ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

# NOSQL
# JOINS, DENORMALIZATION & COST MODELS

nicolas.travers@devinci.fr



## NOSQL VS JOINTURES

Distributed context incompatible with links between data collections

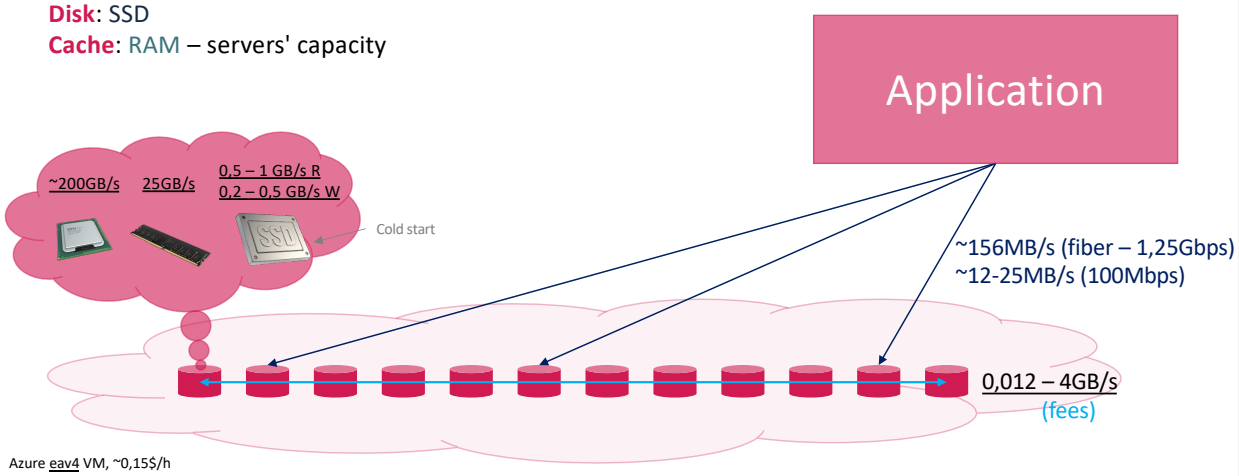
- Need for a cost model
- Cost based on network communications - MapReduce
- Multi-server join

Considering two data collections

- Joins between two collections
  - On which server is the join performed?
  - Distribution of data to be joined?
- Network cost problem (more expensive than local)

# NOSQL VS JOINS: COST

**Network**: inter communications + intra communications
**Disk**: SSD
**Cache**: RAM – servers' capacity




Azure eav4 VM, ~0,15$/h





ESILV
Joins, Denormalization & Cost Models

# CLOUD'S COST MODEL

Need for a cost model

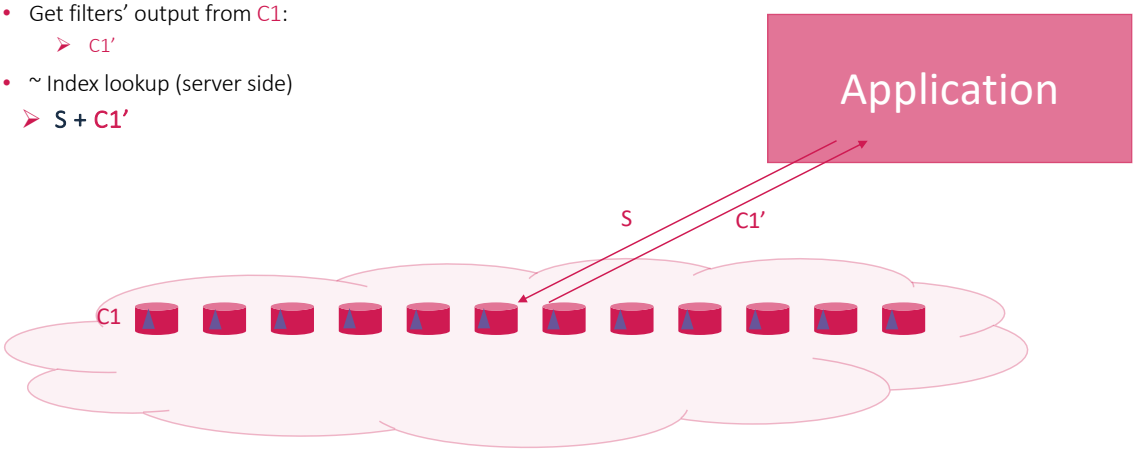
- Estimate the cost of queries submitted to the DB
  - 1 query = 1 sequence of operations
  - 1 operation = 1 algorithm = 1 cost formula
- Operations:
  - Filter / Selection - reduces number of documents = *Map*
  - Projection - reduces document size = *Map/Reduce output*
  - Grouping - aggregates results = Reduce
- Cost impacted by:
  - Communications
  - CPU
  - RAM access
    - Disk is ignored after "cold start"



LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr
Nicolas.travers@devinci.fr

# FILTER

- Data access of servers (S) => ***Map on collection C1***
- Get filters' output from C1:
  - C1'
- ~ Index lookup (server side)
  - S + C1'



