Spivak - Database Migration Theory

* $\mathcal{C}$-instance is a functor $I : \mathcal{C} \to Set$ where $\mathcal{C}$ is a schema

* An **instance homomorphism** between 2 instance $I, J : \mathcal{C} \to Set$ is a natural transformation $I \to J$

* If $\mathcal{C}, \mathcal{D}$ are 2 categories, $F : \mathcal{C} \to \mathcal{D}$, (also 'schema mapping') is a functor

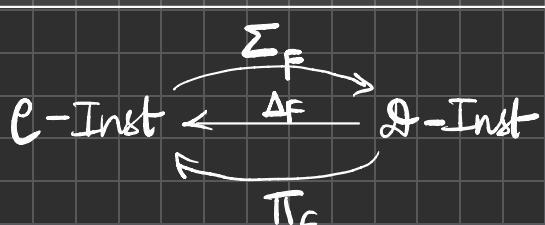
for any set-value functor $I : \mathcal{D} \to Set$ we refer 'its composite $F;I : \mathcal{C} \to Set$ as pullback of I along F.

$$\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{I} Set \qquad \mathcal{C} \xrightarrow{F;I} Set$$

Given $F : \mathcal{C} \to \mathcal{D}$, a data migration functor Δ_F turns $\mathcal{D}$-Inst to $\mathcal{C}$-Inst

Note:

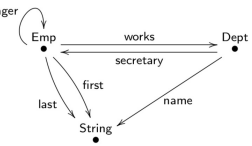
* A 'database schema' C consist of a pair $C = (G, \cong)$ where G is a graph & $\cong$ is a congruence on G.



Diversion →

Schemas and Instances

Schema (graph)



[manager.works] = [works] [secretary.works] = []

Inst.

Emp				
ID	mgr	works	first	last
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

Dept		
ID	sec	name
q10	101	CS
x02	102	Math

String
ID
Al
Bob
...

CQL code for the schema:

```
entities
    Emp
    Dept

foreign keys
    manager : Emp -> Emp
    works : Emp -> Dept
    secretary : Dept -> Emp

attributes
    first last : Emp -> string
    name : Dept -> string

path equations
    manager.works = works
    secretary.works = Department
```

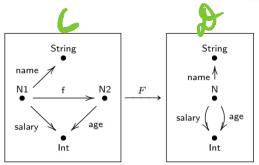
Δ → Duplicate / Destroy tables / columns
Σ → Union Data
Π → Pair & Query Data (Join)

Example: (from λ conf. video)

A **schema mapping** $F : S \to T$ is an equation-preserving function:

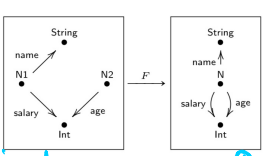
$nodes(S) \to nodes(T)$ $edges(S) \to paths(T)$

C D ↳ Not edges necessarily



C —F→ D

$F(\text{Int}) = \text{Int}$ $F(\text{String}) = \text{String}$
$F(\text{N1}) = \text{N}$ $F(\text{N2}) = \text{N}$
$F(\text{name}) = [\text{name}]$ $F(\text{age}) = [\text{age}]$ $F(\text{salary}) = [\text{salary}]$
$F(\text{f}) = []$



C-inst

N1		
ID	name	salary
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	age
4	20
5	20
6	30

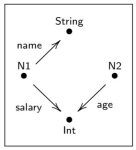
Δ_F

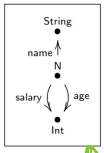
D-inst

N			
ID	name	salary	age
a	Alice	$100	20
b	Bob	$250	20
c	Sue	$300	30

What Δ_F did here?

