Embracing the Friction: Towards a computationally aware approach to humanistic data interfaces

by

Anna Sollazzo

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Arts

Digital Humanities
University of Alberta

Abstract

Inherent to interdisciplinary work is the negotiation of two or more sets of—often contradictory—domain epistemologies and methodologies. In the context of the Digital Humanities, the friction between its composite domains is particularly strong with respect to data processing and display, where the ambiguity, complexity, and nuance that characterise humanities data stand in opposition to the binary and discrete representations required by computationally compatible encodings. Digital Humanities data interfaces have historically submitted to the simplification and categorisation imposed by prevalent forms of scientific visualisation, and previous work imagining interfaces better suited to humanistic inquiry has largely focused on augmenting these standards. In this thesis I explore the potential of instead stripping away layers of abstraction to expose, centre, and challenge the principles and assumptions underpinning them. This work is built around a proof of concept of interface in this alternative style that is modelled after natural language database query systems, but which atypically exposes each step and difficulty of the question to query translation. I argue that an approach that forefronts human to machine conceptual mapping both addresses key humanistic concerns such data trajectory and user positionality, and also has an important capacity to engage with questions of computational possibility and limits. The theoretical grounding implied in the latter is essential to the elaboration of novel and critically conscious technologies that embody humanistic principles.

For my grandfather, John Hvozdanski, who earned his Master's in Civil Engineering from the University of Alberta in 1962.

I couldn't imagine a better person in whose footsteps to follow.

(Sort of. You'll have to forgive me for not mentioning silt in this thesis. Not even once.)

Thank you for raising me to always be curious.

Acknowledgements

When I asked you to be my supervisor, you responded to my email in five minutes flat and called the decision easy. I was baffled.

Thank you.
Mille fois merci.
I'm running out of languages.
System.out.println("Thank you!")?

I don't have the right words (or code), but I want to first extend my deepest gratitude to my advisor Sean Gouglas. For all you insist you're just doing your job, I have never seen anyone work harder to support and advocate for their students in so many ways. I count myself exceptionally lucky to have been one of them. Thank you for your careful mentorship, impressive repertoire of (b|d)ad jokes, and boundless kindness.

Thank you to the members of my committee. To Geoffrey Rockwell, for your generosity with your time and expertise, and for your thought provoking comments. To Harvey Quamen, for being one of the main reasons I ended up in the DH program in the first place.

À la merveilleuse Sara Harvey - je vous dois tellement que je ne saurais même pas par où commencer. Merci d'avoir noté les autocollants sur mon ordinateur.

Je dois aussi une fière chandelle à tous et toutes les membres de l'équipe du Programme des Registres de la Comédie-Française. Merci de m'avoir accueilli si chaleureusement au projet, de votre patience à l'égard de la petite informaticienne anglophone qui vous pose de drôles de questions et surtout de votre curiosité et générosité intellectuelle. Votre expertise m'a été indispensable.

I may only be 90% sure that some of you even exist below the shoulders, but I am 100% in awe of the kind hearts and generous spirits of my incredible Digital Humanities cohort. Particular thanks to Cate Alexander, for making me feel welcome in DH from day one; to Kenzie Gordon, for somehow always having the answer but also making it seem ok not to know anything at all; to Julia Guy, for being such a constant and supportive presence in the early stages of this project; and to Schyler Palm, for your insightful feedback and heartfelt encouragement.

Nicola DiNicola—I can't imagine what the DH program would be without you. Thank you for all that you do. I have a feeling I don't even know the half of it.

Since I'm fairly convinced that your tenth grade English class is the main reason I know how to write even semi-coherently, a special thank you to Chris McDonald.

To the VSHD crew and the Bon Accord & overs—thank you for keeping me laughing and leaping.

Last but not least, thank you to my family.

Table of Contents

List of Figures

Introduction

Hippolyte Clairon made waves when she tread the boards at the Comédie-Française (CF) in the mid-eighteenth century. Praised in the periodical press more than any other actor of her, or any previous, generation,[1] she was known for her precision and finesse.[2] Clairon was an anomaly. Unlike her fellow actors, notably her mentor turned rival, Dumesnil, Clairon insisted that the best most natural performances were those that were the product of extensive study of political, social, and cultural context,[3] as opposed to reliance on an internal instinctual understanding inherent to "the organisation of mankind, creation of god."[4]

The latter vision of the world was more common to eighteenth century knowledge production. In the first volume of Diderot and d'Alembert's *Encyclopédie ou Dictionnaire Raisonné des Sciences, des Arts et des Métiers*, the authors affirm that the aim of the work is to "expose the order and sequence of human knowledge".[5] *Expose*, not actively create, organise, or define. This is the intellectual context that forged William Playfair (1759-1823), Scottish engineer and statistician widely regarded as the 'father of modern graphical methods'. Playfair's absolute emphasis on simplicity and completeness in turn directly and significantly influenced modern visualisation icon Edward Tufte, who insists that the role of visualisation is to *reveal* an inherent truth present in quantitative data[6].

As Lauren Klein points out, though humanists have begun to explore the oxymoron that is 'raw data', discover the affective potential of chartjunk, and poke at the mismatch between the epistemology of humanistic scholarship and the dominant forms of information processing and visualisation thrust upon them by the Playfair-Tufte lineage, little work has been done to

[1] Based on actor mentions across 11 periodicals spanning from 1714-1791.
[2] Diderot, Denis. *Paradoxe sur le comédien: ouvrage posthume* (A. Sautelet, 1830).
[3] Clairon, Hippolyte *Mémoires d'Hyppolite Clairon, et réflexions sur l'art dramatique* (F. Buisson: Paris, 1799)
[4] Dumesnil, Marie-Françoise. *Mémoires de Mlle Dumesnil, en réponse aux mémoires d'Hippolyte Clairon* (Ponthieu: Paris, 1823), my translation.
[5] Diderot, Denis, and Jean Le Rond d'Alembert. *Encyclopédie, ou, Dictionnaire raisonné des sciences, des arts et des métiers*. vol. 1. (Pergamon Press, 1776): 1, my translation
[6] Klein, Lauren. "What Data Visualization Reveals: Elizabeth Palmer Peabody and the Work of Knowledge Production." (2022).

re-examine and call into question the history of visualisation on a larger scale. Klein's own work is frequently focused on using data to identify and elevate hidden figures, voices at the margins who are erased by aggregative, generalising, and, in all frankness, patriarchal histories. As a historical counterpoint and "alternative epistemological lodestone" to Playfair, she offers Elizabeth Palmer Peabody (1804-1894), American transcendentalist and educator whose seminal work in visualisation was the creation of the 'Polish-American' system of representing chronological history using a grid of subdivided colour and pattern coded squares. Where Playfair intended his graphics to provide "immediate insight" into a definitive truth, Peabody intended for her charts to be "abstract rather than intuitive" so as to "promote sustained reflection". While he designed for "men of high rank", she worked primarily with children. Where he envisioned his graphics as authoritative, she emphasised participatory learning that flattened the hierarchy between the creator and the viewer.[7]

The advent of the digital has undeniably changed the way in which we interact with information and, like visualisation, seldom are its origins and underlying epistemology scrutinised. Rather, much as Playfair intended his work to be unquestionably 'accurate', computing has long promoted a similar association with objectivity. This is perhaps not surprising given that barely two decades separate Playfair's most prolific period and Charles Babbage's development of the difference engine, widely considered to be the first mechanical computer. The Digital Humanities (DH) find themselves the inheritors of this complex and compound intellectual legacy. My work looks to continue in line with Klein's critical examination of visualisation and knowledge production, but in a DH context. I don't believe it is necessarily useful to position them as diametrically opposed, but there are undeniable methodological and epistemological tensions between computing and the humanities. This thesis adopts the context of digital data interfaces to deconstruct that dynamic and look at how it might be productively navigated. It more broadly considers how we might imagine tools or processes reflective of a more foundationally entwined and reciprocally critical blend of DH's constituent domains.

Though unquestionably modern, this exploration will nonetheless call back to the origins of data visualisation.The project's data context is anchored in Playfair's time—in Clairon's world, more

[7] ibid

specifically. The Comédie-Française, the theatre where Clairon made her career, kept detailed registers of daily revenue, expenses, programming and casting from its foundation in 1680 through to the revolutionary period in 1793. These records were legally tied to the company's royal patronage and altogether coincide with the rapid development of population statistics, primarily for state use, in the 18th century. This archive is an economic, cultural, and political dataset all in one, particularly notable for its completeness and consistency with regard to its period. I find it especially compelling as a DH dataset because it embodies the conflict between the enlightenment era elevation of quantification, whose legacy undeniable influenced the development of modern computation, and the affective, subjective, embodied, and transient aspects of humanities study, particularly in the areas of performance and cultural heritage.

Recognizing the influence and naturalisation of beliefs and conventions enshrined in a long history, Johanna Drucker argues that "to overturn the assumptions that structure conventions acquired from other domains requires that we re-examine the intellectual foundations of digital humanities." This deconstruction and re-evaluation is the purpose of chapter one. Klein describes Peabody's work as "rethinking…the multiple ways in which we understand the nature of knowledge itself", and, predictably, there are strong parallels between the foci of her analog deconstruction and re-interpretation, and those of investigations carried out in the digital humanities. Peabody's emphasis on constructionism and the "generative potential of aesthetic judgement" dovetails with the emphasis DH scholars—notably Drucker—have come to place on the constructed nature of data. Her ambition to "provoke a unique imaginative response in each viewer" in view of reflexive learning further aligns with the humanistic re-imagining as of the viewer of an interface as a 'subject'—an agent who entertains a bidirectional relationship with the object of information with which they are interacting.[8] In an overview of previous theorisations and instantiations of DH interfaces, the first chapter of this thesis highlights the notion of interface transparency that stems from questions of data ambiguity and transformation, alongside generativity and interpretability as common core facets of humanistic information processing, display, and democratisation. It further details the CF dataset and analyses four tools built around it with respect to those three core considerations of humanistic data interfaces.

[8] Drucker, "Humanities approaches to interface theory."

Peabody's system for encoding historical events was not exactly evident to understand, something which its creator regarded as "both a liability and the point."[9] The same critiques of complexity and illegibility are often leveled at the computational representations of information that underlie digital interfaces. However, rather than facing them, there is a tendency to bury these models beneath levels of abstraction. I have consistently been struck by the imbalance that exists in the digital humanities between its two composite domains with respect to theory. The use of digital tools is widespread but rarely are the underlying computational logics that influence and define the what and how of digital inquiry afforded the same degree of scrutiny as the subjects to which they are applied. Chapter two presents CLAIRON, a proof of concept for a style of humanistic interface based around redressing this imbalance. CLAIRON is a derivative of the class of natural language database interfaces but deviates from convention by exposing every step of the process of transforming a user's question into a machine interpretable form, and further allows them to take an active role in the procedure. 'The point', as it was in Peabody's case, is to engage the user in critical reflection around knowledge production and create space for them to induce challenges and limitations.

Much as Peabody is overshadowed by the 'father of modern graphical methods", the dominating figure of Babbage, the 'father of computing', obscures the nuanced aspects of the work of Ada Lovelace. Despite all efforts by her mother to distance Ada from the artistic leanings of her infamous poet father, she nonetheless imagined possibilities for Babbage's analytical engine beyond the number crunching for which it was conceived. "It might act upon other things besides *number*" wrote Lovelace, "were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine."[10] As Lovelace speculated on the potentiality of computing, the focus was not on the result, but on the act of re-examining and deconstructing the structures and processes constitutive of our understanding of the world to imagine how they might be transformed to be made machine readable. Given that "humans w[ould] always supply the creativity and intentionality",[11] it would seem that, for Ada

[9] Klein, "What Data Visualization Reveals"
[10] Essinger, James. *Ada's algorithm: How Lord Byron's daughter Ada Lovelace launched the digital age*. (Melville House, 2014).
[11] Isaacson, Walter. "The Intersection of the Humanities and the Sciences." *43rd Jefferson Lecture in the Humanities, National Endowment for the Humanities* (2014).

Lovelace, the foremost interest of computing was as, not unlike Peabody's system, a "framework … through which … difficult thinking could take place."[12] This focus, I argue in chapter three, must be emphasised in humanities contexts, where problems whose nuance and complexity jut up against the systematicity and regularity required of machine encodings constantly push at the limits of computational possibility. The final chapter explores how this theoretically grounded and process-oriented approach to computation is precisely what is needed in the digital humanities, despite it currently being conspicuously ignored. There, I speculate about how tools like CLAIRON might aid in ushering in a new wave of digital scholarship that has a deeper understanding of the influence of digital mediation on knowledge production, and theorise the kind of pedagogical changes needed to facilitate this shift.

Taking up the mantle from Lovelace, Grace Hopper always asserted that "A human must turn information into intelligence or knowledge. We've tended to forget that no computer will ever ask a new question."[13] Her work correspondingly focused on making computing more accessible to humans by creating mechanisms for interfacing with machines via more abstract descriptions of processes and relationships. Her development of the A-0 proto-compiler, and contributions to the first English-like programming language, COBOL, fundamentally changed how we interact with computers today. Yet, she initially faced backlash over the crisis of the de-mathematisation of computing.[14] At around the same time as Hopper was at Eckert-Mauchly fighting to develop the human side of computation, Roberto Busa was over at IBM undertaking work in the computational side of the humanities that would birth a field that would eventually face a comparable mirror criticism over the supposed disregard for theoretical rigour. I don't think that Hopper would not have tolerated Busa or Thomas Watson Sr. for long—she notoriously refused to work for IBM because of the stifling corporate environment complete with a company flag and songs about it[15]—but I can't help but wonder what she would have thought of the project in the abstract. Hopper famously had a clock in her office that ran backwards. When asked about it, she would explain "Humans are allergic to change. They love to say, 'We've always done it this way.' I try to fight that."[16] DH, as a domain, follows very much in that spirit, but I believe that its

[12] Klein, "What Data Visualization Reveals"
[13] Schieber, Philip. "The wit and wisdom of Grace Hopper." *The OCLC Newsletter* 167 (1987).
[14] Dijkstra, Edsger W. "How do we tell truths that might hurt?." *ACM Sigplan Notices* 17, no. 5 (1982): 13-15.
[15] Beyer, Kurt W. *Grace Hopper and the invention of the information age* (Mit Press, 2012): 172.
[16] Schieber, "The wit and wisdom of Grace Hopper."

interdisciplinary potential for innovation remains hindered by epistemological conflicts born of 'what we have always done' that have largely remained under-investigated and unchallenged on a computational front.

This thesis is driven by a curiosity about what DH could have been if at its origin there had been a concerted focus on establishing a more equal but also more blended union of computing and the humanities at a methodological but also, crucially, theoretical level. Luckily, the delineation between what could have been and what could be is blurred when the clock is running backwards.

Chapter 1: The Humanistic Interface

As a domain with roots in the scientific tradition, Human Computer Interaction (HCI) scholarship has largely focused on design mechanics. Advances in interface design driven by engineering values have consistently pushed towards an ideal that is "task-oriented and efficiency driven" and "designed to abstract their use from any whiff or hint of ambiguity."[17] This, however, as argues DH scholar Johanna Drucker, fails to account for the fact that an interface, unlike other components of computational systems, is meant to provoke a "cognitive experience",[18] which supposes an engagement and individuality that is lost in design practices that strive for objectivity and universality.

In her analysis re-imaging HCI through a media theory lens, Drucker highlights the ways in which these kinds of interfaces are particularly maladapted to humanities contexts as they "preclude humanistic methods from their operations because of the very assumptions on which they are designed."[19] Her argument is centred on two key points—first, that all data is constructed, and that it is essential that this be explicitly reflected in any interface; and, second, that in the act of knowledge creation, information cannot be separated from graphical forms and individual context.

'Data', Drucker argues, is perhaps not the best term. 'Capta'—signifying 'taken' as opposed to 'given'— better reflects its inherent 'constructedness'.[20] Data is only ever a simplified, reduced complexity rendering of reality, whose dimensions are chosen by humans who are unavoidably biased. The danger of interfaces entrenched in a tradition of empiricism is that they flatten "the planes of reference, discourse, and processing so that they appear to be a single self-evident surface."[21] This erasure of granularity, incompleteness, and ambiguity crucially eliminates the utility of an interface as a tool for interpretation. Relatedly, Drucker insists on the recognition

[17] Drucker, Johanna. "Humanities approaches to interface theory." *Culture machine* 12 (2011): 1.
[18] Ibid, 9
[19] Drucker, Johanna. "Humanistic theory and digital scholarship." *Debates in the digital humanities* 150 (2012): 85.
[20] Drucker, Johanna. "Humanities approaches to graphical display." *Digital Humanities Quarterly* 5, no. 1 (2011): 1-21.
[21] Drucker, "Humanities approaches to interface theory," 13

that the modes by which data is presented and accessed impose a reading of its nature and limits. This implicit rhetoric, however, has been naturalised to the point of invisibility by the default to 'best practices' selected according to "expectations of performance or tasks or even behaviors" of a nonexistent universal user. This additionally means that the mode of interaction is decidedly unidirectional, without any exchange with the user. In designing interfaces for the humanities, she emphasises the need for mechanisms which support observer co-dependency such that an interface is "space of being and dwelling"[22] capable of meaningfully adapting to the user's context, but also make an effort to broadcast the ways in which they influence knowledge construction. "The digital humanities can no longer afford to take its tools and methods from disciplines whose fundamental epistemological assumptions are at odds with humanistic method",[23] writes Drucker—which begs the question: what theories and practices are constitutive of a fundamentally humanistic interface?