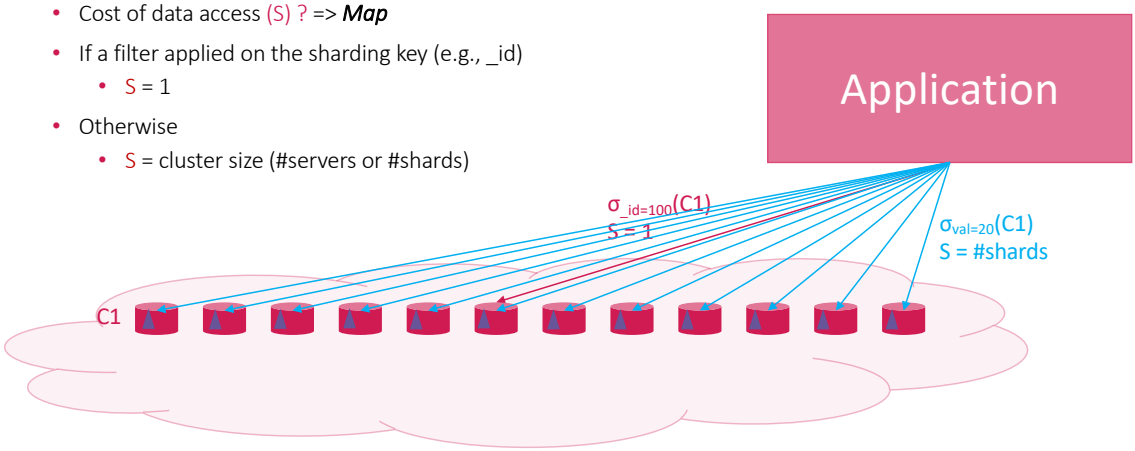






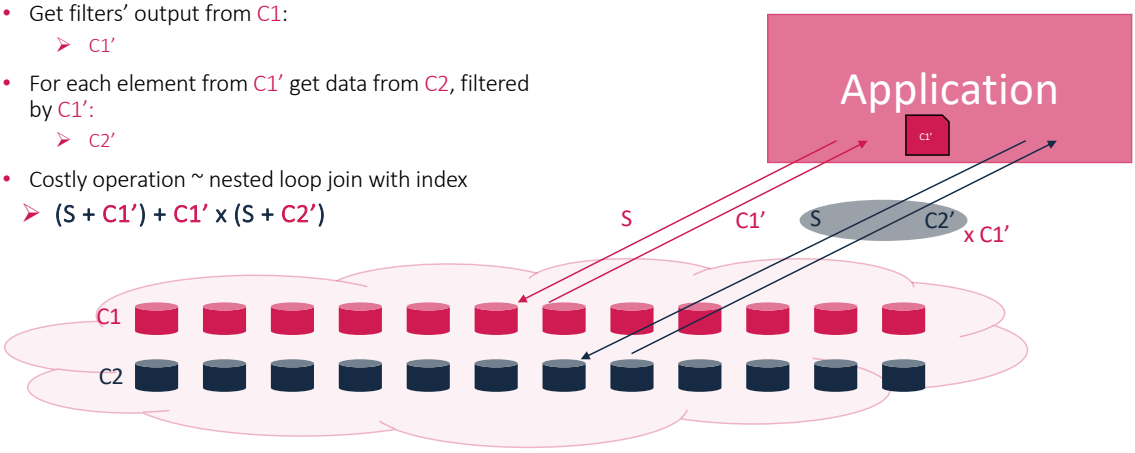

# DATA ACCESS « S »

- Cost of data access (S) ? => ***Map***
- If a filter applied on the sharding key (e.g., _id)
  - S = 1
- Otherwise
  - S = cluster size (#servers or #shards)



# APPLICATION JOIN

- Data access of servers (S) => ***Map on collection C1***
- Get filters' output from C1:
  - C1'
- For each element from C1' get data from C2, filtered by C1':
  - C2'
- Costly operation ~ nested loop join with index
  - (S + C1') + C1' x (S + C2')



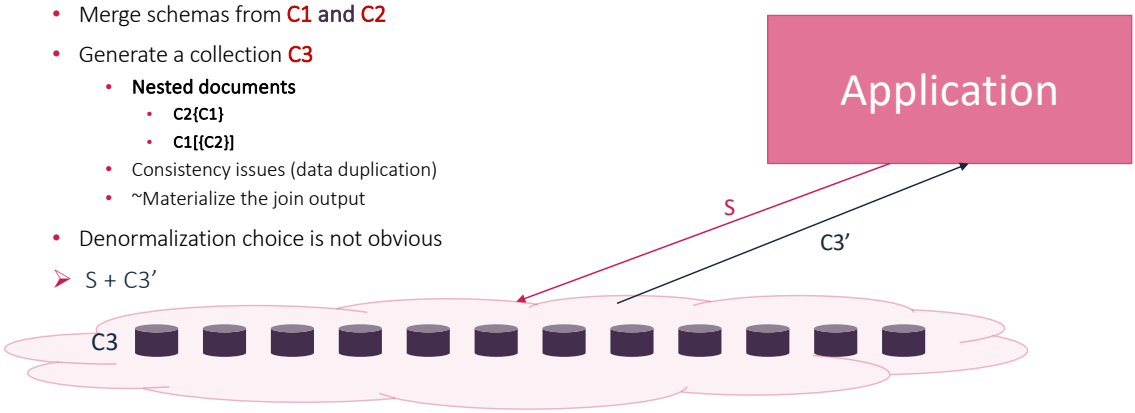







# APPLIED TO A DENORMALIZED COLLECTION

- Merge schemas from C1 and C2
- Generate a collection C3
  - **Nested documents**
    - **C2{C1}**
    - **C1[{C2}]**
  - Consistency issues (data duplication)
  - ~Materialize the join output
- Denormalization choice is not obvious
  - S + C3'



# JOIN APPLICATION AT SERVER SIDE

- Join data can be distributed on servers for local computation (same amount of data transfer)
- Few NoSQL databases enable this distributed join
  - e.g., *Hadoop/Pig, Spark, CouchBase, MongoDB 6.1, Cassandra (future version)*
  - Similar to application join (distributed)
  - Depends on collection sizes to be distributed
  - Relational-based join algorithms (hash, tri-merge)

