FACT-DM : A Framework for Automated Cost-Based Data Model Transformation

Jihane Mali, Shohreh Ahvar, Faten Atigui, Ahmed Azough, Nicolas Travers

Demonstration Paper

FACT-DM : A Framework for Automated Cost-Based Data Model Transformation

Jihane Mali
Léonard de Vinci Pôle Universitaire, Research Center
Paris La Défense, France
jihane.mali@devinci.fr
ISEP
Paris, France
jihane.mali@isep.fr

Shohreh Ahvar
Nokia Networks
Massy, France
shohreh.ahvar@nokia.com

Faten Atigui
CEDRIC, Conservatoire National des Arts et Métiers (CNAM)
Paris, France
faten.atigui@cnam.fr

Ahmed Azough
Léonard de Vinci Pôle Universitaire, Research Center
Paris La Défense, France
ahmed.azough@devinci.fr

Nicolas Travers
Léonard de Vinci Pôle Universitaire, Research Center
Paris La Défense, France
nicolas.travers@devinci.fr

ABSTRACT

As data continues to grow at an unprecedented rate, the complexity of database systems also increase significantly, requiring information systems (IS) architects to constantly adapt their data model and carefully select the optimal solution(s) for storing and managing data that align with new queries, settings, and constraints. Having a tool to visually show the impact of different changes, will help IS architects in their decision making. In this paper, we propose a framework to demonstrate the impact of denormalization of data models on their cost and consequently IS architects can choose the best trade-off.

1 INTRODUCTION

Data's explosion especially characterized by the 3V (Volume, Variety & Velocity) has opened up major research issues related to modeling, manipulating, and storing massive amounts of data [15]. The resulting so-called NoSQL systems correspond to *four families* of data structures: key-value oriented (KVO), wide-column oriented (CO), document oriented (DO), and graph oriented (GO).

Guaranteeing the efficiency and availability of information is challenging for Information Systems (IS). Better restructuring of database schemas is needed but often driven by subjective choices, which does not take all the factors into account. Approaches from the literature [1–5, 7, 8, 13, 14, 16] are also essentially oriented towards a specific NoSQL solution or family; therefore the optimal solution may be missed, by not taking into account IS use cases.

The main issue is to provide the optimal data model for a given IS use case. To solve this issue, we need the cost estimation of a solution according to the data model, statistics and queries [12]. This issue is hardly tackled for NoSQL solutions, especially when choosing the target architecture and structure.

In our previous works, ModelDrivenGuide [9, 10] transforms data models by proposing a set of data models providing choices instead of focusing on a dedicated solution. Then, it reduces the search space by taking into account the use case (set of queries).

This article proposes FACT-DM, a framework that integrates the input data model and its use case for a given input setting of execution environment, generates all data models and highlights subset of them proposed by our previous heuristic [9, 10]. Moreover, this framework introduces a multidimensional cost calculator used for sorting data models, based on time, financial and environmental costs. Providing a visualisation, FACT-DM eases decision-making for IS architects as existing solutions lack such tool.

2 FACT-DM FRAMEWORK

Our proposed framework includes 3 components: the Data Models Generator (DMG), the Multidimensional Cost Calculator (MCC) and the Visualization Tool (VT). Our approach ModelDrivenGuide in [9, 10] was used in the DMG component to generate all possible *logical* data models (Fig. 1), it also applies a heuristic to return a subset of data models adapted to the use case. This modeling approach is based on data models and refinement rules.

The MCC component then calculates the costs of data models using a multidimensional cost model. The latter assesses the performance, financial cost, and environmental impact of each data model to determine its overall cost in order to compare with others. The following sections explain each of these components.

2.1 Data Models Generator (DMG)

ModelDrivenGuide starts from the conceptual model, then goes from one logical data model M to another by applying refinement rules recursively [10]. Left part of Figure 1 shows the input conceptual model and settings. The refinement rules we use are: **Merge** (two concepts linked by a reference transformed into one concept), and **Split** (one concept into two new concepts).

