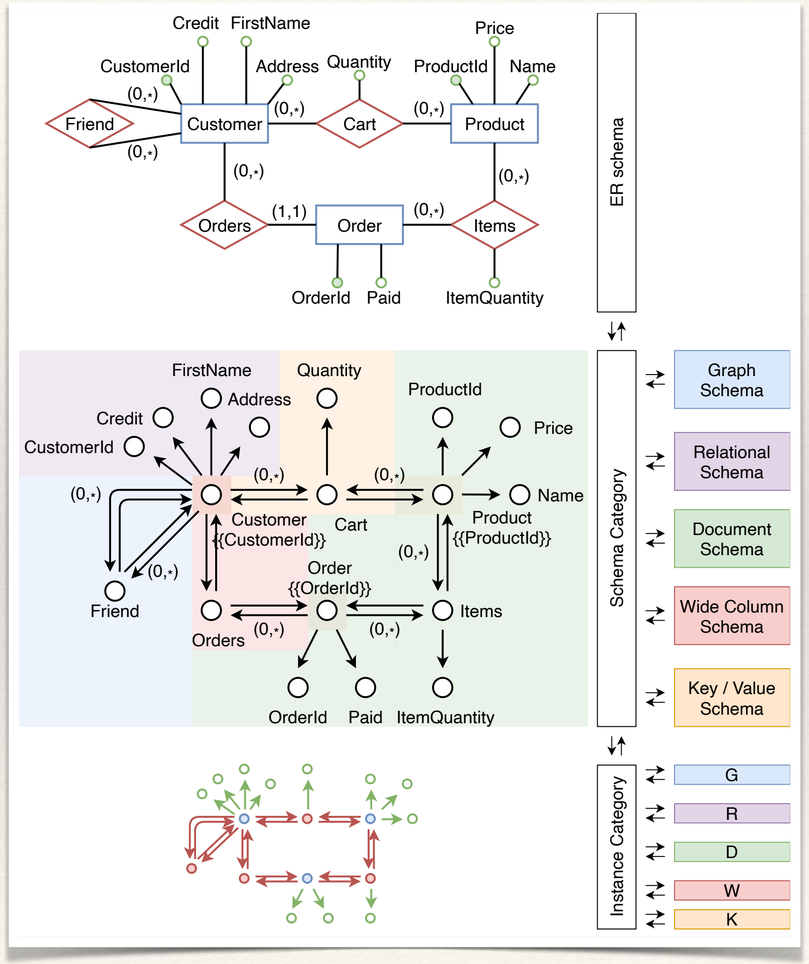




Categorical Framework (Vision)

- *Bridge* between conceptual and logical representation
  - Definition of schema and instance categories as a *unification layer* for the data models supported in MMDBMSs
  - *Simplification* of data models (entities, relationships and attributes are objects)
  - *ER-to-Category* transformation algorithm

- Whole *database framework* based on the categorical model
  - *Extensibility* of the approach, i.e., the ability to add currently not existing models
  - *Mutual transformations* of the schema/instance categories and logical models
  - *Unified* multi-model *querying* approach and processing
  - *Evolution* management



Open Questions and Challenges

- *Capture specifics* of all logical models
  - Multiple labels over the same entity (property labeled graph), i.e., *union types*
  - Possibly infinitely *nested heterogeneous arrays*
  - *Ordering* and *uniqueness* of values
  - *Null* (meta)values (appropriate choice of instance category)

- Multi-model *schema inference*, i.e., construction of a schema from already existing schema-less data
- Single *unifying query language* over multiple underlying logical data models, including *evaluation* of different *query plans*
- Evolution management, including transformations and migrations of the data
- Fully *autonomous* and *unified multi-model database*

Thank you for your attention…