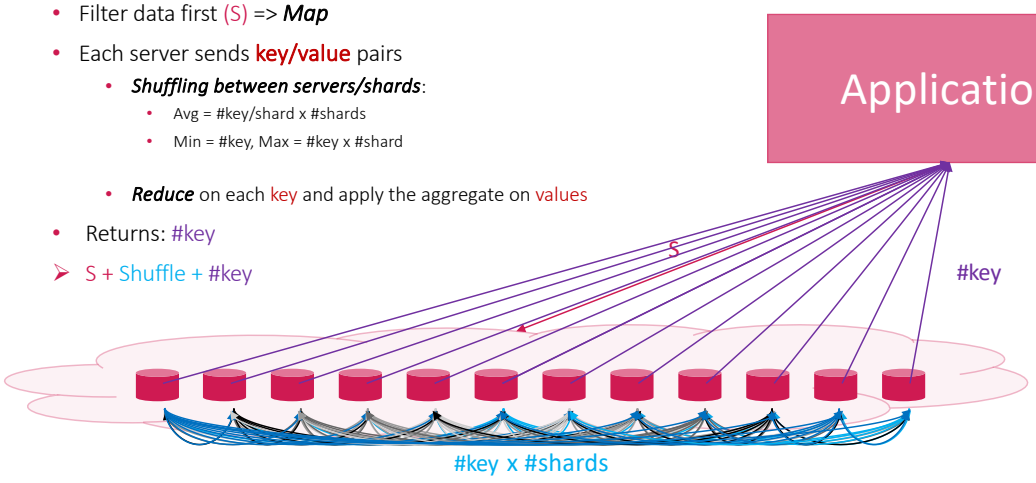





# AGGREGATES "*GROUP BY + COUNT*" ?

- Filter data first (S) => ***Map***
- Each server sends **key/value** pairs
  - ***Shuffling between servers/shards***:
    - Avg = #key/shard x #shards
    - Min = #key, Max = #key x #shard
  - ***Reduce*** on each key and apply the aggregate on values
- Returns: #key

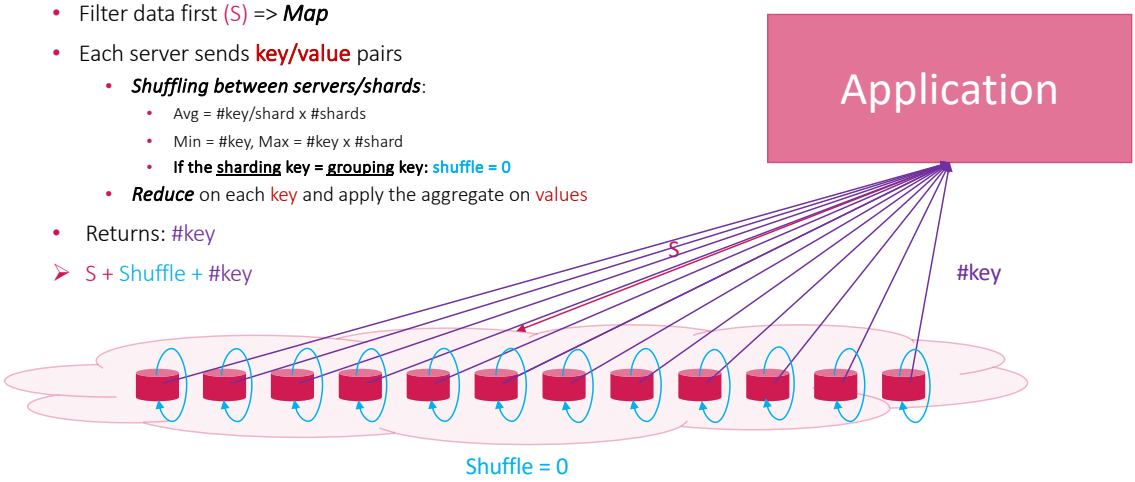
➢ S + Shuffle + #key



# AGGREGATES "*GROUP BY + COUNT*" ?

- Filter data first (S) => ***Map***
- Each server sends **key/value** pairs
  - ***Shuffling between servers/shards***:
    - Avg = #key/shard x #shards
    - Min = #key, Max = #key x #shard
    - **If the sharding key = grouping key: shuffle = 0**
  - ***Reduce*** on each key and apply the aggregate on values
- Returns: #key

➢ S + Shuffle + #key

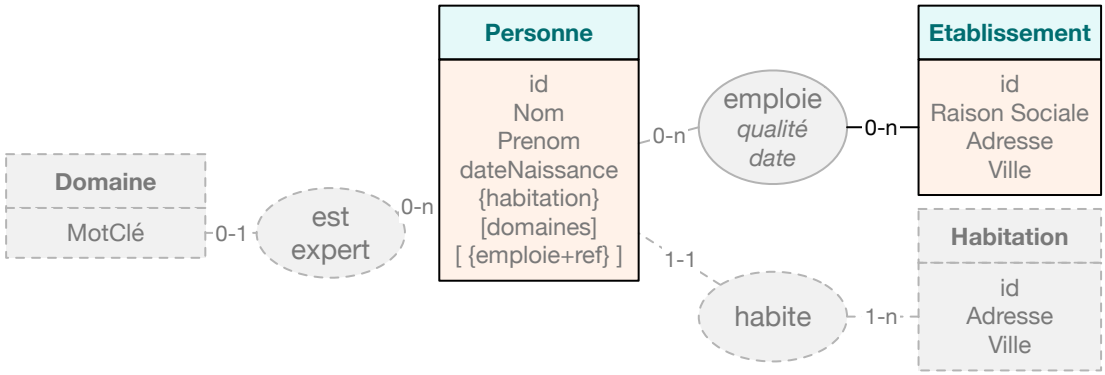






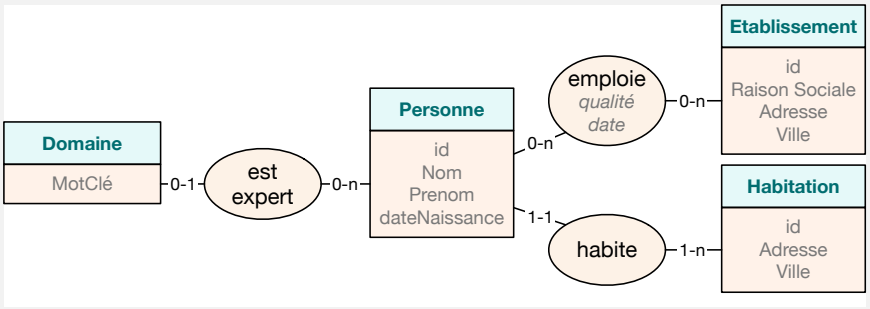

ESILV
Joins, Denormalization & Cost Models

# DENORMALIZATION



# SCHEMA DENORMALIZATION

- Which centrale entity to choose?
- How to merge?
- Which impact?