In the following, denormalization of data models will refer to the combination of merge and split transformation rules. All denormalized data models are produced by recursively splitting and merging keys/rows of logical data models.

This generation process produces all possible data models but the number of possibilities explodes as splits on M can be applied on each key and merges are bidirectional.

Thus, the DMG component relies on a heuristic [9] to produce a list of denormalized data models $\mathcal{M}$, as it prevents duplicates

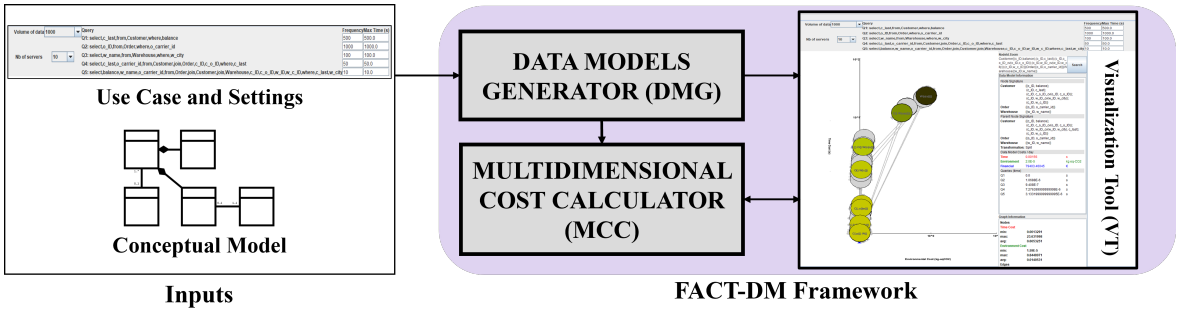


Figure 1: FACT-DM: A Framework for Data Model Transformation

and reduces the search space by prioritizing data models that align with the use case.

2.2 Multidimensional Cost Calculator (MCC)

Besides conventional measures like response time and throughput, NoSQL database systems demand higher requirements due to the massive amount of data they need to handle (which incurs significant costs) in terms of storage, processing and communication.

In order to choose the optimal data model out of the set of proposed possible ones $\mathcal{M}$, we suggest a cost model that automatically calculates the costs of logical data models to compare them.

Definition 2.1. Let $M \in \mathcal{M}$ a data model and $\mathcal{Q} = \{q_1,...,q_n\}$ be a set of queries from the use case. The multidimensional cost function C of M regarding $\mathcal{Q}$ is defined by:

$$C(M,\mathcal{Q}) = \begin{pmatrix} T(M,Q) \\ E(M,Q) \\ F(M,Q) \end{pmatrix} = \begin{pmatrix} T(M) \\ E(M) \\ F(M) \end{pmatrix} + \sum_{i=1}^{n} \omega_{q_i} \times \begin{pmatrix} T(M,q_i) \\ E(M,q_i) \\ F(M,q_i) \end{pmatrix}$$

where $\phi \in \{T, E, F\}$ are cost functions on the data model M either independent ($T(M), E(M), F(M)$) or dependent ($T(M,q_i), E(M,q_i)$,-$F(M,q_i)$) on queries $q_i \in \mathcal{Q}$, with their related query's average daily occurrences ω_{q_i}. T, E, F are sub-functions corresponding respectively to the time, environmental and financial dimensions of the cost model.

To achieve this choice, we need to measure costs with common parameters: data volumes $\#doc$ and servers $\#srv$. Each cost dimension relies on the volume of: stored data V_{SDD}, processed data on servers V_{RAM} and transferred data among servers V_{COM}. Moreover, the combination of these volumes varies based on the data model and queries computation.

The optimization and cost functions are detailed in [11].

2.2.1 Time Cost Dimension. The time cost of a data model can vary according to several factors, including the size and complexity of the data model, the used storage type and processing infrastructure, and the speed of the network. It is expressed in seconds (s).

It exploits both parallelism with sharding and local indexing with distribution which varies with different denormalizations.

