Practice works & Deliverables

Big Data Structure

A5

ESILV
nicolas.travers (at) devinci.fr

Chapter Contents

Chapter 1

Project Settings

1.1 Goal

The goal of this project is to let you learn and experiment the impact of Big Data Infrastructure on the Cloud. We will study the question of Data Modeling in the NoSQL databases in order to make it:

- Scalable,
- Sustainable,
- Financially optimal.

This design of the database relies on several dimensions:

- A schema. Here a UML class diagram and statistics,
- A use case. Here a set of queries and related usage frequencies,
- A Cloud cluster setting. Servers, sharding & indexing strategies.

Thus, the purpose of this project is to simulate the cost of a NoSQL data model on the cloud. This simulation will be applied on different data model solutions which for the end-user will help to determine the **more suitable solution**.

Some formalism of this problem and solutions here (both from academic and industrial contributions):

- Data modeling for NoSQL [Chebotko et al., 2015, Abdelhedi et al., 2017, Gallinucci et al., 2018] [Mali et al., 2020, Störl et al., 2020, Mali et al., 2022, Carey et al., 2025, Belussi and Migliorini, 2026]
- Distribution cost optimization [Forresi et al., 2023, Shirazi et al., 2025] [Gallinucci et al., 2025, Baeuerle et al., 2025, Mali et al., 2026]

1.2 Global Instructions & LLM usage

The project requires to develop programs to compute the simulation. The usage of LLMs is not forbidden. However, we ask you to understand perfectly your source code. Your grade will depend more on the questions we will ask you than the code itself. **We will be uncompromising with evidence of blind trust in the source code produced by the LLM.**[1]

The choice of the programming language is free.

For each session, you will have to produce a package that will be plugged with other ones all along the project. The source code has to be uploaded on the Moodle in order that we will review your code to prepare some questions during the following session.

1.3 Schedule

1.3.1 Data models representation, JSON Schema and impact of the denormalization on volume

1.3.2 Impact of sharding strategies on data

1.3.3 Simulate the computation of a filter query. The simulation will compute both time and environmental costs

1.3.4 Simulate the computation of a join query

1.3.5 Simulate the computation of an aggregate query

1.3.6 Simulate the computation of the use case and data models choice.

[1] Notice that no complain will be accepted. You have to understand and explain what you produced.

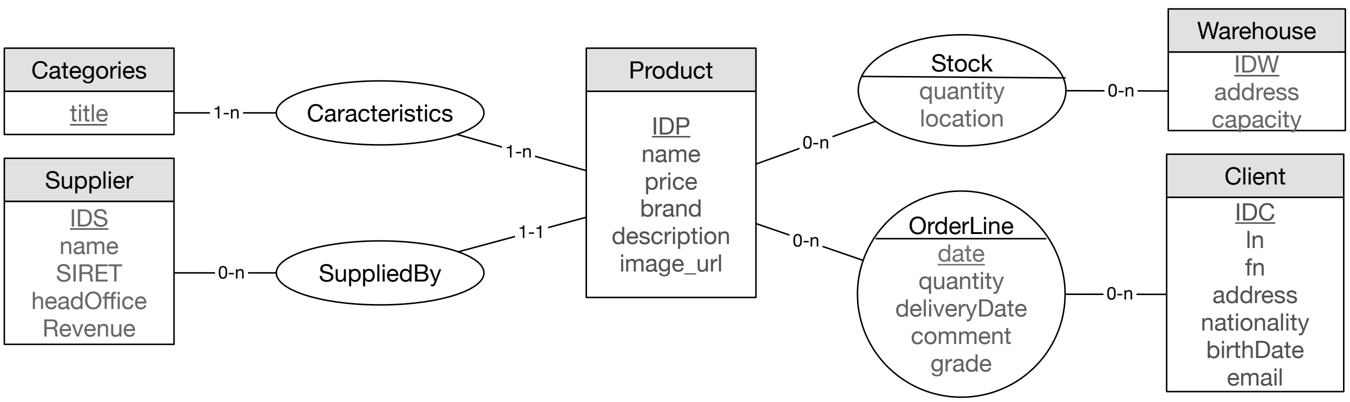
Chapter 2

Data Model Denormalization

This chapter is structured to help understand the issues related to transitioning from a relational database to a NoSQL database. Given a highly distributed context, we will face the traditional join problems of relational databases.