# DENORMALIZATION: METHODOLOGY

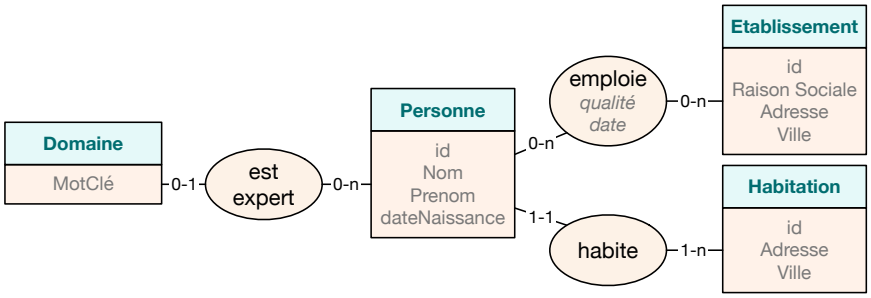
- Query-Driven [Chebotko15]
- Document-oriented NoSQL modelling [Mason15]
- Model-Driven-Architecture [Abdelhedi et al. 17, Mali et al. 20, Mali et al. 22]
- Goal:
  - Avoid costly joins
  - Combine barely updated data and/or independent data

Consistency issues!

1 – [Chebotko15] https://pdfs.semanticscholar.org/22c6/%20740341ef13d3c5ee52044a4fbaad911f7322.pdf
2 – [Mason15] http://proceedings.informingscience.org/InSITE2015/InSITE15p259-268Mason1569.pdf
3 – [Abdelhedi et al. 17] https://cedric.cnam.fr/index.php/publis/article/AAA17
4 – [Mali et al. 20] – https://link.springer.com/content/pdf/10.1007%2F978-3-030-59003-1_9.pdf
5 – [Mali et al. 22] - https://link.springer.com/chapter/10.1007/978-3-031-05760-1_31

# DENORMALIZATION: OPERATIONS

- Merge (A{B} & B[{A}])
- Split ($A_1$ & $A_2$)
- Materialization (aggregates)
- Surcharge (duplicates)

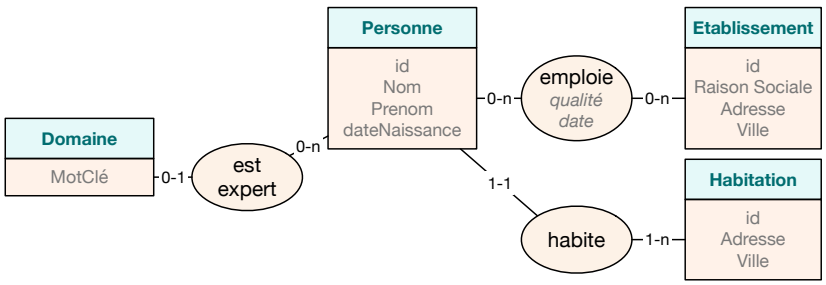






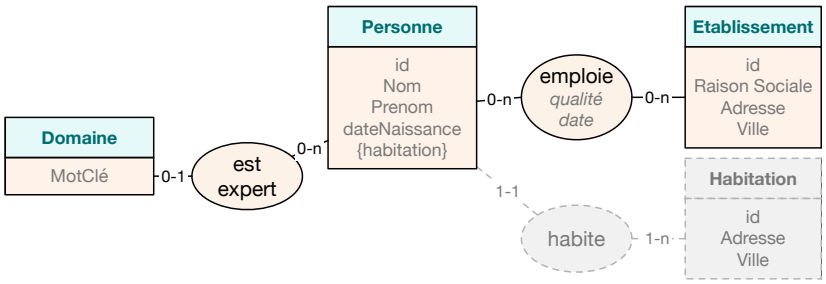


# DENORMALIZATION: MODELISATION STEPS



# DENORMALIZATION: MODELISATION STEPS

- **Merges:**
  1. Frequent queries *(e.g., join between Personne & Habitation)*



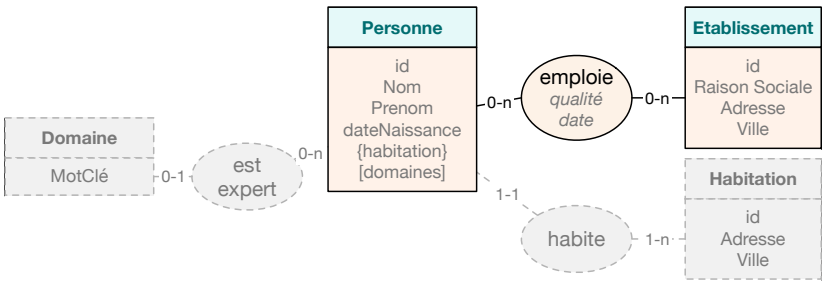




ESILV
Joins, Denormalization & Cost Models

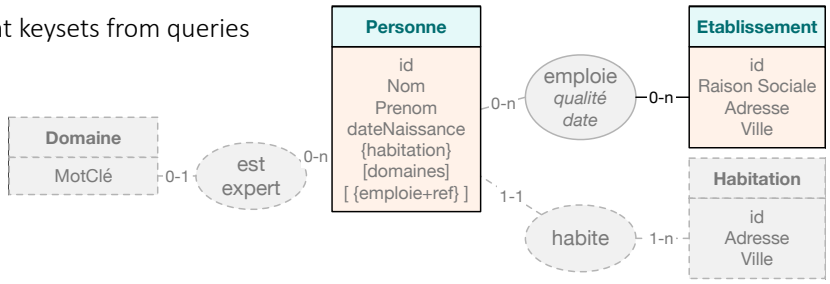
# DENORMALIZATION: MODELISATION STEPS

- **Merges:**
  1. Frequent queries *(e.g., join between Personne & Habitation)*
  2. Independent data *(e.g., Domain in Personne)*