Definition 2.2. Let $T(M,q)$ be the time cost of a query $q \in \mathcal{Q}$ on a data model $M \in \mathcal{M}$. Let $T(M)$ be the query independent time cost of the data model M. We denote:

$$\begin{aligned} T(M,q) &= \frac{V^T_{RAM}(M,q)}{C^T_{RAM}} + \frac{V_{SSD}(M,q)}{C^T_{SSD}} + \frac{V_{COM}(M,q)}{C^T_{COM}} \\ T(M) &= 0 \end{aligned}$$

$T(M,q)$ is calculated based on data volumes in Bytes and speed constants expressed in GB/s.

Notice that $T(M) = 0$ since the computation time is only dependent on queries.

2.2.2 Environmental Cost Dimension. Processing queries has an impact on the energy consumption. In fact, the environmental cost depends on data accesses like CPU, RAM, storage and communication. The cost calculator needs to quantify the amount of data processed for each query on each of the data models.

Measuring the exact environmental impact is an impossible task, especially as most studies focus on the global impact of systems[1] rather than detailed individual treatments. Our aim is to estimate consumption for comparing data models, and not obtaining a precise impact. The environmental E cost is expressed in kg CO2e.

Notice that the environmental footprint of a server is independent of queries where servers whole lifecycles are studied [6]. We can consider that, on average, a single server corresponds to 320 kg of CO2e/year (0.87671 kg CO2e/day). In our case, E is influenced by the number of servers required by a data model. Indeed, some data models contain more redundancy and require more RAM and disk space, hence the number of servers required.

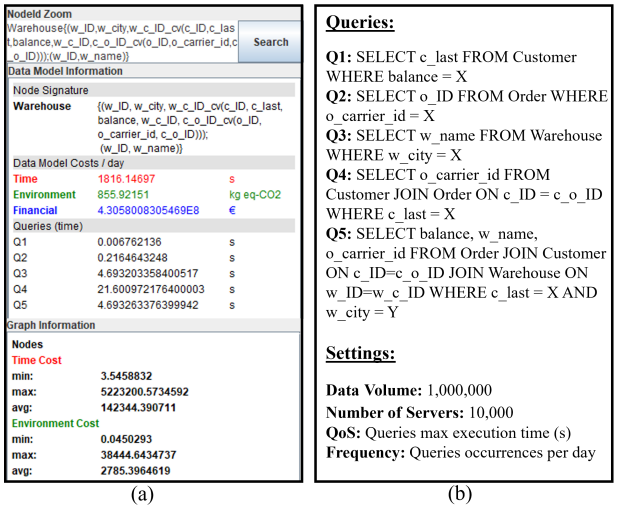
Definition 2.3. Let $E(M,q)$ be the environmental cost of a query $q \in \mathcal{Q}$ on a data model $M \in \mathcal{M}$. Let $E(M)$ be the independent environmental cost of the data model M. We denote:

$$\begin{aligned} E(M,q) &= V^E_{RAM}(M,q) \times C^E_{RAM} + V_{SSD}(M,q) \times C^E_{SSD} \\ &\quad + V_{COM}(M,q) \times C^E_{COM} \\ E(M) &= \#srv \times C^E_{srv} \end{aligned}$$

where $E(M)\&E(M,q)$ are expressed in kg CO2e.

2.2.3 Financial Cost Dimension. The financial cost of a data model has few dependency on query execution, since most of the expenses come from the number of servers $\#srv$ depending on the pricing model (*e.g.*, pay-as-you-go or subscription). However, most service providers have fees for data transfers outside of the datacenter[2]. Our cost model distinguishes internal from external communications. The financial cost F is expressed in currency (*e.g.*, €, $).