Shaowen and Jeffrey Bardzell's theorisation of humanistic HCI, which they define as "any HCI research or practice that deploys humanistic epistemologies…and methodologies…in service of HCI processes, theories, methods, agenda setting, and practices", starts by identifying the throughlines of "humanistic knowledge contributions": contextualisation anchored in history and tradition; conceptual analysis, which is to say a deliberate focus on, and critical examination of, the "concepts we think with"; interpretation and hermeneutic analysis; and social action, with a distinct emphasis on emancipatory thinking.[24]

Off the back of these hallmarks of humanistic work, Bardzell & Bardzell propose five broadly defined practices characteristic of humanistic HCI. The first two, 'interaction criticism' and 'critical discourse analysis', ask that we critically consider the epistemological assumptions underpinning digital tools. This includes interrogating our interaction with the affordances of computation as a medium and an interface's procedural rhetoric in the aim of exposing "hidden epistemic or ideological limitations", as well as calling into question, as Drucker does, the scientific seeming in which HCI practices are frequently cloaked, notably through comparison with methods rooted in firmly humanistic principles such as affective computing. Their third

[22] Ibid, 12
[23] Drucker, "Humanities approaches to graphical display," 2
[24] Bardzell, Jeffrey, and Shaowen Bardzell. "Humanistic Hci." *Interactions* 23, no. 2 (2016): 20-29.

practice, ‘critical social science’, relatedly asks that we reconsider the diametrical opposition of sciences and the humanities and instead strive to enact methodologies that blend and construct them for mutual advancement. Their final two principles, ‘design futuring’ and ‘emancipatory HCI’, imagine disrupting norms and challenging the limits of current technologies through practices which centre participatory and speculative design and give voice to a plurality and diversity of perspectives.

Though compatible with DH scholarship, what is missing from their theorisation of humanistic HCI is the notion of a reciprocal relationship. The authors plainly state that they view humanistic HCI as strictly separate from DH, seeing DH as computers applied to the humanities and HCI as the opposite. While I would argue that, at least in theory if not in practice, DH is crucially both the application of computational theories and methodologies to humanities contexts and the integration humanistic epistemologies and practices with computational problems and logics,[25] the Bardzells’ framework is firmly one-way.

Sinclair et al’s example-based analysis of DH approaches to information visualisation echoes many of the same points raised in more theoretical work.[26] As Drucker did, they pinpoint the challenge of creating an interface that represents the ways in which data itself and the mechanisms involved in its transformation and display are deliberately constructed and partial. For interfaces to be useful tools for research, they must clearly communicate the ways in which “the interpretive work is being guided and biased by the data and software.” This challenge is most acutely felt in the case of exploratory interfaces aimed at users having no prior knowledge of a problem or data. These interfaces must both do the most ‘storytelling’—highlighting for the uninitiated user data facets which may be of interest—but also serve the naive user base most likely to mistake this biased guidance for objective and singular reality. Similar to Bardzell & Bardzell, Sinclair, Ruecker, and Radizikowska emphasise the importance of focusing on narrative multiplicity, disrupting the “converg[ence] towards a single interpretation that cannot be challenged” that is a product of engineering conventions favouring aggregation and

[25] Burdick, Anne, Johanna Drucker, Peter Lunenfeld, Todd Presner, and Jeffrey Schnapp, “A Short Guide to the Digital_Humanities” in *Digital_Humanities*. (Mit Press, 2016), 121-136
[26] Sinclair, Stéfan, Stan Ruecker, Milena Radzikowska, and I. N. K. E. Inke. "Information visualization for humanities scholars." *Literary Studies in the Digital Age-An Evolving Anthology* (2013).

abstraction and de-emphasizing granularity and complexity. Pursuant to this, their most salient measure of success of a humanistic data visualisation is its capacity to support interpretive exploration. Though it may suggest data entry points, it should not be to the detriment of other trajectories. Though many visualisation techniques require the creation of derived categories and hierarchies, this should not preclude access to more ambiguous and nuanced data. They judge the most successful interfaces to be those that afford "new and emergent ways of understanding the material." I see this last element as highly related to the creation of methods of data display and interaction that blend humanistic and scientific epistemologies. Though pure quantification may create a new version of the source information, it does not serve interpretation and new knowledge construction if there is a disconnect between the two. It is the process of transformation that induces reflection by bringing to the fore questions around key facets or implicit taxonomies that might not previously have been considered, but are foundational to the computational model.

Given the established focus on situated knowledge and participatory and emancipatory design, feminist scholarship is highly relevant to defining humanistic data practices. Catherine D'Ignazio and Lauren Klein draw on feminist science and technology studies for its focus on situated knowledge, as well as feminist DH, feminist HCI—in particular, the way in which it challenges the existence and value of universal usability—and critical cartography studies[27] to derive seven core principles of 'data feminism'.[28] In the context of interfaces, the first two principles, 'examine power' and 'challenge power', ask that we consider both the power and influence of the data interface itself as well as the power dynamics underpinning its creation. User interfaces play a crucial role in data democratisation and should therefore be created to be accessible to users spanning a wide range of social, economic, cultural, and technological contexts. A major aspect of this is the creation of a two-way dialogue, which is to say recognizing the joint act of knowledge creation and "acknowledg[ing] the user as a source of knowledge in the design as well as the reception of any visual interface"[29] by incorporating their context into process decisions. Concretely, this implies more participatory design, and a flattening—or, at the least,

[27] D'Ignazio, Catherine, and Lauren F. Klein. "Feminist data visualization." *Workshop on Visualization for the Digital Humanities (VIS4DH)*, 2016.
[28] D'Ignazio, Catherine, and Lauren F. Klein. *Data feminism* (MIT press, 2020).
[29] D'Ignazio and Klein, "Feminist data visualization"

critical examination of—both intra-design team hierarchies as well as those between the researchers and the populations they serve.

Further principles 'embrace pluralism' and 'consider context' double down on this more implicated approach. Pluralism stands as the antithesis to the notion of the universal user, shifting instead to focus on user communities, and contextual design is informed by the knowledge making practices of these communities. The practice of context-informed design requires grappling with the situated and constructed nature of data. All data is a partial—in both senses of the term. The only perfect map is one the size of the territory[30]; it is impossible to have any degree of abstract representation without simplification. All data is capta because it is constituted by humans, all of whom have implicit biases, deciding what parts should stand for the whole. Contextual and community-focused practices put these representational choices in the hands of communities, so as to engage in practices reflective of the values and knowledge systems of the communities to whom they pertain. This can often mean grounding the design in different ways of knowing, an act correlated to two further principles, 'rethink binaries and hierarchies' and 'elevate embodiment and emotion'. Indigenous scholars Maggie Walter and Michelle Suina, for example, describe research methodologies based on the concept of the Indigenous Lifeworld, which emphasise lived experience and relationality.[31] These qualitative metrics and quantitative measures based on scales and practices tied to land relations have historically been devalued by the colonial western fetichisation of objectivity and rationality. Feminist data practices readily invite the deconstruction of the false emotion & subjectivity/reason & objectivity binary and entrenched hierarchy that positions the latter as a more valid way of understanding the world. Interfaces, after all, are mechanisms designed to provoke a specific cognitive experience—the subjectivity is inherent.

No matter the values underlying choices in datafication, D'ignazio and Klein's final principle, "make labour visible", asks that this mapping process be centred, explored, and questioned as opposed to deliberately effaced so that these lacunar renderings of reality do not "pass as

[30] Borges, Jorge Luis. "Of exactitude in science" in *A Universal History of Infamy* (Penguin Books, 1975): 31
[31] Walter, Maggie, and Michele Suina. "Indigenous data, indigenous methodologies and indigenous data sovereignty." *International Journal of Social Research Methodology* 22, no. 3 (2019): 233-243.

unquestioned representations of 'what is'."[32] A major part of this is explicitly recognizing the contributions of everyone who had a hand in constructing the data. In the midst of the rise of 'information capitalism' which Kate Crawford describes as premised on the "dual operation of abstraction and extraction",[33] this emphasis on labour as well as the established focus on perspective diversity echoes Bardzell & Bardzell's assertion that humanistic data practices should work towards emancipation and social change. As Drucker maintains, data collection is in fact data construction, and more often than not, it essentially consists of performing principal component analysis on human lives. A critical examination of the powers deciding what information to retain and of the bodies, the individuals, the human beings from whom the data is taken is essential to countering hegemonic and exploitative practices which have been shown to be particularly detrimental to women and marginalised communities.[34]

1.1 Core Principles

The interdisciplinarity of DH means that digital humanists are interested in high-level overarching statistical trends, but also want a digital mapping that just as easily affords the identification and exploration of singularities. They are searching for an encoding that is machine compatible but also preserves ambiguity and nuance. Cela colle mal, one could say, with typical computational logics born of an engineering tradition that prioritises categorisation, normalisation, and aggregation. Using humanities data does not make an interface humanistic. The design of a truly humanistic interface must "embody specific theoretical principles drawn from the humanities."[35] It is in the handling of three core characteristics that I believe we can read the extent to which an interface enacts humanistic principles: transparency, generativity, and interpretability.

1.1.1 Transparency

The evolution of computing has consisted in no small part of humans adding levels of abstraction for the benefit of other humans on top of levels of already human defined logic and presenting

[32] Drucker, "Humanities approaches to graphical display," 2
[33] Crawford, Kate. "Conclusion: Power." in *The Atlas of AI* (Yale University Press, 2021), 217.
[34] See, notably: Perez, Caroline Criado. *Invisible women: Data bias in a world designed for men* (Abrams, 2019); Noble, Safiya Umoja. *Algorithms of Oppression* (New York University Press, 2018)
[35] Drucker, "Humanistic theory and digital scholarship," 86

this as ‘just the way the computer thinks’. In execution, however, pleasing nested conditional control flow becomes GOTO statements, and readable looping logic gets unrolled. Interfaces are simply another extension of this pattern.[36] The problem is, none of it is ‘just how the computer thinks’. The computer doesn’t think. The layers of mappings between human understandable and machine understandable representations of the world constitutive of data and data interfaces are defined by humans. However, despite these mappings being the site of non-evident reconciliations of differing worldviews, of the definition of ontologies, of reductions in complexity, they are most often obscured by interfaces striving for the pervasive HCI gold standard of invisible ‘natural’ interaction. Loup Cellard and Anthony Masure argue that this “organisation of the legible and the visible is in fact a calculated construction”[37]—a carefully curated false transparency. In a humanities context, I see it as crucial to reject this sanitised version of transparency to instead engage with discord and discomfort rather than sweeping them under the metaphorical rug.

A humanistic interface should make an effort to expose human to machine mappings at two sites—data construction and processing, and the design of data manipulation and display mechanisms. The former is the recognition of data as capta. It is impossible for data to capture all aspects of reality, but the choices that were made should be exposed and explained. The latter refers to acknowledging the procedural rhetoric[38] peddled by the interface. Much as with data, faced with the impossibility of representing everything, an interface designer must adopt a specific viewpoint that guides the selection of a limited set of elements to display or questions to which to respond. Design choices such as navigational cues and visualisation paradigms further contribute to an implicit rhetoric that tells the user what data and relationships are important. Explicit acknowledgement of this storytelling creates space for users to imagine and explore alternate pathways and prevents the merging of interpretation and fact. “All graphical schema are built on the single principle of defining classes of entities and of relations”, writes Drucker, “for a humanistic approach, these have to be defined as rhetorical arguments produced as a result of

[36] Schuwey, Christophe. “Humanités numériques et études littéraires : une question d’interfaces” *La lettre de l'InSHS* (2018): 25-27.
[37] Cellard, Loup, and Anthony Masure. "Le design de la transparence." *Multitudes* 4 (2018): 100.
[38] Bogost, Ian. *The rhetoric of video games*. MacArthur Foundation Digital Media and Learning Initiative, 2008.

making, a poetics of graphical form, not in the reductive or abstract logics of Boolean algebra."[39] All knowledge is situated; a neutral 'view from nowhere'[40] does not exist. This reality is not a fault; what is harmful to learning and understanding is the failure to acknowledge it. In both cases, interface self-disclosure crucially also serves to highlight the limits of computational representation, specifically with respect to rendering bias and uncertainty.

1.1.2 Generativity

In an echo of Sinclair et al., Christophe Schuwey argues that the most essential contribution of DH interfaces is that they facilitate the construction of new relationships to our objects of study[41]. At the difference of 'information representations', these 'knowledge generation' interfaces look to create space for "the humanistic tenets of constructedness and interpretation."[42]

The paradox of using data interfaces as a starting point for humanistic research is that they begin at the end, in a way. With the impossibility of representing all data, they are necessarily created to answer a small pre-defined set of questions, "collaps[ing] the critical distance between the phenomenal world and its interpretation."[43] Interfaces are fundamentally interpretative tools, as they make sense of data by presenting a specific interpretation dependent on layers of abstraction. An interface is only a tool *for* interpretation, however, if it allows a user to at least interrogate, if not manipulate and alter, these layers.

To create space for interpretation, an interface must incorporate visualisations that "support combinatoric calculation"[44] through dynamic interactions and/or minimally mediated multiplicity. Because knowledge production is observer co-dependant, an interpretive interface should entertain a bi-directional relationship with the user. Dynamiticity and multiple entry points both create the potential for the user to enter into a dialogue with a tool, curating a personal instantiation that is best adapted to their individual questions. Sinclair's Voyant, for

[39] Drucker, Johanna. *Graphesis: Visual forms of knowledge production*. (Cambridge, MA: Harvard University Press, 2014), 54.
[40] Haraway, Donna. "Situated knowledges: The science question in feminism and the privilege of partial perspective." in *Feminist theory reader* (Routledge, 2020): 202-310.
[41] Schuwey, "Humanités numériques et études littéraires"
[42] Drucker, *Graphesis*, 125.
[43] Ibid
[44] Ibid, 105

instance, is a good illustration of both principles; its library of interchangeable tool panels allows users to interrogate texts at different levels of granularity (word, n-gram, text) and abstraction (clear cut frequencies, derived topics, speculative geographic networks). Though pre-processed visualisations can aid usability, access to detailed, complex, and messy versions of the data similarly makes room for a user to explore the scope of what is possible, induct individualised questions, and derive their own representations.

1.1.3 Interpretability

Interface interpretability is a somewhat ambiguous concept, as it depends on the goal of a given interface. With data exploration and problem solving interfaces, interpretability is often equated to usability and refers to how easy it is to locate or manipulate specific information. In a traditional HCI context, the aim is to maximise efficiency and the user's capacity to learn and retain system functionality so as to reduce error rate and increase user satisfaction.[45] Concretely, this manifests as design elements whose purpose and scope are specific and limited, and structure and feedback conventions selected to streamline interaction in an assumed collectivist vision of usability.[46]

With respect to visualisations, common guidelines for interpretability tend to fall in line with the views of Edward Tufte, who asserts the superiority of minimalist graphics for "reveal[ing] the truth."[47] Tamara Muzner's eight data visualisation rules of thumb, for instance, advocate for reducing colour use and dimensionality, strongly favour starting from aggregative overviews, and privilege the use of techniques which maintain distance from the data over more immersive options.[48] As is evident from the focus on eliminating ambiguity and distractions from a main narrative in order to reveal 'the'—as opposed to 'a'—truth, these conventions are poorly adapted to humanities contexts and epistemologies. It is precisely for this reason that Drucker and Bardzell & Bardzell emphasise the need to create entirely new ones, often premised on the

[45] Ferré, Xavier, Natalia Juristo, Helmut Windl, and Larry Constantine. "Usability basics for software developers." *IEEE software* 18, no. 1 (2001): 22-29.
[46] Masure, Anthony. "Vers des humanités numériques «critiques»." *Repéré à dlis. hypotheses. org/2088* (2018).
[47] Tufte, Edward R. *Visual and statistical thinking: Displays of evidence for making decisions*. Vol. 12. (Cheshire, CT: Graphics Press, 1997), 123.
[48] Munzner, Tamara. "Rules of Thumb" in *Visualization analysis and design*. (CRC press, 2014), 116-141.

integration of contextual definitions, structures, ontologies and hierarchies born of community-focused participatory design.

However, with the exception, perhaps, of affective visualisations, where the aim may be to convey emotion more than information, any new conventions must still be designed so as to strike a balance between disruptive display and usability. Misguided as they may be in imagining that all information is suited to identical visualisation practices emerged from statistical contexts, Tufte-esque methods are rooted in accounting for the limitations of human processing. In light of this, in imagining methods for displaying information that reflect humanistic values and practices, humanities interfaces must still consider questions of memory, perception and interaction affordances.

Interpretable humanistic interfaces that are looking to deconstruct and extend beyond universality must also consider that the 'universal user' at whom widespread HCI design principles and practices are aimed emerged from the same engineering-values centric context. Much like office temperatures, crash test dummies, and medical dosages, supposedly 'universal' digital design standards do not take gender, ethnic, or cultural differences into account.[49] This pattern of oversight is further exacerbated by a lack of diversity in the technology industry and the widespread use of internally oriented evaluation practices like 'dogfooding'.[50] Barnett et al.[51] note, for example, that women tend to interact differently with problem solving software than men, with design conventions favouring the latter. Their experiments show that, due at least in part to lower technical self-efficacy, women are generally more risk averse and will be quick to blame themselves if something malfunctions with the software. Correspondingly, women exhibit a tendency to collect as much information as possible at any level of an interface before moving forward (breadth-first) in contrast to a pattern of following the first promising information and

[49] Perez, *Invisible women*; Reinecke, Katharina, and Abraham Bernstein. "Improving performance, perceived usability, and aesthetics with culturally adaptive user interfaces." *ACM Transactions on Computer-Human Interaction (TOCHI)* 18, no. 2 (2011): 1-29.
[50] 'Dogfooding' refers to the practice of company employees being used as test subjects for their own beta products. See Stevens-Martinez, Kristin and Mark Guzdial. "Live Coding". The CE-ED Podcast. February 3, 2020, https://csedpodcast.org/blog/season1_episode4/, for discussion of how this can negatively impact software accessibility.
[51] Burnett, Margaret, Anicia Peters, Charles Hill, and Noha Elarief. "Finding gender-inclusiveness software issues with GenderMag: A field investigation." in *Proceedings of the 2016 CHI conference on human factors in computing systems* (2016): 2586-2598.

backtracking as needed more common among men (depth-first). Experiments further found that women are additionally less likely to tinker with new unknown features, but if they do, will often benefit more from the experience than their male counterparts as their exploration is more likely to integrate deliberate reflection on the process. Evidently, there is something troubling in attaching these different styles of interaction to gender, and Barnett et al. explicitly acknowledge that it is the set of characteristics and behaviours themselves that are more useful for evaluating software inclusivity, not their link to a specific gender. I do, however, find that it is worth noting that the variations in behaviour born, at least in part, of lack of self-efficacy and risk aversion were most common among a group of individuals who are often socialised to believe that they 'don't have a tech/math brain'—an experience that may be shared, and a belief that may be reinforced as a function of their academic identity, by many humanists. In light of this, it follows that interfaces aiming to engage with users from contexts that often fall outside of or extend beyond science and engineering should provide support for alternative or diverse modes of interaction. Crucial to cultivating a space for collaborative knowledge production is that the user feels supported and confident in their experience of the interface. A space of discomfort can never be a space of dwelling.