# DENORMALIZATION: MODELISATION STEPS

- Merges:
  1. Frequent queries *(e.g., join between Personne & Habitation)*
  2. Independent data *(e.g., Domain in Personne)*
  3. For 0-n cardinalities on both sides:
     1. Nesting the foreign key *(e.g., employer id in the list)*
- *Splits:*
  1. Independent keysets from queries








ESILV
Joins, Denormalization & Cost Models

# DENORMALIZATION: MERGE

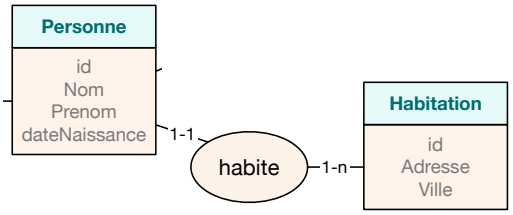
- Nb merges, Fubini number =

$$Fn_{|\mathcal{L}|} = \sum_{k=0}^{|\mathcal{L}|} k! \times \binom{|\mathcal{L}|}{k}$$

⇒ *Document/Wide-columns Oriented*

P{H}: {"idP":1, "habitation": {"idH":10,...}}
{"idP":11, "habitation": {"idH":10,...}}

H[{P}]: {"idH":10, "habitants":[
{"idP":1, ...},
{"idP":11, ...},
...]}




[Mali et al. 22]



LIKE,SHARE, FOLLOW ! @esiv_paris | esilv.fr
Nicolas.travers@devinci.fr

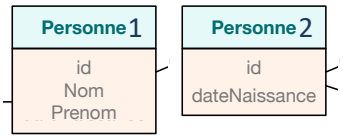
# DENORMALIZATION: SPLIT

- Nb splits, Bell number: $B_{|keys(r)|} = \sum_{k=0}^{|keys(r)|} \binom{|keys(r)|}{k} \times B_{|keys(r)|-1}$

*=> Columnar oriented (eventually wide-column)*

P1: {"idP":1, "Nom":"Skywalker", "Prenom":"Luke"}

P2: {"idP":1, "dateNaiss":"19 av. BY"}




[Belbachir et al. 21]
* Dependent on nb keys (without Primary Key)






# DENORMALIZATION: COMBINAISON

- Nb merges: $Fn_{|\mathcal{L}|} = \sum_{k=0}^{|\mathcal{L}|} k! \times \binom{|\mathcal{L}|}{k}$
- Nb splits: $B_{|keys(r)|} = \sum_{k=0}^{|keys(r)|} \binom{|keys(r)|}{k} \times B_{|keys(r)|-1}$
- Nb denormalizations: $|\mathcal{M}| = (Fn_{|\mathcal{R}|} \times d) \times \prod_{k=1}^{|\mathcal{R}|} B_{|keys(r_k)|}$

# possible splits among rows

# possible merges per split DM

# Hierarchy depth impacting number of possible merges vs splits

$d = \sum_{k=1}^{|\mathcal{R}|} keys(r_k) - |\mathcal{R}| + 1$

[Mali 24, Mali et al. 25]

# DENORMALIZATION: CONCLUSION

- Complex to determine a good denormalization
  - Materialization / Surcharge
  - Beware of updates – inconsistency (*no normal forms*)

➢ The use case can guide the denormalization

- Data models compatible with queries
- How to handle complex and evolving use case?







# RECAP: HOW TO CHOOSE A NOSQL SOLUTION? 225+ NOSQL SOLUTIONS + RELATIONAL

- NoSQL Family
  - Querying languages: *Expressivity vs simplicity*
  - *Consistency issues*
- *Sharding* strategy
  - vs secondary indexes
  - Primary key / foreign key / join key / group key
- Availability vs Consistency: CAP
- Community support

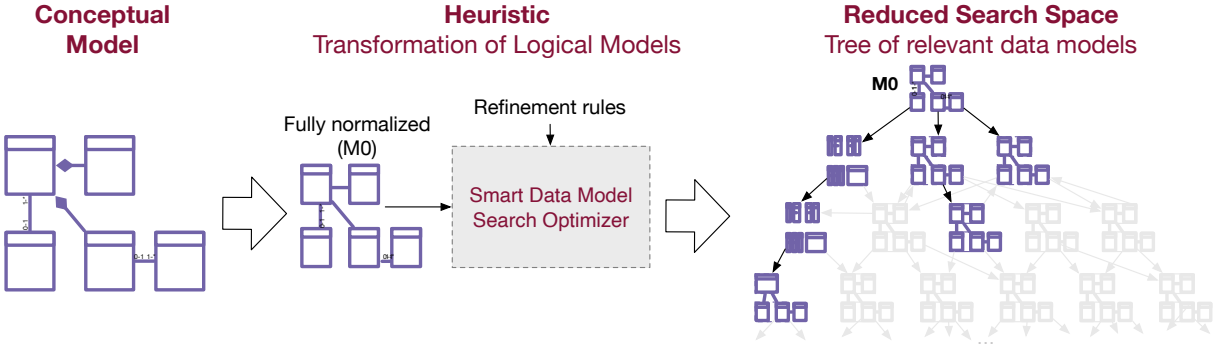
Complex use cases

- Best denormalization / tradeof ?
- Filter/Join/Group queries + frequencies
- Data statistics (Data distribution)
- Other constraints :
  - QoS
  - Environmental impact
  - Financial cost
  - GDPR
  - Use case evolution