[1] Evaluating the Carbon Footprint of a Software Platform Hosted in the Cloud
[2] Azur bandwith pricing



(a) (b)

Figure 2: (a) Dashboard and (b) Use Case Queries and Settings

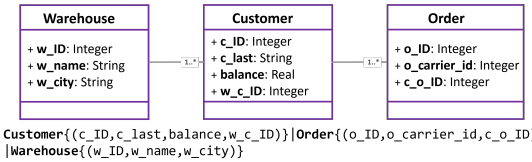


Figure 3: TPC-C Data Model as a Class Diagram and its Signature

Definition 2.4. Let $F(\mathcal{M})$ be the financial cost of a query $q \in Q$ on a data model $M \in \mathcal{M}$. Let $F(M)$ be the independent financial cost of the data model M. We denote:

$$F(M,q) = V^{ext}_{COM}(M,q) \times C^{F}_{COM}$$
$$F(M) = \#srv \times C^{F}_{srv}$$

where $F(M)$ and $F(M,q)$ are expressed in currency (*e.g.*, €, $).

2.3 Visualization Tool (VT)

Our framework is implemented in *Java*[3] and integrates a Visualization Tool which shows the entire process as presented in Figure 1.

2.3.1 Framework Inputs. The FACT-DM framework takes as inputs a conceptual model in the XMI format[4]. It also relies on a use case (*i.e.*, queries) applied on the schema accompanied with their frequencies and Quality of Service (QoS *i.e.*, max execution time per query). To tune the simulation, more inputs like the volume of data, the number of servers and queries' frequency are given.

2.3.2 Data Models Generation. Data models are transformed in the data model generator component and stored as signatures (see in Figure 3). Then, the cost calculator considers the use case to generate a multi-dimensional cost for each data model

2.3.3 Graph of Data Models. Data model signatures and costs are visualized in a dashboard as a graph of data models (right part of Fig. 1) by using graphstream[5]. A node is a data model and edges correspond to transformations (merges and splits).

To visualize costs, nodes' color uses the RGB (Red Green Blue) system, where each color corresponds to time, environmental, and financial costs, respectively. Darker nodes indicate data models with higher costs, while lighter-colored nodes represent more optimal data models. To showcase the reduction of the search space by our generation heuristic, avoided data models are depicted in gray. Furthermore, nodes outlined in red represent data models for which the QoS was not respected (*i.e.*, the maximum time allowed to execute a query).

2.3.4 Dashboard. An information panel is displayed as in Figure 2(a):

- *Search box*: it allows searching for nodes in the graph using their ID (*i.e.*, unique signature),
- *Data model information*: it is displayed by clicking on a node and includes the node's signature, time, environmental and financial costs, and queries time costs on this data model,
- *Graph information*: includes minimal, maximal and average costs among all graph's data models.

Additionally, the tool allows to interact with data models' costs by varying different parameters: data volume, number of servers, frequencies and QoS.

3 DEMONSTRATION

To demonstrate the framework, the user gives a conceptual model, its use case and settings. For this, we use the TPC-C[6] benchmark, focusing on an input data model with three rows: *Warehouse*, *Customer*, and *Order* depicted in Figure 3. We also apply the use case in Figure 2(b), mixing filter and join queries. The first scenario shows the generation of denormalized data models and the effect of the heuristic. Using our TPC-C example and use case, the DMG component generates naively 2,731 data models while only 36 data models are produced by our heuristic.

3.1 One Setting Scenario

The second scenario focuses on the interaction with the graph of data models produced during the first step. The 2,731 data models are displayed as a graph in a two dimensional chart that depicts environmental *vs.* time costs in logarithmic scales.

As shown in Figure 4(a), the graph includes all data models proposed by the DMG component in Section 2. The colored part of the graph is the sub-graph containing the 36 data models generated by our heuristic taking into consideration the use case. Grey nodes are the 2,695 data models that were not generated by the search space reducing heuristic. Some of these data models were redundant, others were not necessary for the use case queries. This reduction eases the selection of an optimal data model for the user.

We will also navigate on the graph from a node (*e.g.*, the initial data model) to another to see the impact of transformations on data models and see the path taken by the heuristic. The demonstration will show the corresponding costs and observe that optimal data models are among the 36 ones produced by the heuristic. The correlation between transformations and costs will demonstrate the benefits (or lack thereof) of denormalization.