1.2 Pushing at the Limits of Computation

Having recognized that the epistemology underpinning many existing visualisation conventions for data manipulation and display is at odds with the fundamentals of humanities inquiry and practice, digital humanists must necessarily "synthesize [humanistic] method and theory into [new] ways of doing as thinking"[52]. Previous work looking to elaborate new computational protocols has primarily focused on the multiplication of 'visual variables'[53] such as colour, texture, and orientation, so as to be able to pack more information into each visual primitive. conventions. For example, in a speculative visualisation looking to illustrate the lengthy process of a novel "appearing" that would have been obscured in a traditional a bar chart showing a basic count increase one year to the next, Johanna Drucker adds timeline-like structures with different shapes and patterns for each step of the publication process (writing, editing, pre-press etc.) that sit perpendicular to each bar in an effort to expose its composition (figure 1.2-1).

[52] Drucker, "Humanistic theory and digital scholarship," 87
[53] Bertin, Jacques. *Sémiologie graphique: les diagrammes, les réseaux, les cartes* (De Gruyter Mouton, 1973).

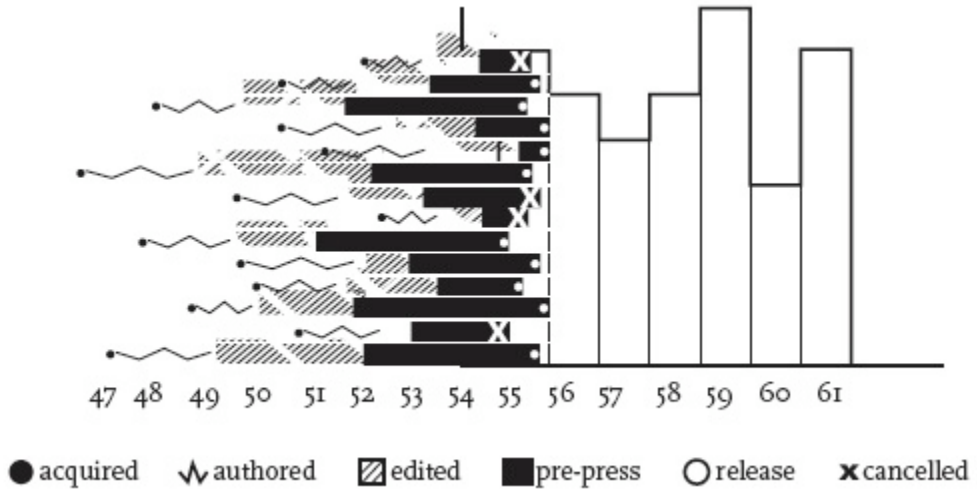

Figure 1.2-1. Graphical representation of the "appearance" of fourteen novels in 1885, with added visual variables used to represent the data trajectory. From Johanna Drucker, Humanities Approaches to Graphical Display (Digital Humanities Quarterly 5, no. 1, 2011), figure 4.

In another example (figure 1.2-2) Drucker mutates standard units of measurement to introduce a time axis that, rather than being constrained to linear and uniform progression, twists and warps to reflect human perception of time[54]. Though visually evocative, it is interesting to note that Drucker's approach to expressing nuance and ambiguity and exposing process is premised on compounding layers of abstraction. Despite one of the primary aims being to collapse the distance between the realities of the data and its display, nowhere is the data itself made visible. The visualisation is built around a distanced symbolic vocabulary.

[54] Drucker, "Humanities approaches to graphical display"

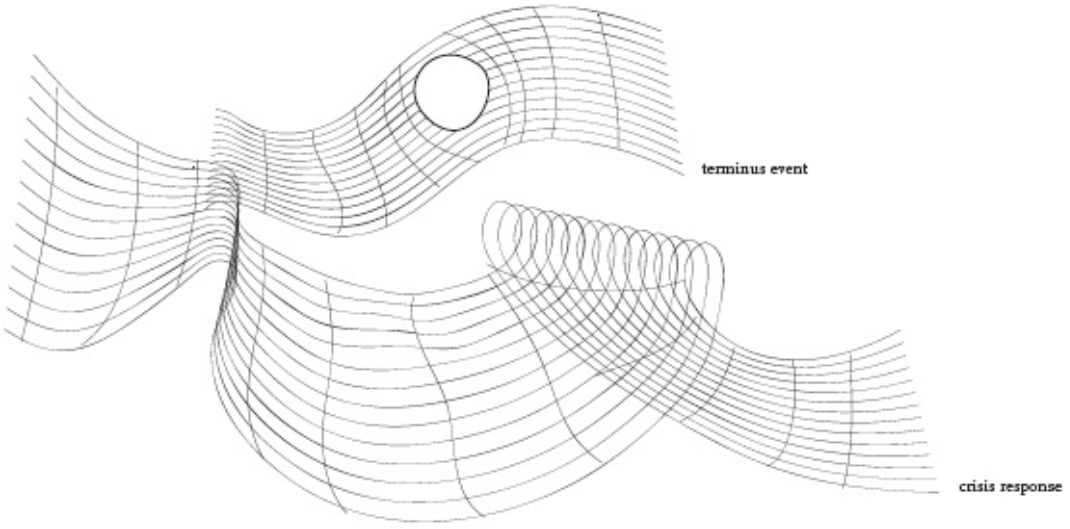

Figure 1.2-2. An alternative representation of time in the context of a crisis event. From Johanna Drucker, Humanities Approaches to Graphical Display (Digital Humanities Quarterly 5, no. 1, 2011), figure 7.

Computational abstraction is a mechanism for reducing the cognitive distance between human conceptualizations of structure and process and their machine representations. Its most primordial instantiation is high level programming languages, which serve to distance the act of programming from the underlying mechanics of calculation.[55] While I am as grateful as the next person not to have to write assembler or use punch cards, media theorist Anthony Masure argues that, with respect to interfaces, this erasure of distance is precisely the reason that "the products of digital humanities projects are rarely reflective, in their design, of [the] epistemological tensions [between computing and the humanities]."[56] One facet of the digital humanities is the use of computation as a tool for analysis, an exploration motivated at least in part by the fact that its logics afford different ways of looking at subjects. In light of this, one has to wonder at the utility of effacing their influence by layering on abstraction designed to soften the dissonance between computational and humanistic models of information processing and knowledge construction. Much of the tone of Drucker's work would seem to imply that the imposition of engineering values is in some way at fault, but I can't help but think that they are only 'imposed'

[55] Cellard & Masure, "Le design de la transparence"
[56] Masure, "Vers des humanités numériques « critiques »," 2; my translation.

through a failure to engage with them. In DH we aim to use technology to help us answer humanistic questions. Fundamental to this process should be an understanding of the way in which these questions must be transformed in order to fit into computational paradigms.

As a counterpoint to Drucker's layers of abstraction, I imagine an alternative style of humanistic interface that forefronts computational theory. By exposing the computational processing, it would centre and ask the user to engage with the human to machine translation whose complexities and necessary compromises are at the heart of enacting a more humanistic style of computing. While an interface which requires reflection into computational thinking could stand to place a greater cognitive load on the user, I maintain that an understanding of computational theory and abstraction is necessary to imagining what is *possible* computationally—something which is absolutely essential to creating new fundamentally humanistic computational models and processes.

1.3 Project Context

Given the emphasis placed on questions of situated knowledge, ambiguity, and data biography, I felt it was important to embed the exploration of humanistic interfaces in a domain specific context. I have opted to use the data of the Comédie-Française Registers Project (CFRP)[57] for in part because of my familiarity with its origins and the conditions of its production,[58] but also because I believe the CFRP data to be particularly suited to this project's context for two reasons. First, theatre data exaggerates the characteristics of humanistic data, and second, there exists an interesting parallel between the function of the CFRP data, in both modern and historical contexts, and the aims of humanistic visualisation. The following sections offer an overview of the CFRP data and its background and examine its four principal data interfaces—the Discovery tool,[59] Cross-tab browser,[60] Faceted browser,[61] and Graph tool[62]—with regard to the three dimensions of humanistic interfaces outlined in section 1.1.

---

[57] https://www.cfregisters.org/#!/
[58] I have been a research assistant with the CFRP since May 2018
[59] https://ui.cfregisters.org/, created by Logilab
[60] https://analytics.cfregisters.org/, designed by master's students at Laval University and created by Christopher York
[61] https://surfacets.cfregisters.org/, created by Christopher York
[62] https://graphe.cfregisters.org/, created by Anna Sollazzo

The Comédie-Française Registers Project is an international digital humanities initiative looking to re-examine French cultural history by way of the study of daily registers of the country's first national theatre company.[63] This archive stretches back to the company's foundation in 1680 and is comprised of four distinct register sets:

1. Receipts registers - These recount, for each evening, which plays were performed—typically two per session—and how many tickets were sold, of what types, and at what prices. They inform not only on the troupe's finances, but also on the composition of the audience. The *recettes* are the most complete set of registers, spanning from 1680-1793.
2. Expense registers - This register set shows the breakdown of the company's expenses. Though initially appended to the revenue information, the *dépenses* were recorded in their own distinct volumes from 1750 onward.
3. Casting registers - The *feux* registers, named for the candles or firewood given to each actor performing on a given evening to light and heat their dressing room, are a record for every performance of which actors played which roles. Though incomplete casting information can sporadically be found in early receipts registers, the casting registers proper only span from 1765 to 1793.
4. Administrative registers - The *registres de l'assemblée* are essentially the minutes from the actors' weekly administrative meetings,[64] during which they made decisions about the troupe's finances, management, and repertory.

The importance of the registers to the troupe and to the institution itself is evidenced by their materiality and comprehensiveness. The registers are mostly in-quarto volumes bound in vellum, using high quality paper and often including pre-printed rubrics and a title page decorated with the royal coat of arms. What's more, despite spanning well over a century, there is a remarkable

[63] Harvey, Sara and Agathe Sanjuan. "Le projet des registres journaliers de la Comédie-Française : les humanités numériques, dialogue entre les mondes de la recherche et de la documentation". *Bulletin des bibliothèques de France*, no. 9 (July 2016): 146-152

[64] Though overseen by the *Gentilshommes de la Chambre*—representatives of the CF's royal patrons— the *sociétaires* (full company members) were completely responsible for managing the theatre's programming, finances, and other administrative facets.

consistency to the way in which the registers are dated, and there are very few gaps in the data. As a historical and documentary resource, they are unparalleled with respect to the records kept by other contemporary European theatrical institutions, such as the Comédie-Italienne or the Drury Lane Theatre. This elevated status is reinforced by the fact that, through to today, new members inducted into the company are gifted a facsimile of the first register. The digital datasets derived from each register set are at the heart of the CFRP. As more fully detailed in section 2.2.2, the scope of the data used in this thesis is limited to the recettes and feux, since only the relational databases born of those two register sets have been fully completed and verified.

Theatre data is an interesting subset of humanities data in that the ephemerality of performance serves to amplify the challenges of, and interpretive bias inherent to, the datafication process. Representations constructed from the texts of the plays, from accounts of performances published in periodical press, or from heterogeneous collections of ticket stubs, musical scores, and set design blueprints are all valid, but impose very different readings of what is artistically, culturally, and historically significant[65]. At the difference of DH projects like 'MOLIÈRE21'[66], which takes the CF's spiritual patron as its sole focus, or 'La haine du théâtre'[67], whose approach pivots around texts related to theatrical controversies, the CFRP adopts a more holistic approach to theatre history by centering narrative multiplicity and flattening hierarchies. In consulting the registers, the reader is confronted with the reality of the company's day to day operations, which depend on the collaborative efforts of a huge number of agents. In their pages, the works of Voltaire are listed identically to those of Françoise Graffigny, page notes name the *souffleurs*[68] before mentioning visiting royalty, and tragic starlette Clairon is on equal footing with Monsieur Bellemont, whose riveting roles include twelve distinct characters simply described as "a lacquey" and the no doubt essential "chair carrier" in Molière's *Les Précieuses Ridicules*. Taking the registers as an object of study provokes a shift away from the view of theatre as literature to a

[65] Escobar Varela, Miguel. "Introduction: Pursuit of Theater's Digital Traces" in *Theater as Data: Computational Journeys into Theater Research* (Ann Arbor: University of Michigan Press, 2021), 1-20
[66] http://moliere.huma-num.fr/
[67] http://132.227.201.10:8086/projets/la-haine-du-theatre
[68] Person who stood in a hole at the front of the stage and prompted the actors if they forgot their lines.

broader consideration of all of the agents and elements constitutive of *la vie théâtrale* as a whole.[69]

Catherine D'Ignazio writes that "data is the currency of power"[70] in reference to the influence of contemporary Big Data, but this accurately describes the registers in their historical context. In a first sense, they were a record of compliance to the conditions of the royal patronage that protected the monopoly of French language theatre in Paris that was key to the company's success. Retrospectively, they also disrupt the attempted disempowerment of actors by the church. Excommunicated for the sin of practising their chosen profession and correspondingly stripped of a number of civil liberties by virtue of the judiciary power of the church, actors and actresses were denied access to the ecclesiastical and legal records that are often significant sources of information about artisans and the working class in the period in question.[71] The registers, however, which document the actors' lives in much greater detail[72] than ecclesiastical records could ever have provided, represent the very record of their existence, value, and importance that the church tried to deny them.

In view of its content and context, the CFRP archive could be viewed as being to 'traditional' hegemonic history what humanistic interfaces are to 'traditional' aggregative ones; both serve to decentre power and give voice to those who are the subject of the study, emphasise alternative measures and perspectives, and expose the labour behind constructed representations.

The CFRP are not the first to undertake a study of the registers, but are rather extending a tradition of scholarship.[73] Between 1752 and 1758, Charles de Fieux, Chevalier de Mouhy published his *tablettes dramatiques*,[74] which are essentially a dictionary of all the plays

[69] Biet, Christian, Sara Harvey, and Agathe Sanjuan. "Postface–Le Programme RCF, de l'archéologie à la futurologie." in *Données, recettes & répertoire: La scène en ligne (1680-1793)* (MIT Press, 2020).
[70] D'Ignazio, Catherine. "Creative data literacy: Bridging the gap between the data-haves and data-have nots." *Information Design Journal* 23, no. 1 (2017): 6-18.
[71] Sanjuan, Agathe, and Martial Poirson. *Comédie-Française: une histoire du théâtre* (Seuil, 2018).
[72] For example, we know the exact time and cause of actor Lekain's death, as well as the time and location of his burial. The registers also record quotes from people present at both events, including the King's reaction to learning of Lekain's passing—"J'en suis bien fâché ; la Tragédie est morte."
[73] Ravel, Jeffrey S. "The Comédie-Française by the Numbers, 1752–2020." in *Databases, Revenues, & Repertory: The French Stage Online, 1680-1793* (MIT Press, 2020).
[74] de Fieux Mouhy, Charles. *Tablettes dramatiques contenant l'abrege de l'histoire du theatre francois, l'etablissement des theatres a Paris, un dictionnaire des pieces... Avec 3 Supplem*. (Sebast. Jorry, 1752).

performed at the Comédie-Française, with titles accompanied by basic information—author, premiere date, genre—as well as occasional subjective commentary. In 1901, Alexandre Joannidès published *La Comédie-Française de 1680 à 1900: Dictionnaire général des pièces et des auteurs*,[75] which likewise listed the company's repertory and authors but notably expanded the scope of Mouhy's work to include a tabular summary of performance numbers. Joannidès' work sought to demonstrate how these rudimentary statistics allowed for the identification of trends in author and play popularity. Finally, in a number of works published between 1941 and 1951,[76] Henry Carrington Lancaster compiled further tabular data, his notably being the first work to include information relating to the troupe's finances. As highlighted by Jeffrey Ravel, Lancaster in particular recognized the way in which the study of the registers affords a change in perspective, and that this move to look beyond the scope of literary canon invites important new insights. Lancaster's study led him to believe that "To limit one's knowledge to three leading writers [Corneille, Racine, and Molière] is comparable to the old method of studying history only in its wars, its political negotiations, and the private lives of its kings and queens."[77] It is this thread that runs through the work of the CFRP and is emphasised by digital affordances. As Lev Manovich argues, the database as a new media object serves to invert the traditional relationship between the 'syntagm'—a single realisation of narrative possibilities—and the 'paradigm'—the collection of all possible choices, dematerializing the former while concretizing the latter[78]. The transformation of the registers into relational databases, beyond extending the capabilities of Mouhy, Joannidès, and Lancaster's static tables, could therefore be seen as compounding the CFRP's focus on narrative multiplicity.