# A GLOBAL MODEL-DRIVEN DENORMALIZATION

[J. MALI, S. AHVAR, F.ATIGUI, A. AZOUGH, N. TRAVERS - RCIS 2022]

- Production of eligible data models by recursive denormalization
- Guide the choice of the best tradeof





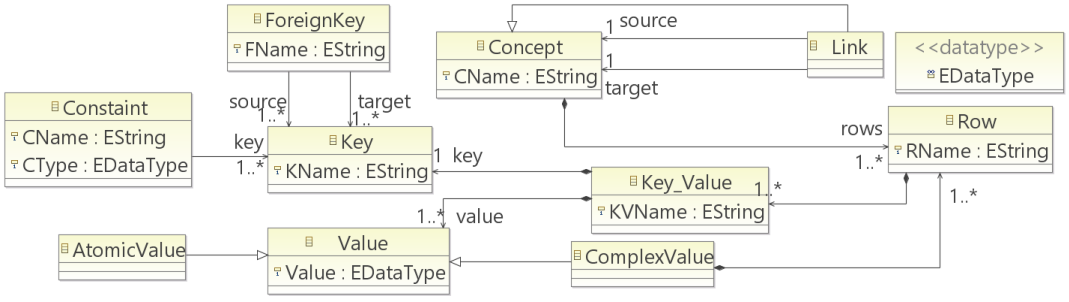





# THE 5FAMILIESMODEL: A UNIQUE METAMODEL

[J. MALI, F.ATIGUI, A. AZOUGH, N. TRAVERS – DEXA 2020]

- Adapted to all NoSQL families + Relational
- Transformations
  - QVT rules: merge & split
  - Data models generation heuristic (based on the use case)



# 5FMHEURISTIC

**Generation heuristic:**

- Remove cycles during recursivity
- Remove Data Models duplicates
- Only eligible Data Models
  - Guided by queries

$$|\mathcal{M}^{opt}| = Fn_{|refs(\mathcal{Q})|} \times \prod_{k=1}^{|\mathcal{R}|} |KeySet(\mathcal{Q}_k)|$$

**Algorithm 1** 5FMHeuristic

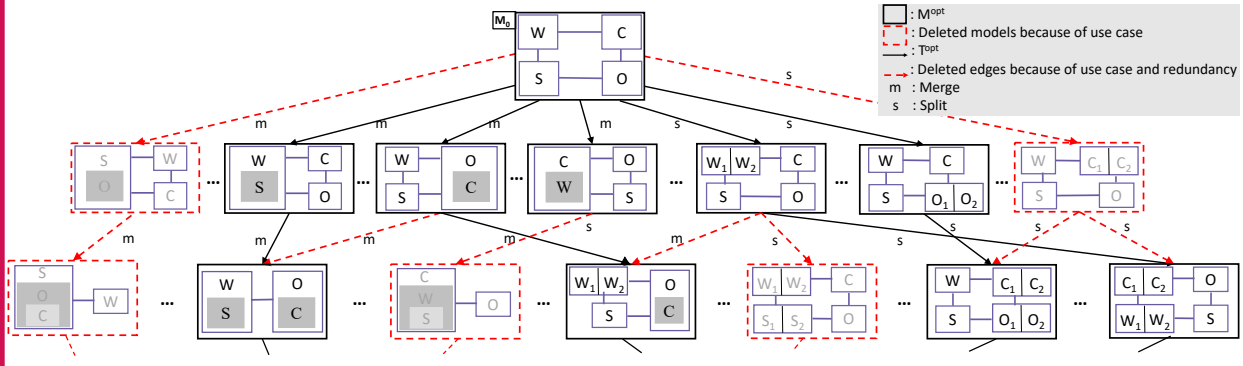
**global:** A list of data models $\mathcal{M}^{opt}$ from *5FamiliesModel*
**input:** a data model $\mathcal{M}_i$ from *5FamiliesModel*, a list of input queries $\mathcal{Q}$ (a query is a set of keys from $\mathcal{M}_i$), a list of used queries $\mathcal{Q}'$
**init:** $\mathcal{M}_i = \mathcal{M}_0$, the relational data model, $\mathcal{M}^{opt} = \emptyset$, $\mathcal{Q}' = \emptyset$

**procedure** 5FMHEURISTIC($\mathcal{M}_i, \mathcal{Q}, \mathcal{Q}'$)
  $\mathcal{M}^{opt} := \mathcal{M}^{opt} \cup \mathcal{M}_i$
  **for all** key $K \in \mathcal{K} \wedge$ !*Constraint*($K$, CType=PK) **do**
    $r := Row(K)$
    **if** $K \notin \mathcal{Q} \cup \mathcal{Q}'$ **then**
      **if** $\exists k \in r | k \neq K \wedge$ !$Constraint(k,$ Ctype=PK) **then**
        $\mathcal{M}_{i+1}$ := SPLIT($\mathcal{M}$, K)
        5FMHEURISTIC($\mathcal{M}_{i+1}$, $\mathcal{Q}$, $\mathcal{Q}'$)
      **else**
        **for all** $q \in \mathcal{Q}$ **do**
          **if** $\forall k \in q | Row(k) = r, \exists k \in R | k \notin q \wedge$ !$Constraint(k, Ctype = PK) \wedge \forall q' \in \mathcal{Q} | \bar{q} \neq q'$ **then**
            $\mathcal{M}_{i+1}$ := SPLIT($\mathcal{M}_i$, q)
            5FMHEURISTIC($\mathcal{M}_{i+1}$, $\mathcal{Q} - q$, $\mathcal{Q}' \cup q$)
          **else if** $\exists k \in q | Row(k) \neq r \wedge Concept(k) = Concept(K)$ **then**
            **else if** $\exists k_1, k_2 \in q | ref_{k_1 \to k_2} \vee ref_{k_2 \to k_1} \in \mathcal{L}$ **then**
              $\mathcal{M}_{i+1}$ := MERGE($\mathcal{M}_i$, q, $k_1, k_2$)
              5FMHEURISTIC($\mathcal{M}_{i+1}$, $\mathcal{Q} - q$, $\mathcal{Q}' \cup q$)
              $\mathcal{M}_{i+1}$ := MERGE($\mathcal{M}_i$, q, $k_2, k_1$)
              5FMHEURISTIC($\mathcal{M}_{i+1}$, $\mathcal{Q} - q$, $\mathcal{Q}' \cup q$)
              $\mathcal{M}_{i+1}$ := REFERENCETOEDGE($\mathcal{M}_i$, q, $k_1, k_2$)
              5FMHEURISTIC($\mathcal{M}_{i+1}$, $\mathcal{Q} - q$, $\mathcal{Q}' \cup q$)