It duplicates / divide N to N_1 & N_2 for whatever column they have (here very simple) → It 'projects'





C-inst

N1		
ID	Name	Salary
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	Age
4	20
5	20
6	30

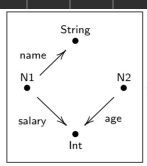
$\xrightarrow{\Sigma_F}$

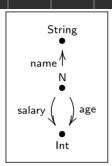
D-inst

N			
ID	Name	Salary	Age
a	Alice	$100	$null_1$
b	Bob	$250	$null_2$
c	Sue	$300	$null_3$
d	$null_4$	$null_5$	20
e	$null_6$	$null_7$	20
f	$null_8$	$null_9$	30

→ unions the data

(Note:
In every row of N either data of N_1 or N_2 is only stores no mixed (happed)
Like "outer - union"





→ Π_F → products the data

N1		
ID	name	salary
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	age
4	20
5	20
6	30

$\xrightarrow{\Pi_F}$

N			
ID	name	salary	age
a	Alice	$100	20
b	Alice	$100	20
c	Alice	$100	30
d	Bob	$250	20
e	Bob	$250	20
f	Bob	$250	30
g	Sue	$300	20
h	Sue	$300	20
i	Sue	$300	30

⇒ Queries in CQL :

Target Schema

$Q : S \to T$

source schema

Intermediate 'X'

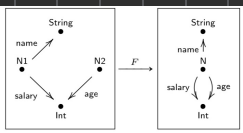
$(F : S \to X)$
$(G : T \to X)$

(query's result)

Evaluation : $eval_Q \cong \Delta_G \circ \Pi_F$

Query execution is done via this.

{ $coeval_Q$ is how the query results to source schema }



N1		
ID	**name**	**salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

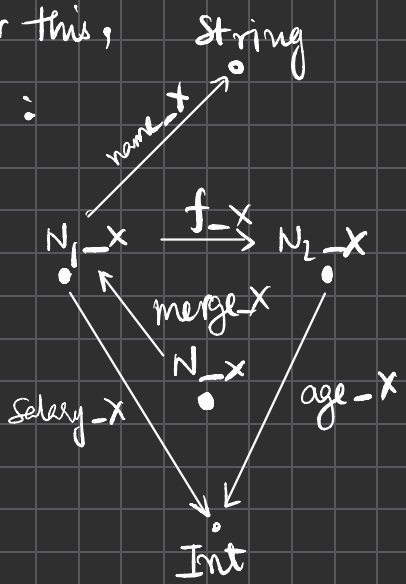
N2	
ID	**age**
4	20
5	20
6	30

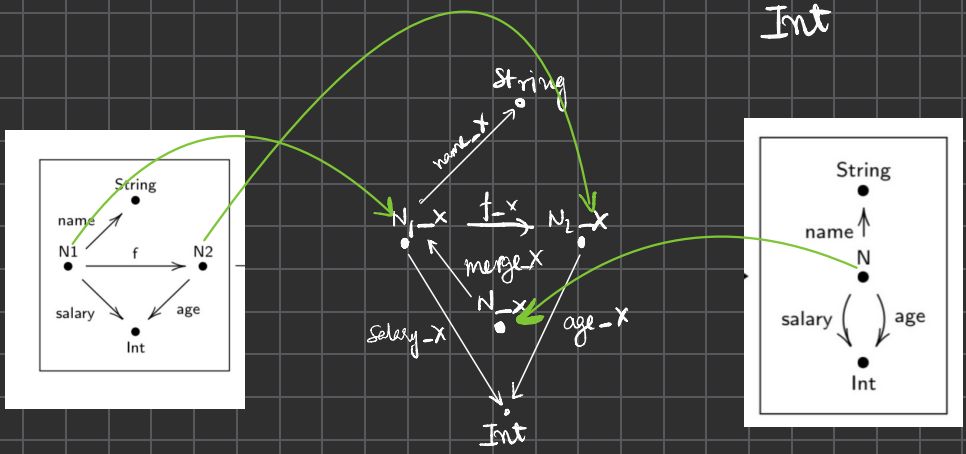
$\xrightarrow{\Pi_F}$

N			
ID	**name**	**salary**	**age**
a	Alice	$100	20
b	Alice	$100	20
c	Alice	$100	30
d	Bob	$250	20
e	Bob	$250	20
f	Bob	$250	30
g	Sue	$300	20
h	Sue	$300	20
i	Sue	$300	30

for this,

X :