[3]GitHub link of FACT-DM
[4]XML Metadata Interchange: OMG specifications
[5]GraphStream 2.0 - A Dynamic Graph Library
[6]TPC-C Benchmark

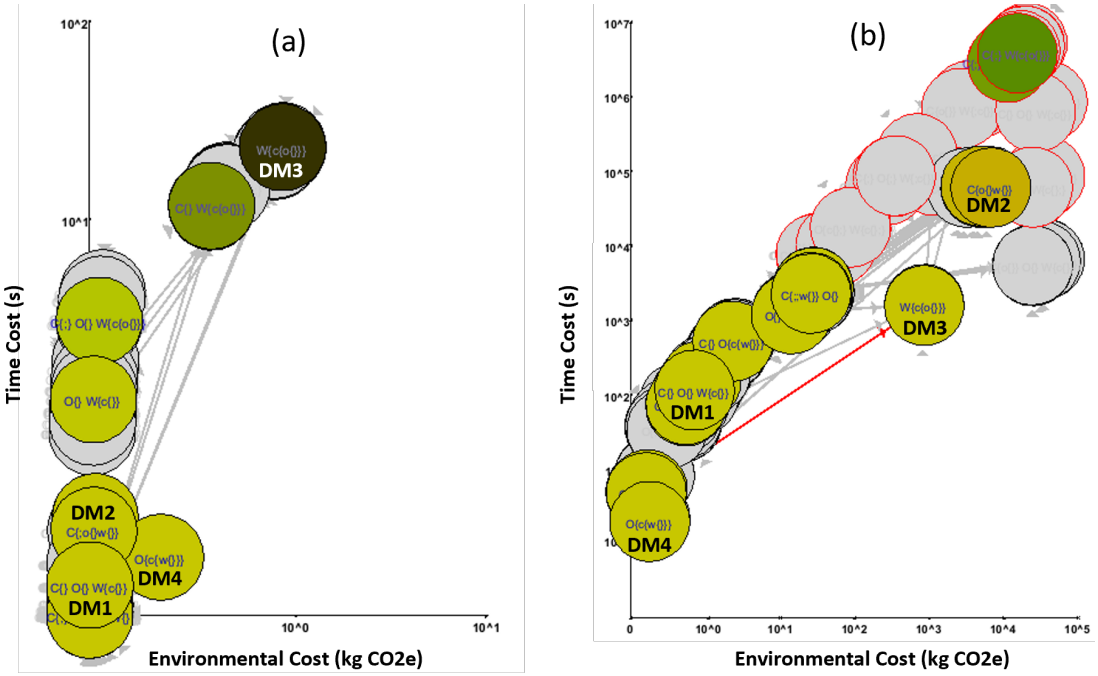


Figure 4: Display of Data Models *wrt.* their costs (a) #doc=1k warehouses, #srv=10 and (b) #doc=1M warehouses and #srv=10k

3.2 Settings Variation Scenario

The third scenario will tune the settings of simulation. We will interact with the costs by changing data volume, number of servers and QoS thresholds. The two figures (a) and (b) in Figure 4 depict two instances of settings with graphs of the generated data models. The corresponding costs on two different settings ($\#doc = 1,000$, $\#srv = 10$) and ($\#doc = 1M$, $\#srv = 10,000$) demonstrate the impact of settings on data models.

Initially, we observe that as the data volume increases, costs increase from Figure 4(a) to 4(b), and nodes show increased separation, indicating a greater variation in data models costs within larger settings. Furthermore, we observe that data models DM1 and DM2, which have relatively lower costs in figure (a), fall into a more expensive range in figure 4(b). In contrast, data model DM3, which had the highest costs in the small setting, proves to be more optimal in the larger setting. On the other hand, data model DM4 maintains a nearly consistent ranking among other data models in both settings. This consistency suggests that this data model may potentially be optimal in the long run.