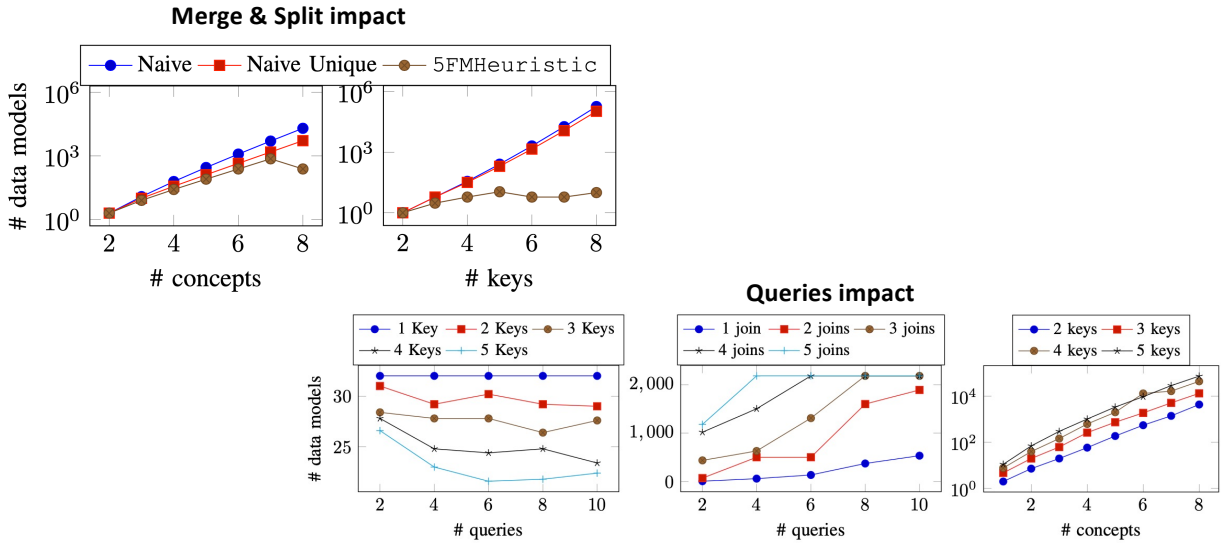

ESILV
Model Driven Guide

# 5FMHEURISTIC




From **2,313** solutions to **27**

# BENEFITS








ESILV
Model Driven Guide

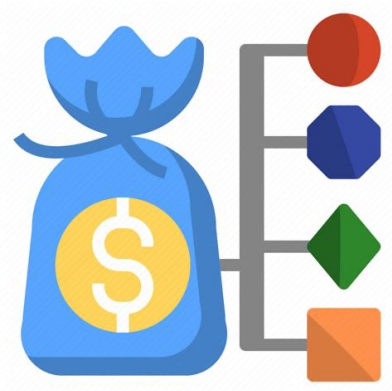
# HOW TO COMPARE DATA MODELS?

- Require a generic cost model based on the meta-model
  - Execution time
  - Environmental impact
  - Financial cost
- Contexte modeling
  - Queries
  - Statistics (data & queries)
  - Cluster information (data center, servers spec, budget, etc.)
  - GDPR: Private vs Public Cloud
  - Data across several database types: Polystore



LIKE,SHARE, FOLLOW ! @esiv_paris | esilv.fr
Nicolas.travers@devinci.fr

# A GLOBAL COST MODEL









# MULTIDIMENSIONAL COST MODEL* [MALI ET AL. – ARXIV'23,BDA'23,EDBT'24,IS'25]

### TIME COST

- Query time
  - Communication
  - RAM
  - Storage
- Use case time (frequencies)

### FINANCIAL COST

- Cluster size
  - #servers
  - Server spec
- Queries
  - External communications (Cloud fees)

### ENVIRONMENTAL COST

- Cluster/storage carbon footprint (life cycle)
- Queries' carbon footprint
  - RAM
  - Storage
  - Communications

Based on data volumes (GB)

*The generic cost model gives estimations for comparison purposes. No real values

# TIME COST

- For a query (q) :
  - Communication time: *Bandwidth*
    - $bandwidth_{network}$ = 100 MB/s – 1GB/s
  - Server-side computation in main memory: Read/Write *RAM*
    - $bandwidth_{RAM}$ = 25 GB/s
    - CPU negligeable
  - ➢ Calculate data volumes
    - Messages size, documents size
    - Data types:
      - Numbers: 1B
      - VARCHAR : ~8B -> 200B
      - Key/value pairs: 12B
    - ➢ Size ~ $\sum_{k=0}^{n}(size_{kv} + size_k)$
      - ➢ Query size
      - ➢ Results size
      - ➢ Documents size