2.1 Example of an Entity-Relationship Model

In this chapter (and the followings), we will use a database managing product sales to customers over the internet. The Entity/Relationship diagram is as follows:



2.2 Statistics

Additionally, we have some statistics associated that will serve as a basis for calculations:

- 10^7 customers/clients, 10^5 products, 4.10^9 order lines, 200 warehouses;
- An order is compose of several order lines where: 1 orderline = 1 client & 1 product type;
- For a stock, even if the quantity of the product is zero, there is still an instance in “stock” (the value “0” is stored for the product);
- A customer makes on average 100 orders, involving 20 different products;
- 1 to 5 categories per product (on average 2);
- 5,000 distinct brands among products;
- 50 products of the “Apple” brand, distributed across all warehouses;
- Order lines are balanced over a year of orders (365 dates);
- Data distributed across 1,000 servers.

2.3 Relational to JSON Documents

JSON (JavaScript Object Notation) is a widely used semi-structured data representation model on the Web. It is particularly used in document-oriented NoSQL systems.

In this section, we will study the transformation (Denormalization) of a relational schema into JSON documents. We will take the table `Product` as input.

2.3.1 Provide a JSON document example on the **Product** collection and merge "Categories" and "supplier", where:

- For the `Product` entity, provide the direct correspondence in JSON in the form of document examples (we do not have yet a JSON schema);
- The price must be detailed with a currency and a VAT rate;
- The categories are independent from each other;

2.3.2 Produce the corresponding JSON Schema (all keys are mandatory).
https://www.liquid-technologies.com/online-json-to-schema-converter

2.4 Denormalization

To avoid the cost of network communications during join queries, some other merges can be applied. However, several solutions can be applied. A denormalization step is necessary.

In order to simplify the understanding of the constitution of a database "**signatures**" will help the notation. Example:

Prod{[Cat],Supp}, St, Wa, OL, Cl

- This list corresponds to all collections stored in this DB. If no keys is specified, by default all keys are linked to their concept;
- Braces {} are used to represent concept nesting (if several, separated by commas);
- Brackets [] are used to represent nested arrays;
- Shortcut namings:
  - **Prod**: Product
  - **Cat**: Categories
  - **Supp**: Supplier
  - **St**: Stock
  - **Wa**: Warehouse
  - **OL**: OrderLine
  - **Cl**: Client

Produce the JSON Schemas for the corresponding denormalizations of those DB signatures:

2.4.1 **DB1**: Prod{[Cat],Supp}, St, Wa, OL, Cl

2.4.2 **DB2**: Prod{[Cat],Supp, [**St**]}, Wa, OL, Cl

2.4.3 **DB3**: **St**{Prod{[Cat],Supp}}, Wa, OL, Cl

2.4.4 **DB4**: St, Wa, **OL**{Prod{[Cat],Supp}}, Cl

2.4.5 **DB5**: Prod{[Cat],Supp, [**OL**]}, St, Wa, Cl

2.5 Database Size

2.5.1 Produce for each collection type, its document size in Bytes with the following keys size (approximations).

- Integer/Number : 8B
- String : 80B
- Date : 20B (a specific string)
- LongString : 200B

- Key+Value pairs/Arrays : 12B + values

2.5.2 Give for each collection, its size in GB;

2.5.3 Give for each database (DB1-DB5), its size in GB.

2.5.4 What are the problems related to those denormalizations?

2.6 Sharding Strategies

Documents from each collection will be stored in a cluster of servers (here 1,000) using sharding. This exercise aims at explaining how documents will be distributed among them.

A sharding strategy for a given collection is defined by the sharding key denoted by "CollName - #key".

2.6.1 For each sharding strategy below, give the average number of documents per server & the average number of distinct values (for the sharding key) per server:

- St - #IDP
- St - #IDW
- OL - #IDC
- OL - #IDP
- Prod - #IDP
- Prod - #brand

2.7 HOMEWORK (2 days before next session)

Submit a program (package) on DVL which automatizes the previous steps:

- Take a JSON Schema and set of statistics as inputs;
- Generate a structure (object) to be manipulated in future programs (calling this package);
- Functions to compute size of: documents, collections and database;
- Functions to compute the statistics of documents distribution with given sharding keys on a given collection.

Test your program with the JSON schemas provided during the practice work.

You will be evaluated on your knowledge about the program you have developed during the next session. You have to be able to explain it.

Chapter 3

Filter and Join Queries

Since we have stored data in a distributed NoSQL database, we can now query it to find data. For this, we wish to estimate the cost of queries in terms of: Time, Carbon footprint, & Price.