1.3.1 Interfaces

In addition to the databases and their corresponding public APIs, the CFRP have built a number of digital interfaces for interacting with the register data. Peripherally aware of the notions of

[75] Joannidès, Alexandre. *La Comédie-Française de 1680 à 1900: dictionnaire général des pièces et des auteurs* (Plon-Nourrit, 1901).
[76] Henry Carrington Lancaster, *The Comédie-Française 1680-1700: Plays, Actors, Spectators, Finances* (Baltimore: Johns Hopkins University Press, 1941); *The Comédie-Française 1701-1774: Plays, Actors, Spectators, Finances* (Philadelphia: American Philosophical Society, 1951); *A History of French Dramatic Literature in the Seventeenth Century*, 9 vols. (Baltimore: Johns Hopkins University Press, 1929-1942).
[77] Lancaster, *A History of French Dramatic Literature in the Seventeenth Century*, vol. 5, *Recapitulation, 1610-1700* (Baltimore: Johns Hopkins University Press, 1942), 148 quoted in Ravel, "The Comédie-Française by the Numbers".
[78] Manovich, Lev. "Database as a symbolic form." *Museums in a digital age* (1998): 64-71.

interface non neutrality and presentation bias, the researchers decided, in the words of one of the principal investigators, to "multiply voices by multiplying the number of interfaces". This effort yielded four general-purpose data interfaces,[79] each of which offers a different entry point into the data. Three were created to explore the *recettes* and the fourth, the *feux*. Each interface is a product of developers with varying backgrounds, levels of subject expertise, and degrees of contact with the researchers and therefore represents a different approach to designing for a humanities context.

The following sections will examine the ways in which each tool is handles the established salient facets of humanistic interfaces described previously:

- Transparency- Does the interface make an effort to recognise and expose the constructed nature of the data? Does it look to engage in questions of visual semiotics?
- Generativity - Does the interface's presentation of the data leave entry points, ontologies and hierarchies open to deconstruction and investigation? Do data representations incorporate dynamicity and multiplicity so as to support interpretative analysis?
- Interpretability - Does the interface provide cues to navigating the system? Does it draw on a combination of domain conventions and support diverse users through varied modes of interaction?

1.3.1.1 Discovery Tool

The Discovery Tool was the first of the four tools to be created and it draws exclusively on the *recettes* database. The landing page (figure 1.3.1.1-1) presents the user with four options of index groups to explore—plays, seasons, authors, and genres. Each option links to a page listing individual instances that can be filtered and sorted according to a limited set of their attributes (figure 1.3.1.1-2). Selecting an item in the list leads to a terminal level page that offers further details on an individual instance. For an author, for example, this includes breakdowns of

[79] Some additional smaller and more targeted tools were created during the project's two hackathons (2015, 2016), but I have opted to limit my analysis to the four principle ones so as to be comparing interfaces of similar scope. The most elaborate of the experimental tools is the seating heatmap, viewable at https://heatmap.cfregisters.org/. The others can be accessed at https://www.cfregisters.org/#!/outils/experimentations.

performances of their plays according to genre and authors alongside whose work they were most often programmed, as well as an overview of their aggregate revenue (figure 1.3.1.1-3)

Figure 1.3.1.1-1. Discovery Tool - landing page

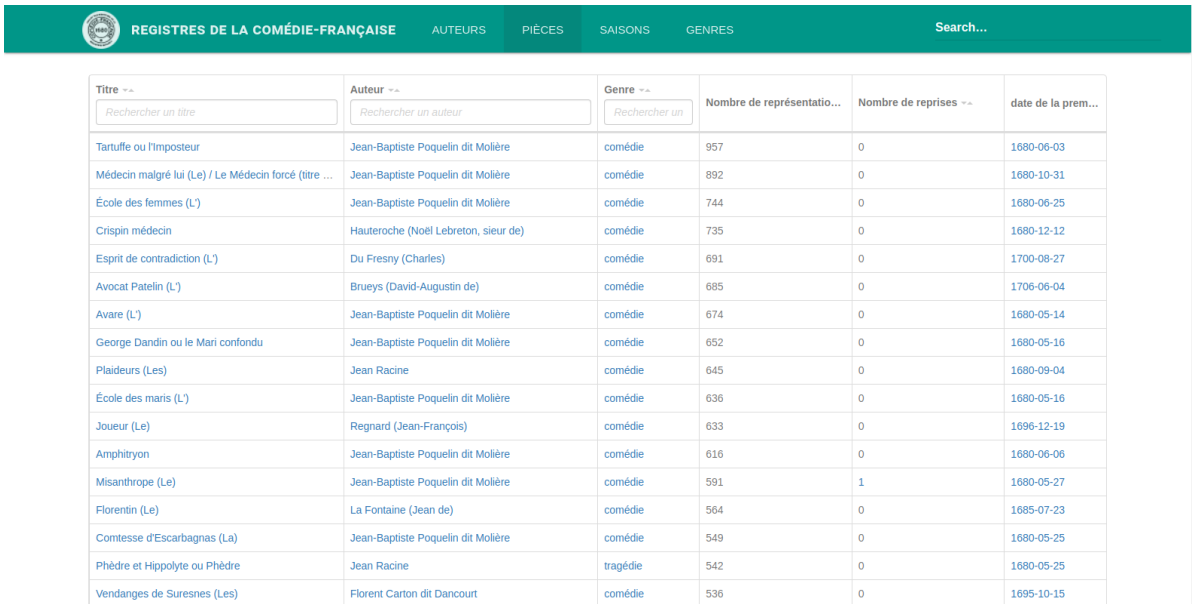

Figure 1.3.1.1-2. Discovery Tool - index list page (plays)

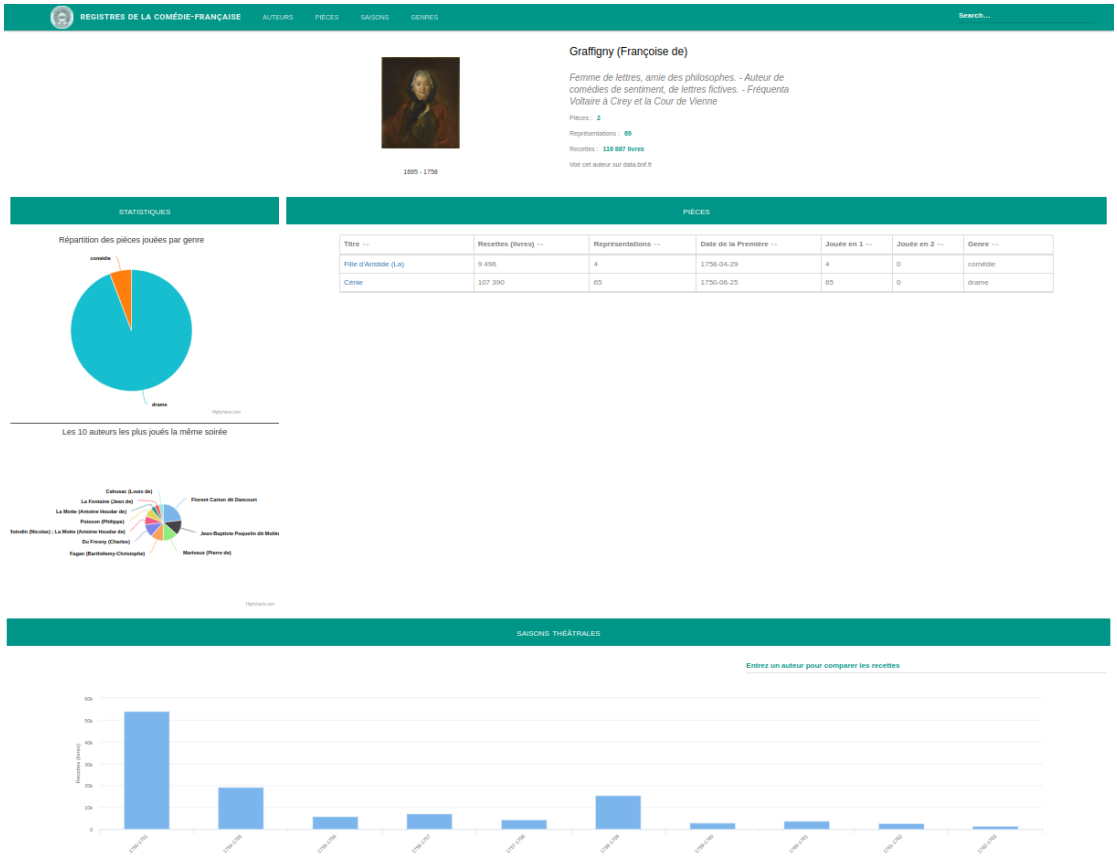

Figure 1.3.1.1-3. Discovery Tool - author index detail page (Françoise de Graffigny)

1.3.1.1.1 Transparency

The encyclopedic nature of the Discovery tool could be seen to lend it an aura of self-evidence and comprehensiveness but, like all encyclopedias, it is the site of ample opaque ontological decision making. Of the four foundational index categories, only two—seasons and plays—are fully endogenous to the registers. There is no indication given in the interface that these are in any way different from author and genre. This is especially notable with respect to genre, where the genres originally assigned by the researchers in accordance with contemporary sources are simplified down to five meta-categories. No details of how this mapping was established are given, and the unprocessed values are never exposed. This data transformation runs counter to the humanistic focus on plurality and participates in an interface rhetoric which potentially reinforces monumental historical tendencies rather than disrupting them. Additionally, though the homepage does not suggest any hierarchy between the different initial paths of exploration,

the fact that the types of graphics chosen for the individual index level pages are ill suited to sparse data further participates in a rhetoric that elevates already central figures.
With respect to data constructedness, the most significant oversight is a lack of nuance in the presentation of aggregate revenue data. Both author and play pages feature individual revenue totals despite it not being possible to calculate these values with any degree of certainty impossible because of the structure of the registers. Tickets at the Comédie-Française were not sold for individual plays, but for sessions that typically featured two performances. The revenue for each play is inextricably linked to its context; to equate an evening's takings with a play's revenue for a given date is confusing at best, if not misleading. For instance, on the 26th of January 1724, the season wise graph of playwright Regnard's revenue shows a notable spike. It is not the case, however, that the actors gave a particularly strong performance of his play *La Sérénade* such that it brought in over 1000 *livres* more than any other evening that season that included one of his plays. Rather, that session saw the play paired with the first performance of a new play—Boissy's *L'Impatient*—with the *création* as the reason for the record ticket sales.

1.3.1.1.2 Generativity

Aggregation-focused, the Discovery tool most often does not provide access to granular data, and the graphics in the interface are principally static. There is no functional mechanism for comparing different indices of the same type and axes of interpretation (performances, revenue) are predefined according to index type. Author pages, for instance, only show season-performance aggregations and play pages are limited to play-revenue graphs. With the exception of season-wise revenue heatmaps, which are not aggregative, no graphics inform on both revenue and performance numbers.

Light networking—links between index instance pages of different types—does, to an extent, allow a user to easily follow their intuition to piece together evidence for unique arguments not addressed by the existing components. Links to external sources[80] further help expand the universe of possible explorations and interpretations.

[80] Gallica - BNF (https://gallica.bnf.fr/accueil/en/content/accueil-en?mode=desktop) and the La Grange database (https://comedie-francaise.bibli.fr/index.php?lvl=cmspage&pageid=6&id_rubrique=84)

1.3.1.1.3 Interpretability

The Discovery tool is interpretable by HCI standards as it closely adheres to Schneiderman's mantra "Overview first, zoom and filter, then details-on-demand."[81] Visualisations are drill-down on click and tooltips show data descriptions. It offers, however, only a single mode of interaction.

1.3.1.2 Cross-tab Browser

The Cross-tab browser takes a more centralised approach than the Discovery tool. Contained to a single frame, filters pertaining to plays and their metadata (genre, authors) as well revenue details (ticket sales, venues) can be applied in various combinations within a customizable time frame to visualisations related to one of four pre-defined measures—aggregate revenue, average ticket price, average revenue, and aggregate performances. Results are displayed in tabular and/or graph form, depending on the user's selection. Figure 1.3.1.2-1 shows a basic search.

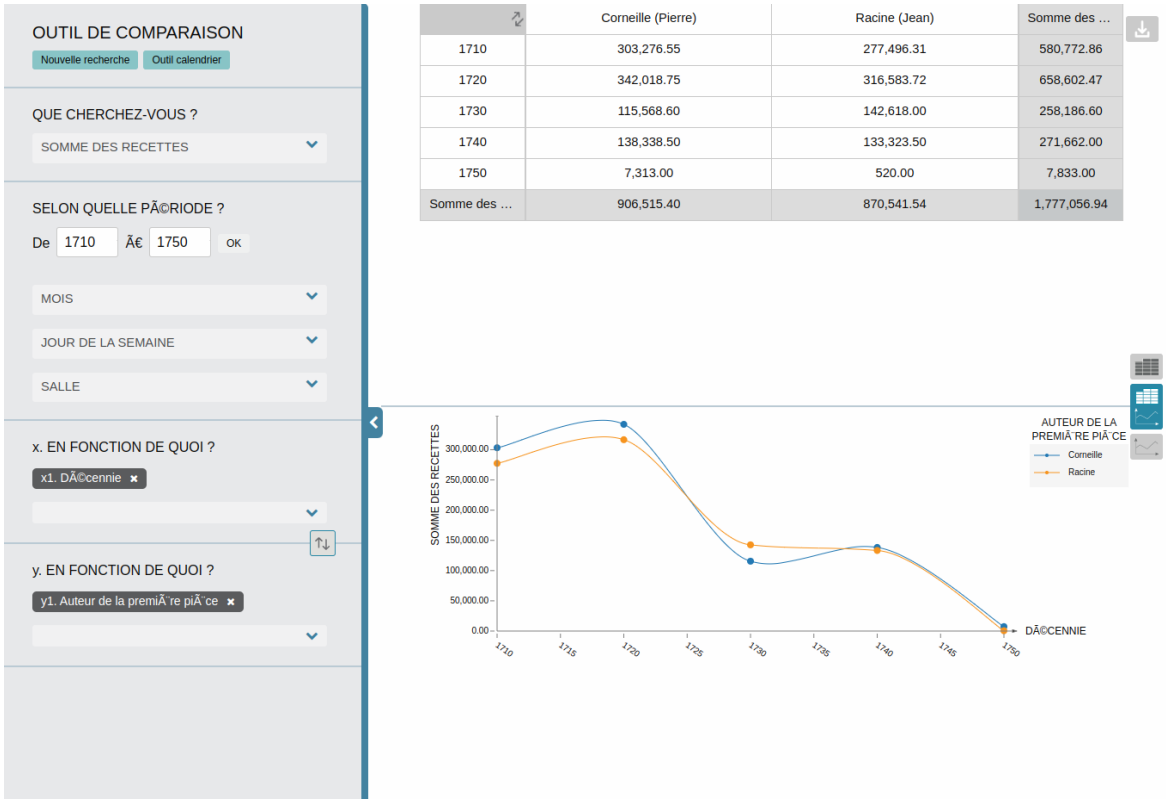

Figure 1.3.1.2-1. Cross-tab Browser

[81] Shneiderman, Ben. "The eyes have it: A task by data type taxonomy for information visualizations." In *The craft of information visualization* (Morgan Kaufmann, 2003): 364-371.

1.3.1.2.1 Transparency

The Cross-tab browser does very little in the way of data obfuscation with respect to index choice, though unprocessed sales and performance count data is not made available. The issue of individual author and play revenue is addressed by the interface clearly noting that the values correspond to sessions where an author's work or a specific play is the nth ($0<n<4$) performance given. The most notable instance of forced perspective is the addition of decade as a unit of time. Additionally, all data is arbitrarily cut off at 1790, though in reality it extends to 1793, and no indication is given as to whether ranges are inclusive or exclusive.

Like the Discovery tool, the Cross-tab browser derives a certain number of index categories. Notably, it separates out the number of acts and the genre from the plays themselves. This means that some combinations of filters are of limited utility; when displaying data for selected plays, a secondary filter to weed out specific genres is the functional equivalent of simply selecting fewer plays. Even more explicitly than the Discovery tool, the Cross-tab browser elevates already central figures. The default example loaded, for instance, is a comparison of the aggregate revenues of Jean Racine and Thomas Corneille, two of the most well known tragic authors of the period.

1.3.1.2.2 Generativity

With four aggregations and two filters with six choices, many of which have a number of sub choices, it is possible to produce a huge number of different graphics. The cross-tab browser is a prototypic example of the potential for interpretative work born of 'combinatoric calculation'. Different index instances—specific plays or seating types, for instance—can be added at any time, allowing the user to iterate on a question. The order in which the two filters are considered and, correspondingly, the format of the display, can also be easily inverted to give the user a new view on the data. The graphics themselves, however, are static and limited to bar and line charts, depending on the nature of the requested data.

1.3.1.2.3 Interpretability

The Cross-tab browser is a powerful tool for answering diverse and specific questions, but its interpretability is hindered by the fact that it attempts to borrow from both subject specific and

scientific conventions and vocabularies without sufficient integration or explanation. The graphical conventions chosen are clear cut, but interactivity, and therefore feedback, is limited. Hover initiated highlight mappings between graph-view and table-view help to clarify and concretize the relationship between the two. However, there are no mechanisms built into the interface to highlight or explain cases where an invalid combination of filters appears to initiate processing but returns an empty or erroneous result.Though someone with a knowledge of the Comédie-Française who knows that the company didn't move to the *Salle des Machines* until 1770 would understand why there are no results for Voltaire plays performed at that venue between 1730 and 1760, for instance, this would likely be confusing to a naive user. Additionally, the names of the aggregation options do not match up with what they represent. An expert user would question the selection 'représentations / jour' knowing that a play would be performed a maximum of once per day. A naive user might come away with the impression that hundreds of plays could be performed per day. Both would be confused by the fact that this option actually displays the total number of performances, as opposed to any sort of average.

The filters are a further site of potential confusion. The two options open to selection are framed as controlling the x and y-axis of the graph, deliberately—though arguably not correctly—invoking mathematical language through the introductory phrase 'as a function of what?'. In reality, though the first selection does influence the x-axis, the second selects the chart series. The true y-axis is always the selected aggregation. The series options themselves are presented in full detail, using subject specific ontologies, but are presented without explanation. Users comfortable with graphs may find the axis relationships confusing and not have the context for meaningful series selection. Users coming from the humanities may be put off by the blatantly mathematical framing. Rather than mutual intelligibility, this combination of conventions instead creates a parallel discomfort.