# TIME COST: FILTER QUERIES

- Communications
  - $vol_{network}(q) = S * size_{query} + |res(q)| * size_{msg}$
  - Joins: $vol_{network}(q_1) + |res(q_1)| * vol_{network}(q_2)$

- Server-side computation
  - Parallelism = max time of each server (*n* servers)
  - $vol_{RAM}(q) = max(vol_{RAM}(q,1), \dots, vol_{RAM}(q,n))$
  - $vol_{RAM}(q,n) = index(q) + sel(att) * |coll_{q,n}| * size_{doc(q)}$
    - Secondary local index (~1 MB) -> pointers on pieces of data
    - Data access
    - **Sel**: Filter query selectivity (q) applied to the local storage

Volume of data read by the query

Volume of data on server *n*

# TIME COST: AGGREGATE QUERIES

- Communications during the **Map** & **Reduce** phases
  - $\boldsymbol{vol_{network}(q) = S * size_{query} + shuffle * size_{msg} + |res(q)| * size_{msg}}$
  - Joins: $\boldsymbol{vol_{network}(q_1) + |res(q_1)| * vol_{network}(q_2)}$
  - Can be combined with filter queries

- Server-side computation for **Map/Reduce**
  - $\boldsymbol{vol_{RAM}(q,n) = reduce_{local} + reduce_{global}}$
  - $\boldsymbol{reduce = \sharp\, doc * size_{doc} * 2}$
    - Local, $\sharp\, doc = nb\ de\ doc\ agrégés$ (par serveur)
    - Global, $\sharp\, doc = nb\ de\ doc\ reçus\ par\ le\ shuffle$ (par serveur)

Read & write for each reduce stage







# TIME COST: TOTAL

- Take into account the whole use case
  - Set of queries
  - Query frequencies

- $\boldsymbol{time(DB) = \sum_{q=1}^{Q} \left( \frac{vol_{network}(q)}{bandwidth_{network}} + \frac{vol_{RAM}(q)}{bandwidth_{RAM}} \right) * freq(q)}$

Internal vs external cloud communications can be split
(S & res)

# FINANCIAL COST

- **External** bandwidth fees:
  - $externalFees$ = 0,011 €/GB
  - Split inter/extern: $time_{network}(q)$
- Virtual machines fees (VMs) :
  - Watch the configuration:
    - RAM
    - Storage (SSD)
    - Bandwidth
    - Price
  - Cluster dimensioning
    - Data volumes
    - Must fit in main memory (avoid cold start)
    - Storage capacity (replication x3 !!!)

| Azure server | capacity RAM | capacity SSD | Price (per month) | Bandwidth |
|---|---|---|---|---|
| B1ms | 1 GB | 4 GB | 30,13 € | 100 Mbps |
| B1s | 2 GB | 4 GB | 25,63 € | 100 Mbps |
| L8as | 64 GB | 1,9 TB | 313,25 € | 32 Gbps |







# DATABASE SIZE

For each collection (C)

- $\boldsymbol{vol_{DB} = \sum_{c=1}^{C} size_{doc}(c) * \#doc(c)}$

For database evolution purpose

Cluster dimensioning:

- $\boldsymbol{\#servers = max\left(\left\lceil \frac{vol_{DB} * 3}{capacity_{storage}} \right\rceil, \left\lceil \frac{vol_{DB}}{capacity_{RAM}} \right\rceil\right) * 2}$

Monthly cost:

- $\boldsymbol{price(DB) = price(s) * \#servers + externalFees * \sum_{q=1}^{Q} vol_{external\ network}(q) * freq(q)}$

# ENVIRONMENTAL IMPACT

- Carbon footprint measure
  - kg CO2-eq
  - Variation between countries (DataCenter)
- Infrastructure impact (fixed cost)
  - Storage: 0,0031 kg CO2-eq/GB
  - Lifecycle of servers: 1 283kg CO2-eq/serveur
- Queries Impact
  - Bandwidth
    - $CO2_{network}$= 0,0110 kg CO2-eq/GB
  - Computation RAM/CPU
    - $CO2_{RAM}$= 0,0280 kg CO2-eq/GB







# ENVIRONMENTAL IMPACT: THE USE CASE

Network impact

- $impact_{network}(q) = vol_{network}(q) * CO2_{network}$

Simplification of computation in RAM

RAM/CPU impact

- $impact_{RAM}(q) = vol^{*}_{RAM}(q) * CO2_{RAM} = \sum_{n=1}^{N} vol_{RAM}(q,n) * CO2_{RAM}$

Global impact

- $impact(DB) = \sum_{q=1}^{Q}(impact_{network}(q) + impact_{RAM}(q)) * freq(q)$

# MULTI-DIMENSIONAL COST MODEL

$$time(DB) = \sum_{q=1}^{Q} \left( \frac{vol_{network}(q)}{bandwidth_{network}} + \frac{vol_{RAM}(q)}{bandwidth_{RAM}} \right) * freq(q)$$

$$price(DB) = price(s) * max\left( \left\lceil \frac{vol_{DB} * 3}{capacity_{storage}} \right\rceil, \left\lceil \frac{vol_{DB}}{capacity_{RAM}} \right\rceil \right) * 2 + externalFees * \sum_{q=1}^{Q} vol_{external\ network}(q) * freq(q)$$

$$impact(DB) = \sum_{q=1}^{Q} \left( vol_{network}(q) * CO2_{network} + \sum_{n=1}^{N} vol_{RAM}(q,n) * CO2_{RAM} \right) * freq(q)$$

Not true for environmental impact ($vol^{*}_{RAM}(q)$)

In practice:

- $vol_{RAM}(q) \ll vol_{network}(q)$
- $bandwidth_{RAM} = 250 * bandwidth_{network}$
- $CO2_{RAM} \approx 2{,}5 * CO2_{network}$

➢ The network part of the cost model remains important

➢ Better target **environmental** impact optimization with time and price constraints (QoS & budget)




Nicolas.travers@devinci.fr