For each query, give:

Database	**Sharding keys**	**Index**	**Algo**	**Costs** time	carbon footprint	price
DB1 - DB5	Involved sharding key(s)	Implied/Required index locally on each server	Nested Loop & Shard / Index / Full scan			

3.1 Filter Queries

Here are two different filter queries we wish to apply on the database. They are written in SQL but have to be adapted to the corresponding NoSQL querying language (MQL, CQL, etc.).

Q1 The stock of a given ID product in a given warehouse:

```
SELECT S.quantity, S.location
FROM Stock S
WHERE S.IDP = $IDP AND S.IDW = $IDW;
```

Q2 Names and prices of product from a given brand (take "Apple" as example):

```
SELECT P.name, P.price
FROM Product P
WHERE P.brand = $brand;
```

Q3 Product ID and quantity from order lines ordered at a given date:

```
SELECT O.IDP, O.quantity
FROM OrderLine O
WHERE O.date = $date;
```

3.2 Join Queries (with filters)

Here are two join queries, with filters (combined with previous queries).

Q4 Stock (list of product names, as well as their quantity) from a given warehouse;

```
SELECT P.name, S.quantity
FROM Stock S JOIN Product P ON S.IDP = P.IDP
WHERE S.IDW = $IDW;
```

Q5 Distribution of "Apple" brand products (name & price) in warehouses (IDW & quantity);

```
SELECT P.name, P.price, S.IDW, S.quantity
FROM Product P JOIN Stock S ON P.IDP = S.IDP
WHERE P.brand = "Apple";
```

3.3 HOMEWORK (2 days before next session)

Submit a package with several *operators* on DVL which automatizes the previous steps:

- Operators:
  - Filter with sharding;
  - Filter without sharding;
  - Nested loop with sharding;
  - Nested loop without sharding.
- Each takes as input:
  - The targeted collection (link with the structure/object defined in the previous program / JSON Schema);
  - The expected output format (involved keys);
  - The filtered key;
    Optionally, the filter selectivity could be provided (by default, take the one computed in Section **??**).
- Provides in output:
  - The number of output documents (and corresponding size);
  - The costs.
- It should be recommended to provide functions that compute different type of costs which estimates the volume of data, etc.
- *Not required*: a program that translates a query into operators and parameters (it can be a bonus);

Test your program with the filter & join queries provided during the practice work (and the previous JSON Schema).

As usual, you will be evaluated on your knowledge about the program you have developed during the next session. You have to be able to explain it.

Chapter 4

Aggregate Queries

For each query, give:

Database	Sharding keys	Index	Algo	Costs		
				time	carbon footprint	price
DB1 - DB5	Involved sharding key(s)	Implied/Required index locally on each server	Map/Reduce & Nested Loop & Shard / Index / Full scan			

4.1 Aggregate Queries

Q6 The 100 most ordered product names and price (sum of quantities).

```sql
SELECT P.name, P.price, OL.NB
FROM Product P JOIN (
   SELECT O.IDP, SUM(O.quantity) AS NB
   FROM OrderLine O
   GROUP BY O.IDP
) OL ON P.IDP = C.IDP
ORDER BY OL.NB DESC
LIMIT 1;
```

Q7 Name and price of the product most ordered by customer no. 125;

```sql
SELECT P.name, P.price, OL.NB
FROM Product P JOIN (
   SELECT O.IDP, SUM(O.quantity) AS NB
   FROM OrderLine O
   WHERE O.idClient = 125
   GROUP BY C.IDP
) OL ON P.IDP = OL.IDP
ORDER BY OL.NB DESC
LIMIT 1;
```

4.2 HOMEWORK (2 days before next session)

Submit a package with the aggregate *operators* on DVL which automatizes the previous steps:

- Operator:
  - aggregate with sharding;
  - aggregate without sharding.
- Each takes as input:
  - The targeted collection (link with the structure/object defined in the program / JSON Schema in Section 2.7);
  - The grouping key(s) (link with statistics is necessary);
  - The expected output format (involved keys);
  - An optional filtered key.

- Provides in output:
  - The number of output documents / distinct key values (and size);
  - The costs.
- It should be recommended to provide functions that compute different type of costs which estimates the volume of data, shuffle, reduce, etc.

Test your program with the aggregate queries provided during the practice work (and the previous JSON Schema).

As usual, you will be evaluated on your knowledge about the program you have developed during the next session. You have to be able to explain it.

Chapter 5

The Data Model Selection's Challenge

The goal of this last session is to challenge your program by:

- Plug all operators for complex queries;
- Test it with a new schema with corresponding queries and statistics;
- Translate the queries into a sequence of operators;
- Compute queries' costs;
- Test different denormalizations to find the less costly and compete with other groups!