1.3.1.3 Faceted Browser

The Faceted browser takes a more centralised and less guided approach than the Discovery tool or the Cross-tab Browser. In default mode, the interface displays card-formatted search results, where each card corresponds to a register page (see figure 1.3.1.3-1). As shown in figure 1.3.1.3-2, the results can be filtered according to user-defined selections relating to the plays

performed and the session's revenue. In calendar mode (figure 1.3.1.3-3)—a view that was initially part of the Cross-tab browser—users can see a season-wise revenue heatmap limited to the dates where the performance conditions match their selected filters.

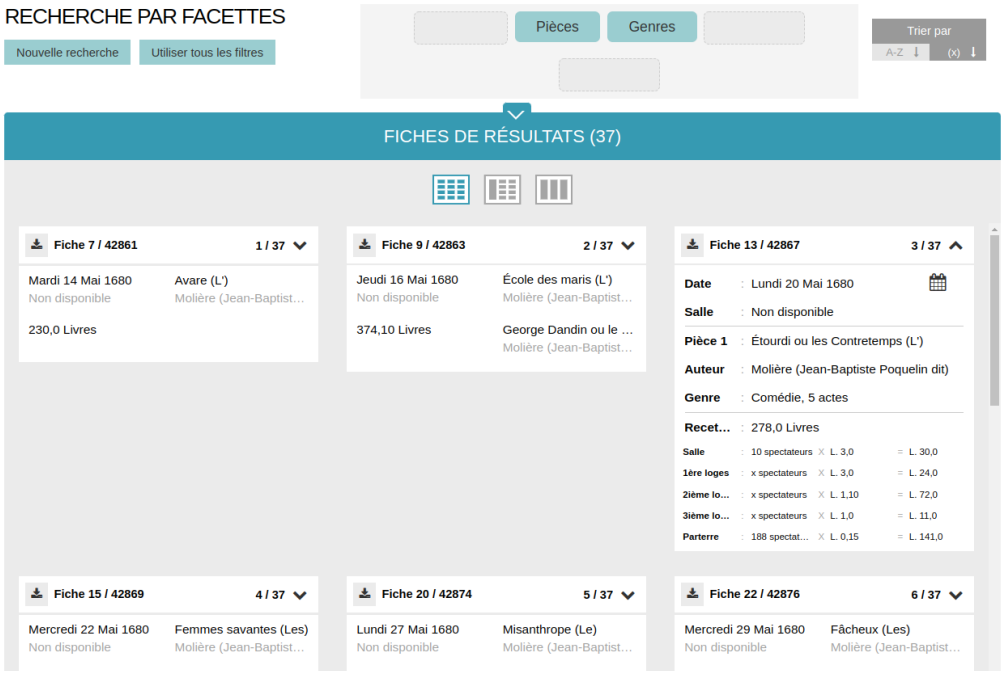

Figure 1.3.1.3-1. Faceted browser - result view

Figure 1.3.1.3-2. Faceted browser - filters

Figure 1.3.1.3-3. Faceted browser - calendar view

1.3.1.3.1 Transparency

The Faceted browser is transparent by virtue of minimal processing. There is a shorter conceptual distance between the registers and the interface's results list than the aggregative graphs of the other tools. Each card in the result view maps to a single page in the receipts registers and, where available, includes a direct link to the digital scan. The information is laid out approximately as it appears in the manuscript, with the date and plays listed at the top, followed by a breakdown of the ticket sales by type. Sales totals, however, are listed exclusively in *livres*, not in *livres*, *sols*, and *deniers* as they are in the source documents. Following in the pattern of the other interfaces, the author and genre of each play have been added to every card, despite not systematically appearing in the registers. The date-ordered card display replicates the way in which the registers flatten hierarchies and highlight complexity. There is no imposed hierarchy, and index filter choices are not grouped or generalised.

1.3.1.3.2 Generativity

The Faceted browser takes the opposite approach to creating space for interpretation to the Cross-tab browser. Overall, its choices in display focus primarily on multiplicity instead of combinatorics. The overlapping filters facilitate quick identification of set intersections that would be slow to calculate manually, but there is no further pre-definition of systems for data organisation and aggregation, or evaluation and comparison. A user is left to create their own procedures for deriving meaning from the data but there is notably no support for undertaking this process integrated into the tool itself.

1.3.1.3.3 Interpretability

The two different modes featured in the Faceted browser provide options for users with varying degrees of knowledge about the data. The bird's eye view presented by the calendar heatmap can help novice users identify potential points of interest, while the grid filters allow users with highly specific questions to zero-in on the data most relevant to them. The filter option lists in grid view react to selections as the user iterates on their request. This means that there is no possibility of confusion over null or empty results, as invalid combinations are simply made unavailable.

There are two instances where the lack of labels could be a potential source of confusion. First, 'x' is used instead of a precise value when specifying the number of spectators seated in the *loges*. While this choice makes sense to users aware of the fact that a single box ticket could correspond to a variable number of people, nowhere is this explained. The calendar also neglects to broadcast the extent of its scope, which is limited by the filters in grid view. Understanding the significance of the data it represents therefore relies solely on accurate recall on the part of the user.

1.3.1.4 Graph Tool

The Graph tool, pictured in figure 1.3.1.4-1, was the first tool to integrate data from the casting registers. It is based on the exploration of the various relationships between actors, roles, plays and authors. By selecting instances of each of the four index types to use as vertices, the user can

build up a force-directed graph based on the connections between them. Hovering over an edge shows details about the relationship—the number of times that two actors performed together, for example, or the frequency with which an actor performed a given role. Clicking on any vertex or on a subset of the edge types brings up a dashboard of visualisations specific to the individual index or index pair connection (see figures 1.3.1.4-2 and 1.3.4.1-3). It is not possible to interrogate groups of three or more indices. Contextual help informs on data significance and processing.

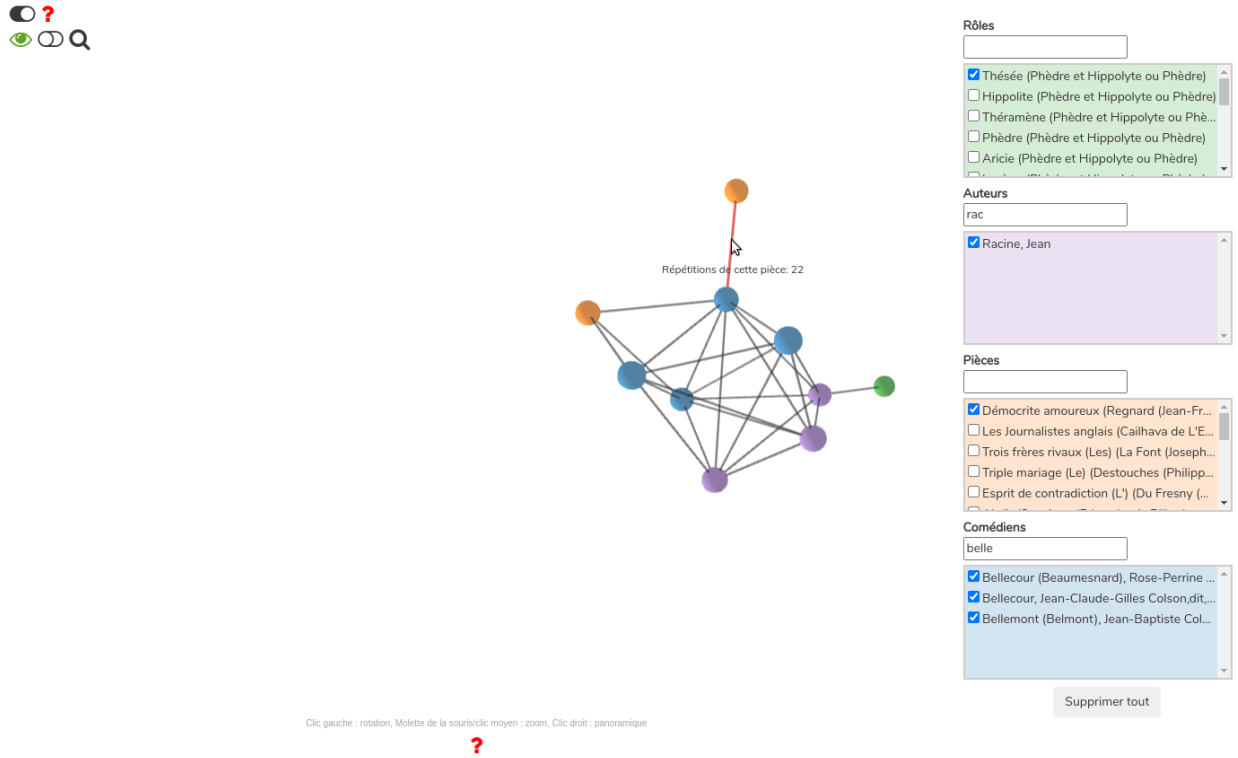

Figure 1.3.1.4-1. Graph tool - graph construction

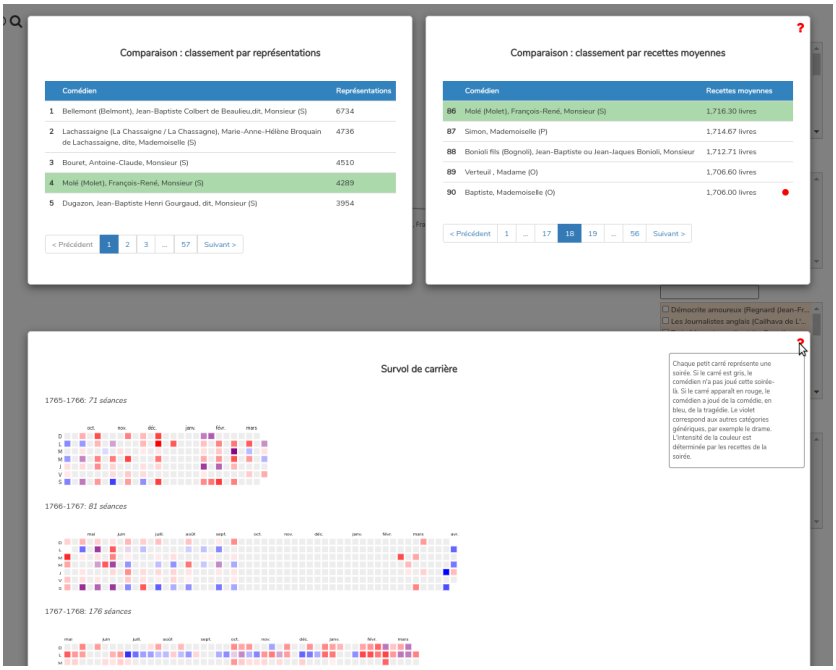

Figure 1.3.1.4-2. Graph tool - limited section of an actor dashboard section (Monsieur Molé)

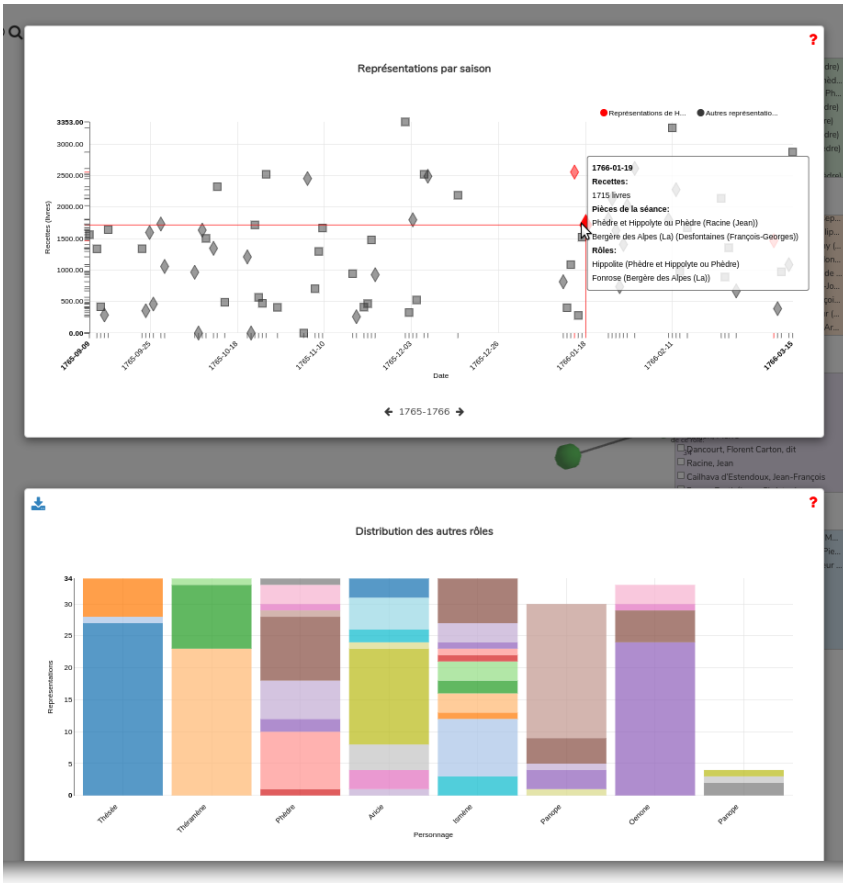

Figure 1.3.1.4-3. Graph tool - limited section of an actor-role dashboard (Monsieur Molé - Hippolyte)

1.3.1.4.1 Transparent

As is the case with the other tools, the Graph tool hides a certain amount of the data processing—all revenue information, for instance, is only given in *livres*—and many of the graphical forms employed are poorly adapted to low-resourced examples. However, the interface consistently includes elements that contextualise and explain data transformations. In lists ordered by averages, for example, red dots flag instances corresponding to single occurrences and the contextual help explains these potentially confusing false averages to the user. In visualisations that in some way feature play revenue, at least the titles of all the plays performed in a given session, if not their full casts, are made available. In cases where an index is assigned a ranking, the total number of options is always listed for perspective.

The graph tool does fall back on 'standard' visualisation forms, but engages with questions of multiple viewpoints and the inherent rhetoric of constructed representations by showing multiple interpretations of the same data, even within a single dashboard element. For example, rather than assuming that it is something users will calculate in their heads given the information, the list of role frequencies featured in each actor's data dashboard includes both the number of times they performed a role, as well as what percent of the time they were the actor who performed that role. The fact that actor Brizard played the titular King Lear in Ducis' translated and gallicised adaptation of the Shakespeare play 38 times, for instance, might seem unremarkable when compared to his 100 performances of Henry IV in Collé's *Partie de Chasse*. However, the knowledge that he was the only actor to ever play the former paints a different picture of the significance of the relationship.

1.3.4.1.2 Generative

The overview graph does not set out to answer any questions. The user is free to add whatever indices are most relevant to them, and the connections and cluster patterns that arise through their combination have the potential to generate new avenues of exploration. What constitutes a connection, however, is predefined and not open to redefinition. Further, the dashboard visualisations are minimally interactive and the configuration of panels cannot be personalised.

1.3.4.1.3 Interpretable

The graph tool notably features two different modes of initial interaction to support different kinds of users. In ‘research’ mode, which is the default, a user adds individual nodes to the graph. ‘Discovery’ mode exists to help users who may be less familiar with the data context, and therefore have no frame of reference for likely relationships, to rapidly build and expand a connected graph. Indices can still be added individually via the lists but, instead of opening the dashboard as it does in research mode, clicking a node in discovery mode brings up a menu of different index collections that could be added (see figure 1.3.4.1.3-1). Clicking the character node Phèdre, for example, gives the user the option to add the node of the play the role belongs to, the node for the author of that play, the nodes of all of the other roles in the play, or the nodes corresponding to all of the actresses who played the role. Each potential option is marked with a traffic-light coded dot to indicate whether or not it would be a ‘good’ idea to select the choice. This interestingly highlights how the graph tool favours HCI values over humanistic ones, as the coding of the options discounts the affective potential of a hairball[82] generated by a dominating presence.

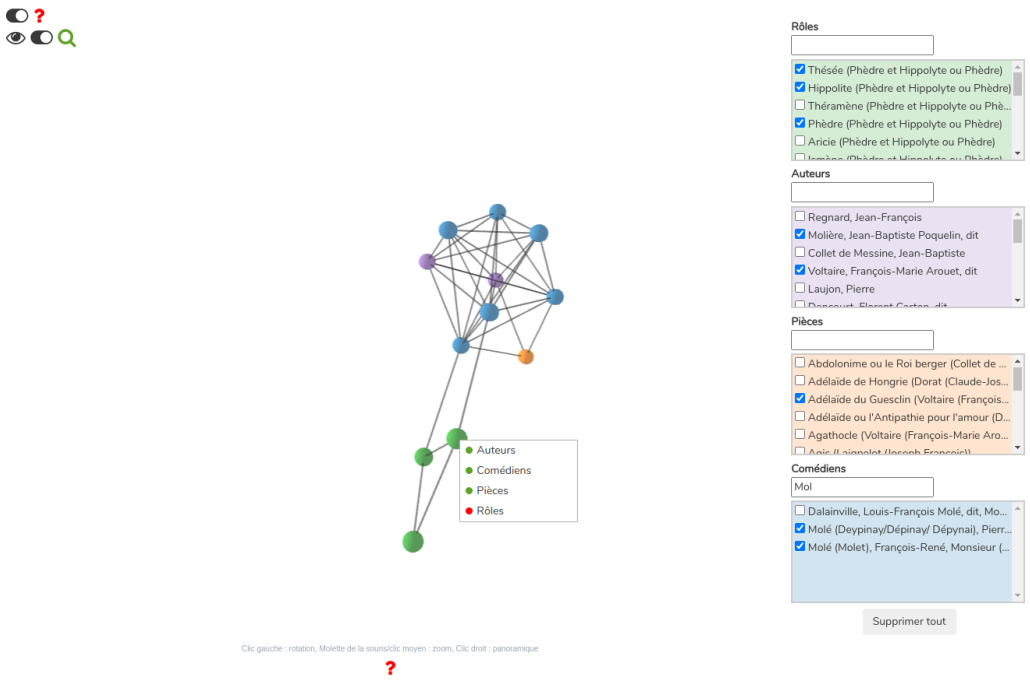

Figure 1.3.4.1.3-1: Graph construction in Discovery mode.