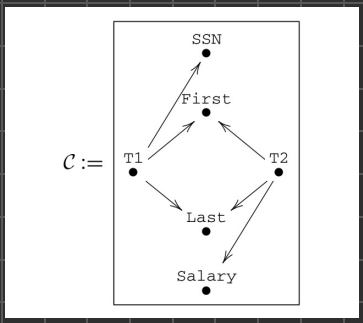


$S \xrightarrow{F} X \xleftarrow{G} T$

$eval_Q = \Delta_G \circ \Pi_F$

other e.g.s. from FDM paper

C-Inst



$\mathcal{C} :=$

T1

ID	SSN	First	Last
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

T2

ID	First	Last	Salary
T2-A101	Alice	Jones	$100
T2-A102	Sam	Miller	$150
T2-A104	Sue	Smith	$300
T2-A110	Carl	Pratt	$200

SSN

ID
115-234
118-334
122-988
198-877
342-164

First

ID
Adam
Alice
Bob
Carl
Sam
Sue

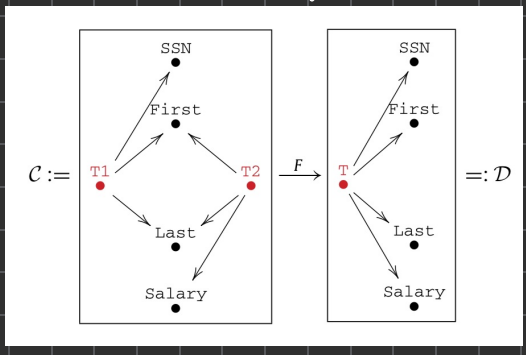
Last

ID
Jones
Miller
Pratt
Richards
Smith

Salary

ID
$100
$150
$200
$250
$300

$\mathcal{C} \xrightarrow{F} \mathcal{D}$



$\mathcal{C} :=$ $=: \mathcal{D}$

Example 2.2.1 *(Pullback).* Let $F : \mathcal{C} \to \mathcal{D}$ be the translation given in diagram (7). In this example, we explore the data migration functor $\Delta_F : \mathcal{D}$-**Inst** $\to \mathcal{C}$-**Inst**[4] by applying it to a $\mathcal{D}$-instance J. Note that even though our translation F points forwards (from $\mathcal{C}$ to $\mathcal{D}$), our migration functor Δ_F points "backwards" (from $\mathcal{D}$-instances to $\mathcal{C}$-instances). We will see why it works that way, but first we bring the discussion down to earth by working with a particular $\mathcal{D}$-instance.

Consider the instance J, on schema $\mathcal{D}$, defined by the table

$J :=$

T

ID	SSN	First	Last	Salary
XF667	115-234	Bob	Smith	$250
XF891	122-988	Sue	Smith	$300
XF221	198-877	Alice	Jones	$100

(8)

$\mathcal{D}$-Inst

Δ_F

and having the four leaf tables from Example 2.1.1, display (5). Pulling back along the translation F, we are supposed to get an instance $\Delta_F(J)$ on schema $\mathcal{C}$, which we must describe. But the description is easy: $\Delta_F(J)$ splits up the columns of table T according to the translation F. The four leaf tables will be exactly the same as above (i.e. the same as in Example 2.1.1, (5)), and the two fact tables will be something like[5]

T1

ID	SSN	First	Last
XF667T1	115-234	Bob	Smith
XF891T1	122-988	Sue	Smith
XF221T1	198-877	Alice	Jones

T2

ID	First	Last	Salary
A21	Alice	Jones	$100
A67	Bob	Smith	$250
A91	Sue	Smith	$300

(9)

After Π_F (Rightpush forward functor)

T

ID	SSN	First	Last	Salary
T1-002T2-A104	122-988	Sue	Smith	$300
T1-003T2-A101	198-877	Alice	Jones	$100

After Σ_F (left Push f.f.)

T

ID	SSN	First	Last	Salary
T1-001	115-234	Bob	Smith	T1-001.Salary
T1-002	122-988	Sue	Smith	T1-002.Salary
T1-003	198-877	Alice	Jones	T1-003.Salary
T2-A101	T2-A101.SSN	Alice	Jones	$100
T2-A102	T2-A102.SSN	Sam	Miller	$150
T2-A104	T2-A104.SSN	Sue	Smith	$300
T2-A110	T2-A110.SSN	Carl	Pratt	$200