Additionally, we notice in Figure 4(b) a larger number of nodes are outlined in red. These data models have time costs that do not adhere to the QoS thresholds specified by the user, leading to their disqualification.

4 CONCLUSION

This paper proposed FACT-DM, a framework to generate various data models using denormalization while taking into consideration a given use case. In order to compare these data models and determine the optimal one(s), our multidimensional cost model integrates time, environmental and financial dimensions to define the cost of each data model and to rank them in order to lead the choice. In this framework, we propose a visualization tool, that implements our entire process and allows to compare different data models.

REFERENCES

[1] Fatma Abdelhedi, Amal Ait Brahim, Faten Atigui, and Gilles Zurfluh. 2017. MDA-based Approach for NoSQL Databases Modelling. In DAWAK'17. 88–102.

[2] Artem Chebotko, Andrey Kashlev, and Shiyong Lu. 2015. A Big Data Modeling Methodology for Apache Cassandra. In ICBD'15. IEEE, 238–245.

[3] Gwendal Daniel, Gerson Sunyé, and Jordi Cabot. 2016. UMLtoGraphDB: mapping conceptual schemas to graph databases. In ER'16. 430–444.

[4] Myller Claudino de Freitas, Damires Yluska Souza, and Ana Salgado. 2016. Conceptual Mappings to Convert Relational into NoSQL Databases. In ICEIS'16. 174–181.

[5] Alfonso de la Vega, Diego García-Saiz, Carlos Blanco, Marta Zorrilla, and Pablo Sánchez. 2020. Mortadelo: Automatic generation of NoSQL stores from platform-independent data models. J. Future 105 (2020), 455–474.

[6] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. 2022. Chasing Carbon: The Elusive Environmental Footprint of Computing. IEEE Micro 42, 4 (jul 2022), 37–47.

[7] Andrea Hillenbrand, Uta Störl, Shamil Nabiyev, and Meike Klettke. 2022. Self-adapting data migration in the context of schema evolution in NoSQL databases. Distributed Parallel Databases 40, 1 (2022), 5–25.

[8] Yan Li, Ping Gu, and Chao Zhang. 2014. Transforming UML class diagrams into HBase based on meta-model. In ISEEE'14, Vol. 2. 720–724.

[9] Jihane Mali, Shohreh Ahvar, Faten Atigui, Ahmed Azough, and Nicolas Travers. 2022. A Global Model-Driven Denormalization Approach for Schema Migration. In RCIS'22. Springer, 529–545.

[10] Jihane Mali, Faten Atigui, Ahmed Azough, and Nicolas Travers. 2020. ModelDrivenGuide: An Approach for Implementing NoSQL Schemas. In DEXA'20. Springer, 141–151.

[11] Jihane Mali, Faten Atigui, Ahmed Azough, Nicolas Travers, and Shohreh Ahvar. 2023. How to Optimize the Environmental Impact of Transformed NoSQL Schemas through a Multidimensional Cost Model? arXiv preprint arXiv:2311.15406 (2023).

[12] M Tamer Özsu and Patrick Valduriez. 1999. Principles of distributed database systems. Vol. 2. Springer.

[13] Leonardo Rocha, Fernando Vale, Elder Cirilo, Dárlinton Barbosa, and Fernando Mourão. 2015. A framework for migrating relational datasets to NoSQL. Procedia Computer Science 51 (2015), 2593–2602.

[14] Uta Störl, Meike Klettke, and Stefanie Scherzinger. 2020. NoSQL Schema Evolution and Data Migration: State-of-the-Art and Opportunities. In EDBT'20. 655–658.

[15] Clarence JM Tauro, Shreeharsha Aravindh, and AB Shreeharsha. 2012. Comparative study of the new generation, agile, scalable, high performance NoSQL databases. International Journal of Computer Applications 48, 20 (2012), 1–4.

[16] Tamás Vajk, Péter Fehér, Krisztián Fekete, and Hassan Charaf. 2013. Denormalizing data into schema-free databases. In CogInfoCom'13. IEEE, 747–752.