[82] This term is used to describe graphs that are so densely connected that they are difficult to read.

Just as the user can toggle between modes of interaction according to their comfort level, they can easily opt in or out of contextual help, depending on their level of experience with the tool. The contextual help hovers scattered through the dashboard visualisations provide context and guidance about how to use or interpret data. This includes both explicit statements of cues that a user may or may not have picked up on from visual affordances, explanation of how the displayed data was derived, as well as highly context-specific definitions, useful for subject novices.

Chapter 2: CLAIRON

Data interfaces are a site of negotiation between human and computational logics. They bridge the gap between the two, but do not necessarily meet in the middle. There are advantages and disadvantages to interfaces that exist along different points of the spectrum. The further an interface gets from direct interaction with machine processes, the more inviting it becomes to a broader user base, but the more control it must also sacrifice.

Previous work in humanistic interfaces has tended more toward the human logics end of the spectrum. This is notably true of the four existing CFRP data exploration tools, none of which exposes or encourages critical examination of the data or transformations applied to it in order to adapt to a more human interpretable form. As mentioned in chapter one, one problem with using these kinds of data interfaces as scholarly tools is that they start at the end, to an extent. Though degrees of mediation vary, these interfaces framed as exploratory are implicitly structured around answers to a specific set of questions. What's more, in the effort to respond to the needs of as broad of an audience as possible, these questions are often the most obvious ones. The frustration of creating tools for the CFRP is that someone always inevitably asks *but can it do this incredible specific thing?* or *can I cross reference these two obscure elements?* to which the perennial answer is *yes...if you know how to directly query the database*.

This chapter introduces CLAIRON, a proof of concept for a different style of humanistic interface that responds to this desire for the flexibility and degree of detail of a more direct interaction with the data by moving towards the opposite end of the spectrum. CLAIRON is based on natural language interfaces to databases (NLIDBs), which are systems that allow a user to request information from a datasource in 'plain' language, as opposed to query languages that require knowledge of specific syntax and data organisation. However, at the difference of these systems, rather than simply returning a list of results, CLAIRON exposes the query used to fetch them as well as all each step of the natural language to query language translation to the user. In so doing, the aim is to engage the user as an active participant, inviting them to think through and critique how their question is transformed during this imperfect mapping process.

The following sections give an overview the history and current state of the field of NLIDBs, introduce the CFRP data subset used to highlight the specific challenges of applying this style of interface to a humanistic dataset, wade into the weeds of implementation, and finally examine how CLAIRON both measures up against the capabilities of similar NLIDBs as and addresses the core humanistic interface tenets of transparency, generativity, and interpretability

2.1 Natural Language Interfaces to Databases

The development of NLIDBs has been driven by the desire to interrogate data without requiring users to have a knowledge of a specific query language (e.g. SQL, SPARQL) or an understanding of the underlying data organisation or structures. These prerequisites "set 'a high bar for entry' for democratized data exploration"[83]; in light of this, natural language interfaces have the potential to play an important role in promoting more widespread data literacy and transparency. These interfaces, at the most basic level, accept a question posed in natural language and return the results of a corresponding query produced as an intermediate step. The challenge of creating a useful NLIDB lies in striking the balance between a system that reliably provides correct answers and one that can handle questions that vary in structure and, correspondingly, in syntactic complexity and ambiguity. There exist two distinct currents of systems— those structured around a natural language processing (NLP) based 'pipeline', and those rooted in deep learning (DL).[84]

Rules based systems predate learning-based ones, but have progressed alongside advancements in natural language computing and are still prevalent today. Though DL systems have begun to outperform them in automatic evaluations of large datasets, rules-based approaches are favoured by systems which interact with users, as their internal logics are more accessible. Not reliant on training data, they are also more demonstrably made domain-independent.[85] Regardless of type,

[83] Utama, Prasetya, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Çetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, and Arif Usta. "DBPal: An end-to-end neural natural language interface for databases." *arXiv preprint arXiv:1804.00401* (2018): 1.
[84] Baik, Christopher, Hosagrahar V. Jagadish, and Yunyao Li. "Bridging the semantic gap with SQL query logs in natural language interfaces to databases." In *2019 IEEE 35th International Conference on Data Engineering*, (2019): 374-385.
[85] Kim, Hyeonji, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. "Natural language to SQL: where are we today?." in *Proceedings of the VLDB Endowment* 13, no. 10 (2020): 1737-1750.

every NLIDB system must address two core challenges—token mapping and join path inference.[86] Token mapping refers to the problem of establishing a mapping between input tokens and datasource attributes and relationships. Rules-based 'pipeline' systems most frequently start from a lexicon derived from the datasource and rely on linguistic resources such as WordNet[87] to find synonyms for out-of-vocabulary tokens. At an individual word level, deep learning systems tend to use word embeddings to identify probable matches. Join path inference refers to finding the optimal way to unify all of the data required to answer a given question. The need for join path inference arises from the distance between the user's mental model of the data and the true structure of the datasource.

To avoid repetition and facilitate classification, relational databases map components of an entity to be represented into a collection of distinct tables connected by a 'foreign key' that most often consists of a single shared column. This style of decentralised organisation can seem counterintuitive to humanists, and more generally people who have not been exposed to databases.[88] Such users tend to instead picture all of the information as being stored in a single location—a single table—and their questions correspondingly do not explicitly specify the way in which all the tables required to access all of the requested information should be linked; this is up to the system to infer.

'Pipeline' NLIDBs broadly fall into four categories: keyword based, pattern based, parse-based and grammar based—with only the latter two able to handle complex query structures. Grammar based systems use a context free grammar (CFG) to define and constrain possible user inputs. Though this severely limits scope and flexibility, this approach is favoured by systems who aim to scaffold query construction, with the grammar leveraged to suggest valid terms and structures. Parse-based systems focus on understanding grammatical structures and are the class typically best at handling more verbose queries with nuanced relationships between linguistic components.[89] The architecture of grammar or parse-based systems generally includes three core components: question structure analysis, which looks to understand the relationships between

[86] ibid
[87] Miller, George A. "WordNet: a lexical database for English." *Communications of the ACM* 38, no. 11 (1995): 39-41.
[88] Quamen, Harvey, and Jon Bath. "Databases." in *Doing Digital Humanities*, (Routledge, 2016): 181-198.
[89] Affolter, Katrin, Kurt Stockinger, and Abraham Bernstein. "A comparative survey of recent natural language interfaces for databases." *The VLDB Journal* 28, no. 5 (2019): 793-819.

components; token matching, which seeks to map input tokens onto database elements or, in the case of relationships, query language keywords; and query generation, which transforms an internal structured representation of the matched tokens into a valid query.

PRECISE is an early NLIDB implemented as a demonstration of Popescu et al.'s theorisation of the category of 'semantically tractable' questions that an NLIDB should be able to reliably answer.[90] To belong to the class of semantically tractable questions, each input token must match to a single unique database element, each token that matches to an attribute (column) must have a corresponding value (row-cell) token, each token denoting a relationship must be attached to either a value token or an attribute token, and the input must contain a 'wh' word—'what', 'which', 'where', 'who', or 'when'. In the case of viable questions, after PRECISE retrieves all possible element matches for each token, the constraints imposed by semantic tractability allow the system to treat token-element matching as a maximum-bipartite-matching problem and to find the optimal path—the most likely semantic interpretation—using the max-flow algorithm. In experiments on three benchmark datasets, approximately 80% of the questions were found to be semantically tractable. PRECISE was able to handle these with high accuracy, but would not address the remaining 20% unless the user provided a paraphrase. In addition being limited to a small set of possible structures, it notably cannot handle words that are outside of the vocabulary of the database, nor any tokens that imply a calculation or combination of fields. For instance, unless it exists as a pre-calculated field, PRECISE is unable to handle a concept like 'density'.

ATHENA[91] and SQLizer[92] are both NLIDBs that add an intermediate level abstract representation to the NL to query translation process. ATHENA does not map input tokens directly to database elements but rather to elements in a pre-defined domain-specific ontology. Though this does not permit the same context agnosticity as PRECISE, it does allow for more complex relationships between tokens and aids with token disambiguation. This notably includes derived fields that do not map to a single database element but are able to be expressed by a

[90] Popescu, Ana-Maria, Oren Etzioni, and Henry Kautz. "Towards a theory of natural language interfaces to databases." in *Proceedings of the 8th international conference on Intelligent user interfaces* (2003): 149-157.
[91] Saha, Diptikalyan, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, and Fatma Özcan. "ATHENA: an ontology-driven system for natural language querying over relational data stores." in *Proceedings of the VLDB Endowment* 9, no. 12 (2016): 1209-1220.
[92] Yaghmazadeh, Navid, Yuepeng Wang, Isil Dillig, and Thomas Dillig. "SQLizer: query synthesis from natural language." in *Proceedings of the ACM on Programming Languages* 1, no. OOPSLA (2017): 1-26.

single ontology element, making them available as possible matches. Using the semantic information provided by the ontology, ATHENA creates a ranked list of valid semantic interpretations—graphs that necessarily include all tokens, are weakly connected, and do not include or lack any connections which contravene ontological hierarchies. The system models join path inference as a Steiner tree problem. SQLizer's additional layer of processing looks to simplify query generation through the use of templates. Working from a basic semantic parse that includes linguistic metadata such as part-of-speech and entity tags in addition to structural information, SQLizer generates a generic SQL 'query sketch' with slots to be filled by database entities and join paths. Type information is used to narrow the field of possible matches alongside the names of schema elements, key constraints, and row values. The system iteratively tests out different likely token combinations, using predefined repair rules—grammar defined structural transformations—to expand and alter the query in the case of low-confidence matches.

Rather than attempting to guess the correct matches for ambiguous tokens, Li and Jagadish's interactive NLIDB, NaLIR,[93] simply asks the user for clarification. Starting from a dependency parse, this system groups the token nodes into classes corresponding to different query components. For nodes likely referring to database elements, it uses a Wordnet synonym and hypernym relationships as well as structural constraints to identify candidate matches but ultimately relies on user interaction to select the right mapping when other metrics are not enough. NaLIR uses a grammar to check the validity of parse trees corresponding to possible semantic interpretations with respect to path constraints and query structure, making adjustments where possible and discarding non-viable options. This process notably includes the insertion of implicit nodes needed to create valid join paths. Join paths are ultimately generated using the key relationships between all the tables of the database elements involved in the query. NaLIR's performance is highly dependent on human feedback, something which its later extension, TEMPLAR, aims to mitigate, to an extent, through a semi automation of user interaction.[94] TEMPLAR, which sits on top of NaLIR, reuses information from query logs corresponding to inputs of similar structures to identify the most likely token matches and ideal join paths.

[93] Li, Fei, and Hosagrahar V. Jagadish. "Constructing an interactive natural language interface for relational databases." in *Proceedings of the VLDB Endowment* 8, no. 1 (2014): 73-84.
[94] Baik et al., "Bridging the semantic gap with SQL query logs in natural language interfaces to databases"

Deep learning NLIDBs leverage training data composed of NL-SQL question and answer pairs to learn appropriate translations, much in the same way neural translation systems between natural languages rely on parallel corpora. The architecture of these systems is commonly Recurrent Neural Network based and frequently augmented with some sort of attention mechanism.[95] When it comes to token matching, DL systems are better than 'pipeline' systems at handling complex relationships and derived or abstract fields. As long as it appears in the training data, a neural system can come to 'understand' conceptual tokens like 'major cities' and correctly map them to potentially complex query fragments. Join path inference is likewise scoped by the samples which the system has ingested. It's no mystery that the biggest limitation of these systems is the availability and quality of training data. Since hand annotated examples are onerous to produce, recent attempts have been made to create and augment training sets through automatic techniques. DBPal,[96] for example, is trained on a dataset created using slot filling, which consists of creating examples by filling in question-query template pairs with a variety of database elements of the appropriate types. Its training set was further expanded using automatic rephrasing, which draws on language models coupled with synonym retrieval using linguistic resources such as WordNet to produce paraphrases of existing natural language questions. While more efficient, these generative techniques have important drawbacks. Training sets generated by slot filling will only ever expose the system to a limited number of query language formulations. Rephrasing can similarly produce examples that, while technically grammatical, are distinctly different from naturally elicited questions. One of DBPal's templates 'Show me the <attribute> of the <table> with <filter>' and its rephrasing 'For all the <table> with <filter>, what is their <attribute>', for instance, yielded the questions 'Enumerate the names of all the patients with age 80' and 'For all of the patients with age 80, what are their names?'. Though true to the researchers' goal of increasing structural diversity, the system's lack of syntactic nuance—not knowing age, for instance, is something that someone *is* rather than *has*—resulted in distinctly un-natural questions.

2.2 A Humanistic NLIDB

[95] Kim et al., "Natural language to SQL"
[96] Utama et al., "DBPal"

The point at which modern technology is often considered to have reached its peak is when it has been naturalised to the point of near invisibility.[97] The layers of programmatic abstraction necessary to create this illusion, however, obscure the assumptions and simplifications performed by these interfaces, or unconsciously by a user, in the course of mapping a human question to a form which aligns with computational logics. Consequently, users are not required to confront the epistemological friction between their questions and the interface's answers, or the influence the interface's formulation of their question may have on their interpretation of the results.

As Micheal Mateas argues, programming, and by extension computing, is about describing process and complex flows of cause and effect—an act of translation that will never be frictionless[98]. Bringing an explicit exploration of this friction to the fore is, in my view, essential to instantiating a humanistically oriented NLIDB. The aim of showing not only the results, but the process, and highlighting the equivalencies and by extension, differences in formulation between the natural language and SQL versions of an input, is to guide the user to reflect critically on this imperfect process of translation and to consider the ways in computational logics impact and transform their research and understanding. Ease of interaction is not sacrificed, since translation is still taken care of by the machine, but the user is actively engaged in the process—deconstructing, questioning, and correcting it. Though following in the logic of existing NLIDBs in that it allows a user to interact with the database via NL questions, without requiring either a knowledge of the database schema or of query language, CLAIRON—perhaps paradoxically—looks to encourage users to build an understanding of both.

2.2.1 Design Context

Three key implementation choices grounded in the theorization of humanistic interfaces define the design of CLAIRON:

1) The choice to use a parse-based system over a grammar based system or a deep learning based system - Given the goal of interface transparency, opting for an NL pipeline style system was an

[97] Cellard & Masure, "Le design de la transparence"
[98] Mateas, Michael. "Procedural literacy: Educating the new media practitioner." *On the Horizon* 13, no. 2 (2005): 101-111.

obvious choice. Deep learning systems are black boxes, and the ability to illustrate the process was an essential consideration.

2) The choice of SQL and a relational database over SPARQL and an RDF repository - Though known today as SQL (Structured Query Language) the language used to query relational databases was initially conceived as SEQUEL—Structured English Query Language.[99] Its authors chose to use English keywords and a sentence-like structure to "reflect how people use tables to obtain information." They emphasised that SEQUEL was specifically designed to accommodate a "new class of users", whose work utilised computers but who were not experts. SQL was built assuming a familiarity with, and ability to draw parallels with, spreadsheets. SPARQL (SPARQL Protocol and RDF Query Language), on the other hand, was designed to interact with data collections specified using RDF (Resource Description Framework), a graph-based architecture with which fewer non-experts are familiar. The intent of the interface is to expose the guts of the process, including the underlying data structures, and to invite users to inductively come to understand the problem deconstruction and mapping. Discovery based learning of this type is scaffolded by prior knowledge of the world,[100] so the choice of a relational model was driven by the fact that users are more likely to have prior knowledge of, and comfort with, spreadsheets as opposed to networks.

3) The choice to leave input unguided - Many NLIDBs leverage user interaction to help resolve ambiguity. NALIR,[101] for example, relies heavily on the user to pick between potential token matches, and the system has been shown to be significantly less effective when run in non-interactive mode.[102] In a similar vein, the primary advantage of grammar-based systems is the ability to suggest valid query components to the user. These systems follow in the same logic as many query building interfaces such as Sparnatural,[103] whose potential inputs are limited to fill in the blank templates. Such systems implicitly guide the user to create an easily machine

[99] Chamberlin, Donald D., and Raymond F. Boyce. "SEQUEL: A structured English query language." in *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control* (1974): 249-264.
[100] de Jong, Ton. "Scaffolds for scientific discovery learning." *Handling complexity in learning environments: Theory and research* (2006): 107-128.
[101] Li & Jagadish, "Constructing an interactive natural language interface for relational databases"
[102] Utama et al., "An end-to-end neural natural language interface for databases"
[103] https://github.com/sparna-git/Sparnatural

interpretable question. Their templates and grammars result in inputs that, while technically 'natural' language, are not typically the type of question that would be naturally elicited, as their components are meant to provide the cleanest mapping to query blocks. Though useful for ensuring queries are valid, this approach directly contradicts the centering of the differences between human and computational problem expressions. Instead, the system performs these adjustments implicitly, without reflection of awareness on the part of the user. Input left unguided, to the contrary, asks users to explicitly confront the ways in which their questions must be altered in order to fit computational logics.