Chapter Bibliography

[Abdelhedi et al., 2017] Abdelhedi, F., Ait Brahim, A., Atigui, F., and Zurfluh, G. (2017). MDA-Based Approach for NoSQL Databases Modelling. In Bellatreche, L. and Chakravarthy, S., editors, *Big Data Analytics and Knowledge Discovery*, pages 88–102, Cham. Springer International Publishing.

[Baeuerle et al., 2025] Baeuerle, M., Bodner, T., Boissier, M., Rabl, T., Díaz, R. S., Schmeller, F., Strassenburg, N., Tolovski, I., Weisgut, M., and Yue, W. (2025). Tco2: Analyzing the carbon footprint of database server replacements. *Proc. VLDB Endow.*, 18(12):5223–5226.

[Belussi and Migliorini, 2026] Belussi, A. and Migliorini, S. (2026). From er conceptual models to document-based nosql logical models. In Chrysanthis, P. K., Nørvåg, K., Stefanidis, K., Zhang, Z., Quintarelli, E., and Zumpano, E., editors, *New Trends in Database and Information Systems*, pages 56–66, Cham. Springer Nature Switzerland.

[Carey et al., 2025] Carey, M. J., Alkowaileet, W. Y., Digeronimo, N., Gupta, P., Smotra, S., and Westmann, T. (2025). Towards principled, practical document database design. *Proc. VLDB Endow.*, 18(12):4804–4816.

[Chebotko et al., 2015] Chebotko, A., Kashlev, A., and Lu, S. (2015). A Big Data Modeling Methodology for Apache Cassandra. In *2015 IEEE International Congress on Big Data*, pages 238–245, Santa Clara, CA, USA. IEEE.

[Forresi et al., 2023] Forresi, C., Francia, M., Gallinucci, E., and Golfarelli, M. (2023). Cost-based Optimization of Multistore Query Plans. *Information Systems Frontiers*, 25(5):1925–1951.

[Gallinucci et al., 2025] Gallinucci, E., Golfarelli, M., Radwan, W., Zarate, G., and Abelló, A. (2025). Impact study of nosql refactoring in skyserver database. In Maté, A. and Lissandrini, M., editors, *Proceedings of the 27th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP 2025) co-located with the 28th International Conference on Extending Database Technology and the 28th International Conference on Database Theory (EDBT/ICDT 2025), Barcelona, Spain, March 25, 2025*, volume 3931 of *CEUR Workshop Proceedings*, pages 1–11. CEUR-WS.org.

[Gallinucci et al., 2018] Gallinucci, E., Golfarelli, M., and Rizzi, S. (2018). Schema profiling of document-oriented databases. *Information Systems*, 75:13–25.

[Mali et al., 2022] Mali, J., Ahvar, S., Atigui, F., Azough, A., and Travers, N. (2022). A Global Model-Driven Denormalization Approach for Schema Migration. In *Research Challenges in Information Science*, pages 529–545, Cham. Springer International Publishing.

[Mali et al., 2026] Mali, J., Ahvar, S., Atigui, F., Azough, A., and Travers, N. (2026). Damoop: A global approach for optimizing denormalized schemas through a multidimensional cost model. *Information Systems*, 136:102598.

[Mali et al., 2020] Mali, J., Atigui, F., Azough, A., and Travers, N. (2020). ModelDrivenGuide: An Approach for Implementing NoSQL Schemas. In *Database and Expert Systems Applications: 31st International Conference*, DEXA'20, page 141–151, Bratislava, Slovakia. Springer-Verlag.

[Shirazi et al., 2025] Shirazi, S. H. A., Wang, X., Carey, M. J., and Tsotras, V. J. (2025). Optimizing big active data management systems. In Maté, A. and Lissandrini, M., editors, *Proceedings of the 27th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP 2025) co-located with the 28th International Conference on Extending Database Technology and the 28th International Conference on Database Theory (EDBT/ICDT 2025), Barcelona, Spain, March 25, 2025*, volume 3931 of *CEUR Workshop Proceedings*, pages 39–48. CEUR-WS.org.

[Störl et al., 2020] Störl, U., Klettke, M., and Scherzinger, S. (2020). Nosql schema evolution and data migration: State-of-the-art and opportunities. In Bonifati, A., Zhou, Y., Salles, M. A. V., Böhm, A., Olteanu, D., Fletcher, G. H. L., Khan, A., and Yang, B., editors, *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT*, pages 655–658, Copenhagen, Denmark. OpenProceedings.org.