2.2.2 Data Context

Fundamental to humanities data methodology is the notion of data as capta.[104] DH scholars recognize that data is deeply contextual and promote practices that, from collection through to dissemination, focus on the conditions and contexts from which it emerged, and the ways in which those conditions shape and inform all manipulation and interpretation. Catherine D'Ignazio offers up the creation of detailed 'data biography' as a crucial first step in any humanistic data-oriented project.[105] This reflexive documentation asks the researcher to identify the how, by who, and why of data collection and analysis, and to take stock of the potential negative impacts of the data and its limitations. We recognize that data is always "is always an intentional simplification of a more complex and rich reality"[106]; the data biography exists to help users make sense of that simplification process, showing them how to read for data gaps and biases introduced in the mapping. Though primarily targeting datasets destined for machine learning, Gebru et al.'s 'datasheets'[107] follow in the same vein. This slightly more detailed formalism comprises seven sections—motivation (why was the dataset created?), collection (who created the dataset and how?), composition (what is and isn't the dataset representing?), processing (what cleaning, labelling, or derivation was part of the creation of the dataset?), uses (who has and will use the dataset, how, and what potential negative impacts could it have?), distribution (who will be given access to the dataset, how, and under what conditions?), and maintenance (will the dataset be maintained and, if yes, how and by whom?).

---

[104] Drucker, "Humanities approaches to graphical display"
[105] D'ignazio, "Creative data literacy"
[106] ibid
[107] Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. "Datasheets for datasets." *Communications of the ACM* 64, no. 12 (2021): 86-92.

It would be counterproductive to embark on the creation of a humanistic data interface without engaging in this process of reflection. The following sections comprise the data biography of the custom dataset which I derived to use as an example for the NLIDB. They address, in turn, the datasheet questions pertaining to the first four proposed aspects, the latter three being outside the scope of the current project.

2.2.2.1 Motivation

The interface, like all NLIDBs, is intended to be able to work with any database. However I chose to create a new database fusing a subset of the CFRP data (*recettes* and *feux* - the two complete datasets) for three distinct reasons.

First, I needed a single datasource that matched the scope of the submitted questions. The archive-per-phase approach of the CFRP means that the *recettes* and *feux* register sets were initially transferred into two distinct databases by two different teams. The interface requires a single datasource, hence the creation of a merged database.

Second, I wanted a datasource that, as much as possible, would exemplify the specificity of relational database organisation. Despite their similarities and the way that the terms are often used interchangeably by DH practitioners, databases are not spreadsheets. The logic of databases looks to maximise access to different points of entry into the data, and to minimise data duplication and wasted space. For example, when storing data about plays, a spreadsheet might list each unique play and list duplicate author details every time. A database, on the other hand, would typically treat the authors and plays as separate entities, placing them in different tables and linking them together through a numeric identifier. This both reduces data duplication and changes the focus of the data to place authors and plays on equal footing—with inquiry based around either just as easy—which is not necessarily true of the spreadsheet. Recognizing this organisational and conceptual difference is an important part of the interface's goal of encouraging users to think through the cognitive distance between human and computational expressions of a question and to interrogate the ways in which technical mediation shapes and alters inquiry. In light of this, it was important that the underlying database used to demonstrate the NLIDB embody relational database best practices, as its schema is fully exposed. Both the

feux and *recettes* databases bear the marks of a continuous development process that worked to adapt to the needs and interests of the researchers as their understanding of the data grew. As alterations were being made on the fly, some were not necessarily integrated into the predefined schema in accordance with ideal database design standards. The fused database, having the advantage of retrospect, aims to improve the integration of these ad-hoc additions.

Third, though not functionally necessary, I wanted a database whose naming conventions reflected domain vocabulary. English has become somewhat of a technical *lingua franca*; very few programming languages support non-english keywords and, correspondingly, the tendency is to use english variable names, even in non-english contexts. The fused database deviates from this tradition, one that was perpetuated by both of the databases from which it draws, instead using French names for all database elements. This choice was made to respect the precision of domain specific vocabulary and, to a lesser and unnecessary extent, to streamline entity matching.

2.2.2.2 Collection

In terms of datafication, we might consider the data to have been collected at two different points. Historically, the register data was recorded by the very people to whom it relates. A rotating roster of actors would take on the role of *semainier*, whose job it was to record the evening's performance data. In the modern context, data transcription was performed by undergraduate and graduate students in French, Literature, and Theatre programs who were employed as research assistants, and verification was performed by the most experienced students, doctoral researchers, and professors.

Data input for both the *recettes* and *feux* registers was done using bespoke data interfaces, though the respective tools were created by two different developers and the transcription likewise undertaken by two different teams. Datafication practices as they relate to manuscript records, in general but notably specifically with respect to similar theatre records, tend to adopt one of two approaches: focusing on entities or focusing on page elements. The data for RECITAL project,[108] for example, who are working on the registers of the Comédie-Italienne, is organised around

[108] https://recital.univ-nantes.fr/

‘annotations’. Each entry in the central table of the database is first and foremost conceived of as an annotation on a page that is defined by size and coordinate information. This means that, while they are identifiable through a tag field, actors, for example, are not an easily interrogable index of the data as they are not an entity in the database schema. This style of data organisation is very flexible, as it is not beholden to any sort of predefined rubric, but might also be considered to be more detached from the data context. The other approach consists of creating a predefined data ontology. This style is much more reliant on content knowledge.The advantage of this approach lies in easier interrogation of facets of interest, but it could be seen to more readily encode researcher worldview and lose aspects of the data deemed unimportant by a few individuals. The CFRP chose the latter approach. The only data entered during the transcription phase corresponded to what was written on the register page, with the structure and fields decided in consultation with experts. That data was then further used to build up tables of indices identified as important by the researchers—authors, plays, seating categories, etc.

2.2.2.3 Composition

The registers of the Comédie-Française are a detailed record of all of the company’s day to day activities, kept meticulously from its foundation in 1680 through to today. The fused database is a combination of two of the four archives dating from the 17th and 18th centuries—the *recettes* and *feux* . The recettes database spans from 1680 to 1793 and includes ticket sale data pertaining to 34435 dates. Though expense and approximate casting information was sometimes present in the receipt registers before being relegated to discrete registers, this data was not captured during the datafication of the recettes. The feux registers only span from 1765 to 1793. These registers capture casting data for 8462 dates, notably including 128 not seen in the recettes data, as they pertain to performances given outside of the company’s Paris theatres.

There are fifteen tables in the fused database (see figure 2.2.2.3-1).

1) *séances* (34537 rows) - Each entry represents a single session (date). Typically, two plays were performed each day.
2) *représentations* - Each entry corresponds to a performance of a single play.

3) *pièces* (1059 rows) - Entries offer details about the plays performed at the CF—genre, length, premiere date etc.
4) *auteurs* (312 rows) - Entries give detail about the authors. This information is notably not listed in the registers except in the case of new plays. It was added to the *recettes* database by content experts.
5) *attributions* (1075 rows) - Entries correspond to authorship credits. This information is encoded as a junction table rather than as a single foreign key field in the plays table because of cases of co-authorship.
6) *comédiens* (566 rows) - Each entry gives information about a specific actor. As with authors, much of this data came from sources outside of the registers.
7) *débuts (*260 rows*)* - Each entry associates an actor with the year of their formal debut.[109] As with authorship, this information is formatted as a junction table because of the possibility of multiple debuts.
8) *rôles* (4981 rows) - Each entry gives information about a role, and each role is associated with a specific play.
9) *interprétations* (152208 rows) - Each entry in this table represents the performance by an actor of a specific role on a given evening; in other words, each row associates a *représentation* (single instance of a play being performed) with an *acteur* and a *rôle*. It is possible for one of the latter two to be null.
10) *lieux (25 rows)* - Each entry corresponds to a performance location. Data about 'special locations'—visits to court or to the country, for example—comes directly from the registers, and entries retain the noted level of granularity.[110] Additional rows correspond to the four different theatres[111] the troupe occupied between 1680 and 1793

[109] An actor's formal début was a two week period during which they were afforded the chance to prove their talent by playing a number of roles of their choice, often corresponding to a specific character archetype (which came to be known as an *emploi*)—*reines, soubrettes*, *rôles à manteau* or *confidents*, for example. After their two weeks were up, the company's governing body of *sociétaires* would either recommend to the Royal bureaucrats who were officially (though mostly symbolically) in charge of the company that the new actor be admitted *à l'essai*, meaning that after a trial year they could potentially be inducted to the company as a full *sociétaire*, or judge their attempt to be unsuccessful. Many actors who were initially refused would go gain experience in provincial theatres before returning to Paris for a second attempt. It is therefore possible for an actor to have multiple entries in the *débuts* table

[110] For example, *Versailles* and *Trianon* are noted as two different locations, despite one being inside the other, and a distinction is maintained between *Fontainebleau* and *Fontainebleau, au théâtre de la ville*.

[111] Hôtel Guénégaud (1680-1689), Fossées-Saint-Germain-de-près (1689-1770), Palais des Tuileries (salle des machines) (1770-1782), Théâtre de l'Odéon (1782-1793).

11) *places* (148 rows) - Each row in this table details a different seating category type. These changed over time, primarily as a function of theatre architecture.
12) *ventes* (181850 rows) - Entries in this table link together a *séance* and a *place*, detailing how many tickets of that type were sold, at what price, and how much revenue that brought in. Though it is generally bad practice to save fields in a database which can be calculated, as is the case with revenue, these figures were kept for two reasons. First, it is more true to the source document and, second, in ~3.27% (5949/181850) of instances, clerical errors or exceptional circumstances—both of which are of interest to researchers—mean that the numbers do not match up.
13) *registres_feux* (8462 rows) - Though séances already covers dates, in order to preserve a trace of the multiple data sources, each entry in this table corresponds to a page in the casting registers. Each one is linked to a *séance* and gives details specific to the *feux* archive.
14) *registres_recettes* (34409 rows) - This table serves the same function as *registres_feux*, except the information relates to the receipts registers. In a similar fashion, each entry is linked to a *séance*. That said, each *séance* won't necessarily have both associated *feux* and *recettes* pages; many have only one or the other.
15) *pages_de_gauche* (4882 rows) - This table tracks what kind of information was found on the left hand page of the receipts registers (the details of the performance are always on the right). Though this table is not as directly related to drawing the data picture of an evening at the CF, it may nonetheless be of interest to those researching administrative practices.

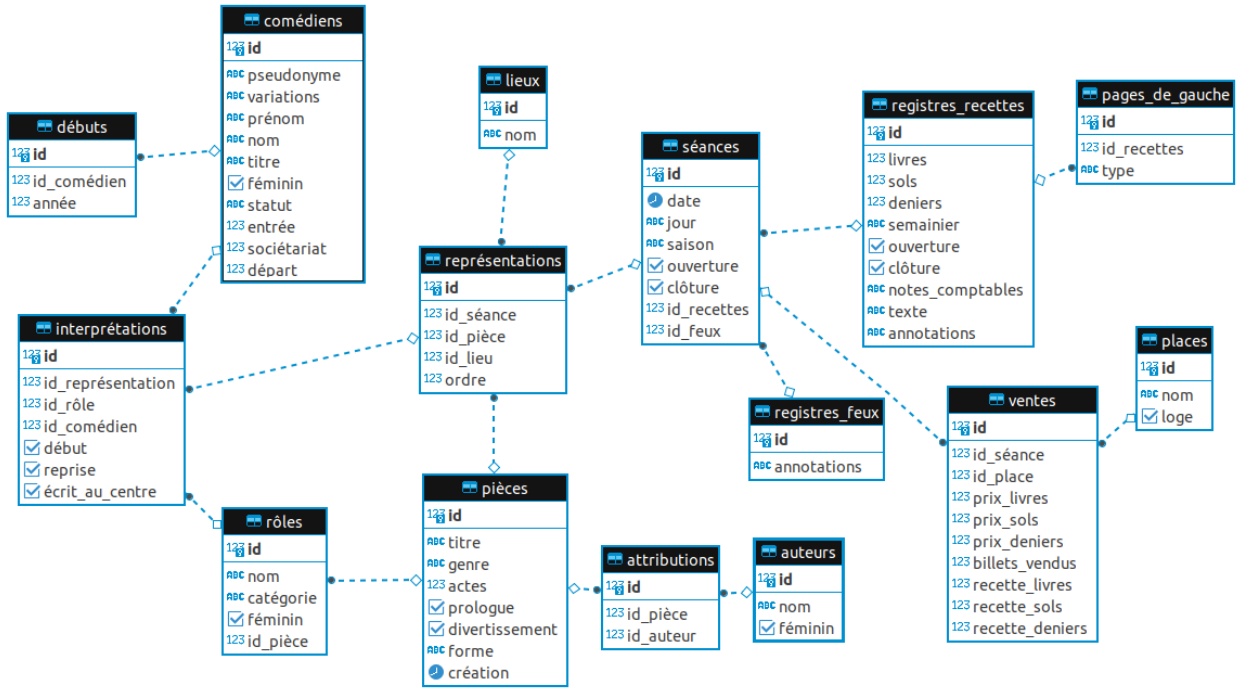

Figure 2.2.2.3-1. The fused recettes-feux database

There are a few limitations inherent to the data. The most significant is that there is no way to determine the financial success of a single play. Except on very rare occasions, each evening's performance included two plays—typically a comedy and a tragedy, or longer comedy and a shorter one. Tickets were sold for performances, not plays, so the registers consequently do not provide individual play revenue information. Knowledge of exactly how many people were in the room is likewise limited by ticket sale ambiguity. Unlike many seating categories, the *loges* (boxes) didn't have a 1-1 person-to-ticket ratio. Even with ticket sale numbers, the capacity of a loge was not always known (or stable) so to know from the register data can only ever support an informed guess at the true makeup of the audience.[112] Finally, though we know it is not binary, the registers only ever provide binary actor and role gender information.

[112] Velde, François R. "An Analysis of Revenues at the Comédie française, 1680-1793." in *Données, recettes & répertoire: La scène en ligne (1680-1793)*. MIT Press, 2020.

2.2.2.4 Processing

A good portion of the collected register data directly maps to what was written on the page. In the course of the original data entry, extra information added describing actors, authors, and plays was drawn from contemporary sources, but these additions are nonetheless possible sights of introduction of researcher bias. Play genre is notably only listed with any degree of consistency when a play debuts. This means that many genres listed in the plays table had, for the most part, to be pulled from contemporary scholarly publications or periodical press. In some cases, however, even these sources disagreed; Marc Antoine Legrand's *Plutus*, for example, is listed as a *comédie* in the Chevalier de Mouhy's *Tablettes Dramatiques*[113] but called a *vaudeville* when it is first read in the actors' *assemblée*.[114] In the database it is listed as a comedy. For all that it is representative of *a* contemporary ontology, this decision could be called into question given that it aligns more with the dominant narrative style of history that the CFRP actively tries to deconstruct than with the vision of the author (who was one of the actors). The same logic of choosing the dominant form was applied to actor and author names, though in these cases, alternative forms were retained and noted in an additional field. The boolean *divertissement* field in the plays table is another simplification. A *true* value flags that a play included something outside of simple recitation—specifically dance, music, or machinery—but the database doesn't go into any further detail. Additionally, plays which at some point may have contained these elements were at others performed without them, due to cost or limits imposed by formal legislation.

Most of the data processing I performed in the creation of the fused database was cleaning to assure consistency. It was not so much scrubbing as tidying up—removing entries that were clearly tests, condensing tables, replacing accidental empty strings with nulls, and formalising foreign key links that had not been properly established in the *recettes* database. As the aim of the interface is to answer questions about the data itself, I removed all tables and fields relative to team members and work logistics. In two instances I augmented the data with outside information: I added author gender information from a list compiled by an expert researcher on the project, and supplemented the location information to include the Paris theatres as locations

[113] Mouhy, "Tablettes Dramatiques"
[114] https://flipbooks.cfregisters.org/R52_6/index.html#page/103/mode/1up

so that every performance had an associated *lieu* rather than only unusual cases. Additionally, actor debuts were shifted from being an array field to a junction table.

There are a few database elements ported from the *recettes* that are not representative of best practices that I chose not modify for reasons of overhead. The first is that there is no way of identifying the *comédiens-poètes*, which is to say no link between actors and authors who are one in the same. The second is that *semainiers* is a plaintext field, so there is no key link to any people entries; solving this would demand the creation of a more inclusive people index as, early on, the semainiers were sometimes company personnel (accountants, prompters) rather than actors. The *semainers* question is also complicated by the fact that actors would sometimes sign in the name of their colleagues, something we only know through additional annotations.

2.2.3 Scoping

Though the NLI pipeline approach does not require training data as machine-learning (ML) systems do, my first step in implementation was nonetheless question generation. The goal of collecting a list of potential questions that someone might ask of the CFRP data was to help scope the project. In examining the different types of questions, I was able to abstract upward to determine what structures needed to be handled, and build an understanding of the various ways in which different SQL operations could be expressed in natural language.

It was above all important to me to ground the interface in naturally formulated questions from people with diverse perspectives. A frequent issue with ML model training sets or pipeline model test sets is that they cover only a limited number of phrasing variants and/or include specific formatting that was added to aid machine interoperability. For example, one of the benchmark datasets used to evaluate NaLIR always encloses multiword expressions in quotation marks, so as to signal the parser to interpret them as a single token (e.g. What year is the movie "Dead Poets Society" from?). A second benchmark consists entirely of queries that are all very similarly phrased—each one begins with ‘return to me’ then requests a specific entity with all filtering at the end (e.g. return me all the organizations in "North America"), and the awkward phrasing of some examples suggests that the vocabulary used maps directly and exactly to database entities (e.g. return me the area of PVLDB). As previously seen, the use of synonym

substitution or automatic rephrasing to supplement datasets can similarly result in maladroit examples.[115]

Hall et al. argue that immersion in both the technical and the content domain is essential in interdisciplinary computing projects, so as to address "challenges introduced by gaps in both knowledge bases and cultures."[116] Though designing for dataset agnosticism, the aim was nonetheless to create an interface able to handle specifically humanistic questions. As not an expert in eighteenth century theatre nor a humanities scholar by training, I would never claim to know the types of questions that would be most useful to a humanist, nor the most natural language for expressing them. Similarly, I am aware that my knowledge of the way in which the data is organised would colour any attempts to write questions. Finally, at the difference of previous NLIDBs, I chose to handle French rather than English input. This meant that an authentic set of test questions necessarily needed to come from native French speakers. In light of these considerations, I solicited the help of the core CFRP team, who are both subject experts and francophones. The team have varying levels of knowledge of the database contents and structure, but I specifically requested that they not limit themselves to only questions they thought would be answerable. Their interests span diverse facets of the data and this is reflected in their responses. To ensure coverage of core sql concepts, I hand-augmented their submissions with lightly modified rephrasings.

In addition to scoping what would be possible, the questions were also essential in identifying what the system wouldn't or shouldn't handle. One of the primary interface goals is to bring to the forefront what the computer doesn't understand, rather than coping with uncertainty by making and obfuscating assumptions about the data. This entails explicitly choosing not to handle obviously ambiguous terms. In order to be able to answer the question *Quelle est la pièce la plus populaire de Beaumarchais ?*, for instance, the system would need to implicitly decide on a definition of 'popular' which, in this particular case, could equally refer to profit or attendance. This kind of hidden processing is exactly what CLAIRON's design is intended to avoid. When

[115] Utama et al., "DBPal"
[116] Hall, Kyle Wm, Adam J. Bradley, Uta Hinrichs, Samuel Huron, Jo Wood, Christopher Collins, and Sheelagh Carpendale. "Design by immersion: A transdisciplinary approach to problem-driven visualizations." *IEEE transactions on visualization and computer graphics* 26, no. 1 (2019): 109.

an interface makes these kinds of decisions for the user, it runs the risk of confusing them by giving a result that clashes with their mental model or, more importantly, colouring or altering their line of inquiry by presenting one interpretation of many as an objective truth. An input such as *Quelles sont les vedettes que le public vient plus nombreux pour voir jouer?* opens up additional questions in this same vein. This query assumes that the system has an understanding of the term *vedette*, which is a rather lofty assumption given that the notion of *vedettariat* was newly emerged and quickly evolving in the eighteenth century, and its definition correspondingly fluid. Born in part of the rise in popularity of first-person writing and the expansion of the periodical press, the relative stardom of any given actor was dependent on a number of factors extrinsic to the data captured in the registers.[117] An expert user might mentally equate the label with the likes of Lekain, Talma, Lecouvreur, or Clairon, but the system has no frame of reference for what distinguishes them from their fellow actors, or even that *vedette* might refer to an actor. Expressing a question or problem computationally is an act of translation. As with natural languages, the translation process involves mapping the knowledge, structure and logics of one language to those of the other. From this follows the way forward for cases like *vedette*, where a one-to-one mapping doesn't exist. How do we translate untranslatable words or concepts? We talk around them. Many data interfaces, in similar situations, fall back on preprogrammed hidden definitions which may not even be expert defined in the aim of streamlining user interaction. For CLAIRON, I instead opted to refuse to handle this type of question and pinpoint the problem term so as to highlight the limits of computational interpretation as well as prod the user to specify what *they* mean by *vedette*. Questions relating to who plays major roles, who brought in the most revenue, or even who got to originate the most roles could all help paint the picture of who the biggest stars of the time were, but these definitions need to be formulated by the user.

2.2.4 Query to Question Translation

The back end architecture of the CLAIRON NLIDB comprises four steps—parsing, matching, query tree construction, and query generation. During parsing, questions in natural language (French) are broken into linguistically distinct tokens and structured as a tree according to their dependencies. The matching step uses the database structure and lexicon to map relevant tokens

[117] Filippi, Florence and Sara Harvey. "Émergence du vedettariat théâtral en France (xviie-xixe siècles)" in *Le Sacre de l'acteur. Émergence du vedettariat théâtral de Molière à Sarah Bernhardt.* (Paris, Armand Colin, « Collection U », 2017): 11-26

to tables, columns, or row values according to linguistic evidence and context clues. During query tree construction, an index of keywords is leveraged to help infer query logic from input tokens and to add nodes to the tree that represent the corresponding SQL operations. Finally, the query generation step traverses the tree to piece together the final SQL query.

2.2.4.1 Data Representation, System Configuration, and Input Parsing

Existing NLIDBs model data directly according to database structure or require a user-defined ontology to map database elements to domain specific terms and specify the relationships between them[118]. The advantage of the latter approach is that ontology can provide rich semantic information that may have been lost in RDB schema structure. The ATHENA system notably maps NL input to an intermediate ontology query language, and it is that intermediate representation that is then translated into SQL, so as to "decouple the physical layout of the data in the relational store from the semantics of the query, providing physical independence."[119] While the advantage of being able to specify complex relationships is undeniable, especially with regard to humanities data, I wanted to avoid adding a level of abstraction. A primary goal of the interface is to be as transparent as possible about the processes and capabilities of the computer—something which layers of abstraction only serve to obscure.[120] As a sort of compromise, the data representation mechanism used by CLAIRON includes an optional lightweight configuration which allows the user to establish mappings between database elements and domain specific vocabulary and, to a limited extent, define, name, and integrate more complex concepts that integrate calculations or field combinations. For each table, an expert user with knowledge of the database schema can provide a list of synonyms for the table name, and for the names of any of its attributes. Some systems do this automatically by leveraging linguistic resources such as WordNet to fetch synonyms and well as reduced or extended forms,[121] or use pre-trained word embeddings to identify probable synonyms. However, as mentioned earlier with regard to slot filling, this can yield some questionable results. What's more, these general purpose resources are likely to perform less well with regard to the hyper

[118] Kim et al, "Natural language to SQL"
[119] Saha et al., "ATHENA", 1209
[120] Cellard and Masure, "Le design de la transparence"
[121] Jammi, Manasa, Jaydeep Sen, Ashish Mittal, Sagar Verma, Vardaan Pahuja, Rema Ananthanarayanan, Pranay Lohia, Hima Karanam, Diptikalyan Saha, and Karthik Sankaranarayanan. "Tooling framework for instantiating natural language querying system." in *Proceedings of the VLDB Endowment* 11, no. 12 (2018): 2014-2017.

domain specific terminology common to humanities datasets.[122] For the CFRP data, the word *représenter*, for example, is used to refer to performing a play. The verbs *jouer*, *donner*, or *répéter* can all be used interchangeably to mean the same thing, but only the first shares a WordNet synonym set with *représenter* and would therefore be identified using automatic methods. The configuration file creates a way for the user to circumvent this kind of issue and include precise terms, as well as variants like their nominal forms, as possible table match tokens.

The configuration file additionally allows field names to be mapped to database functions using specific arguments as a means of handling more complex concepts. This is notably useful for derived fields, which historically have caused problems for non-neural systems.[123] In the CFRP data, for example, money related data is stored as *livres*, *sols*, and *deniers*,[124] with each currency breakdown in a different column so as to match the level of granularity present in the registers. However, users expect revenue totals to be a single value. To account for this, the configuration file creates a pseudo-field called *recette* which maps to a function which calculates the value in *livres*. This notably does not preclude access to the individual currency fields. As with any other field, these pseudo-fields can also be assigned synonyms. This is particularly useful in the case of *recettes* and its user-defined synonym *rentable* (profitable), which is not an obvious synonym or word form derivative, but is commonly employed in reference to financial data in the sample question set. The primary limitation of this functionality is that it relies on predefined functions. Database functions cannot be created in the configuration file nor can pseudo-fields be mapped to arbitrary filters. Additionally, the current setup requires all function arguments to belong to the same table. I feel it is also important to recognise that, while it was not the original intention, this feature could be used to define a seemingly objective shorthand for subjective concepts or to obfuscate data gaps. For example, though a descriptor like *chef d'oeuvre* (masterpiece) is a primarily subjective categorisation, it would nonetheless be possible to create a simple boolean function that looks at a play's revenue and performance numbers, or even checks against a human-defined list, with this additional processing hidden from the user.

---

[122] Yaghmazadeh et al., "SQLizer"
[123] Saha et al., "ATHENA"; Kim et al., "Natural language to SQL"
[124] *Livres* were the common French currency from approximately 781 to 1795, after which point they were replaced by francs. Operating on a semi-duodecimal system, under the Ancien Régime (which covers the years in question) there were 20 *sols* (sometimes called *soldes*) to one *livre*, and 12 *deniers* to a *sol* (1 *livre* = 240 *deniers*)

The database schema is internally represented as a functionally undirected graph G(V,E). Each element in V is a table relation R_i which contains a non-empty set of attributes (columns) A^i with a primary key $R_iPK \in \geq 1(A^i)$ used to identify unique n-ples. All $v \in V$, $a \in A_j^i$ as well as all possible values of A_j^i will be referred to in future sections as *database entities*. E comprises primary-key-foreign-key relationships. These edges are directed but created in mirrored pairs, maintaining reflexivity to facilitate join path inference. Structural information is read directly from the metadata schema and potentially augmented using the configuration file, as outlined above.

Natural language input is initially internally represented as a dependency parse tree. Dependency relationships as well as pertinent linguistic information (parts of speech, morphological characteristics, lemmatized form, etc.) are extracted using spaCy.[125] To improve naive tokenization, special parsing rules can be introduced via the configuration file such that certain patterns that would typically be broken into several tokens are treated as a single unit. This allows parsing to be tailored to cases specific to a unique database while keeping the system generalizable. These rules are described using the format required by the spaCy matcher.[126] A user must essentially provide a list of dictionaries that together represents the pattern, each element corresponding to a token as identified by the standard breakdown rules that is to be combined with the others in the list. I further built in the option to include an extra validating lambda function[127] which is intended to take a list of the tokens matching the specified pattern and return a boolean indicating if the pattern instance should indeed be treated as a single token. If no validating function is given, it is assumed that all instances identified using exclusively the matcher rules are valid. In the case of the CFRP dataset, I added two token patterns—one to detect dates of the form YYYY-MM-DD, and another to match seasons, which are strings of the form ‘m-n’, where n = m+1 and both m and n are of the format YYYY. The former consisted of incorporating regex into the spacy matcher rules to weed out invalid dates. The latter addition used simpler rules that would match anything of the form ‘YYYY-YYYY’ but included a validation function to filter out ranges that didn’t qualify as seasons; the result is that something

[125] https://spacy.io/models/fr
[126] https://spacy.io/usage/rule-based-matching
[127] Using Python syntax, see appendix A

like ‘1756-1757’ is treated as a single token, while ‘1750-1790’ is treated as three distinct tokens - ‘1750’, ‘-’, and ‘1790’.

2.2.4.2 Matching

There are 78 different characters in the database named *Lisette*. There are seven different actors with the last name *Fleury*, four distinct plays entitled *Coriolan*, and two separate major authors who could both be referred to as *Corneille*. Entity mapping is a challenge.

I use ‘entity mapping’ throughout to refer to the process of linking natural language input tokens to database elements. It falls under the larger umbrella of token matching, which also includes identifying patterns of tokens that map to SQL structures and keywords. I am including in ‘database elements’ all tables, columns, rows, and individual row values. I opted for an iterative approach to entity matching. An initial round of processing identifies all possible candidate database element matches for each relevant input token, and the list is then narrowed down to a single match using a series of metrics. As previously seen, a variety of techniques have been employed across other systems to resolve ambiguities. PRECISE, for example, applies the max-flow algorithm to a graphical encoding of the different semantic interpretations, and ATHENA looks more holistically at candidate trees, producing a ranked list according to how well they adhere to a series of relationship constraints. SQLIZER similarly ranks candidate matches embedded in possible query structures according to a set of pre-defined linguistic and syntactic features, and NALIR asks for humans to intervene in ambiguous cases.

The matching technique used by CLAIRON most closely resembles the second and third approaches. Matching is done entirely without human intervention and uses a collection of metrics relating to linguistic features, entity relationships, and match prevalence. The initial identification of candidate matches relies on the text of the input tokens and their lemmatized versions. Tokens that, per their part-of-speech tag, could not reasonably map to database elements (e.g. conjunctions, participles, determiners) are ignored. The candidate search techniques are shaped by, and limited by, the fact that the final query is eventually exposed to the user. This means, for example, that searches for possible matches within textual fields are performed using SQL’s SIMILAR TO operation with basic regular expressions, rather than employing Levenshtein distance or other fuzzy match techniques that it would not be possible to

express in a query. I deliberately opted not to use the stemmed versions of tokens, as word forms are important to selecting the correct match in many cases in the sample data. For instance, tokens *comédiens* and *comédie*, where the former matches to a table and the latter to a column in a completely different table, have different lemmas but the same stem. The validity of this decision could be called into question with respect to different data contexts. Data types are also used to identify matches. This is especially important with respect to dates, where different sub divisions such as months—both numeric and textual representations—are taken into account. Following the initial round of matching, tokens which do not appear to contribute to meaning, being neither entities nor possible indicators of structure, are trimmed from the parse tree. Most determiners, for example, are discarded, as the gender and number information they encode is captured by the morphological information provided by spaCy's parser relative to the tokens that follow them. Auxiliary verbs are similarly dropped. Several rounds of disambiguation follow the initial matching, with the aim of weeding out unlikely candidate matches. The likelihood metrics rely in part on parse tree relationships as well as complexity—with tables being the least complex matches, and individual values the most complex. The working assumption is that, unless another indicator is given, the user is likely to be referring to the most 'obvious' match, which is oftentimes the least complex, as lower complexity (table) matches refer to less detailed and consequently more broadly known concepts than specific value matches.[128]

A first cluster of metrics focuses on close[129] parse tree relationships. These filter the candidate matches according to how likely they are given context. A first filter prioritises textual matches in short form over long form text, unless there is a direct parental reference. This means, for example, that for the token 'Bellecour', value matches within short form text columns like 'pseudonyme' in the actor table will be kept over mentions of 'Bellecour' in the long form text in the register annotations column. A second filter looks for cases of a child bringing specificity to a lower complexity parent. If a lower complexity parent has a single child and one or more of the child matches fits 'inside' the parent match—i.e. it is a column of a parent table match, or a value match inside a column match— and there is a determiner marking the desire for this degree

[128] In all descriptions and examples that follow, matches are written out in the same way as they are represented in the interface, with successive nested parts of the match separated by backward slashes. For example, a match corresponding to the literal 'tragédie' in the 'genre' column in the 'pièces' table is written as 'pièces/genre/tragédie'.
[129] Difference of depth < 2

of specificity, candidates are filtered to preserve only those matches. This contextual metric is good for correctly identifying matches that may not be the most 'obvious' were the emphasis to be placed only on complexity or volume. For example, the historical sources on which the play table was based assign the genre 'pièce' to a number of plays which do not follow the lines of traditional comedies, tragedies, dramas, etc. This means that the question *Quelles pièces ont le genre de pièce ?* is completely valid, and that the second 'pièce' token should map to the value match 'pièces/genre/pièce', as indicated by the 'le' token, rather than the table match 'pièce', which is the less complex match. The question *Quels comédiens jouent le rôle de Voltaire dans la pièce Corneille aux Champs-Élysées ?* is similarly complex as a correct match requires that the token 'Voltaire' map to a character in the metatheatrical play under consideration rather than the famous writer's author table entry. Because 'Voltaire' ends up as a child node of 'rôle'—which matches to the rôles table—this filter ensures that the correct match is selected. Another filter further narrows down match possibilities for high complexity matches by looking at foreign key match prevalence with respect to the most direct table match ancestor. For example, for the question *Quelle est la pièce la moins jouée de Racine ?*, there are, at this point in the process, two match options left for the token 'Racine'—'auteurs/nom/Racine' and 'rôles/nom/Racine'. The closest ancestor token with a table match is 'pièce', which matches to the 'pièces' table. Since paths[130] exist between the 'pièces' table and the 'rôles' and 'auteurs' tables, a check performs the relevant joins to see which potential match is associated with the greatest number of instances (rows). Since there are more plays written by Racine the author (12) than characters named 'Racine' (1), only the former match is kept.

Once the basic dependency constraints have been accounted for, all node match sets are reduced to retain only the lowest complexity matches. This means, for example, that in the context of the question *Quelles sont les deux pièces les plus rentables de Barbier ?*, 'pièces' will map to the play table 'pièces' rather than any number of plays which have 'pièce(s)' in their titles, since the table match is less complex than the value match. A final metric judges match likelihood according to global similarity to other column or value matches in the parse tree. Matches are considered similar if they share at least a table, if not a table and a column. For example, with the question *Qui sont les auteurs des cinq comédies les plus jouées qui ont été créées dans les*

[130] These are not required to be of length 1

années 1750-1790?, the tokens identified as dates (‘1750’ and ‘1790’) could have matched to date columns in the séances or débuts table, but the matches in the ‘création’ column of the play table were chosen because of the presence of a ‘pièces/création’ column match elsewhere in the tree.

After all of the tree structuring described in the following sections, a final round of disambiguation picks a single match to be used in query generation for any remaining nodes with multiple options. The metrics used at this stage are more broad—not well suited to initial decisions but necessary and sufficient for cases where few other clues are given. The first metric looks again at relationships to surrounding matches, but this time via any existing key relationships. It keeps the matches that are the most connected, which is to say whose tables have the highest degree in the schema graph. The second test looks at the distance from the tables in the candidate matches to other match tables in the parse tree, keeping only those with the shortest average distances. This metric performs poorly in cases where a number of junction tables are involved, as distance is perhaps not the most just measure of relevance. Since the first two metrics have no effect on match sets where all elements are in the same table, the final metric, which operates only on high complexity matches, simply defaults to frequency, keeping the matches with the most row instances. If the node match remains ambiguous after all checks, the default is to choose the first option in the set.

Incorrect matches are undoubtedly the biggest source of error in the interface. Unlike some systems which test out all possible matches and corresponding queries, I opted to select a single match prior to query generation for the purpose of being able to produce a clear and linear visualisation of the process for the user, and highlight the limits of computers with respect to knowledge inference. Though discarded matches are recorded, there is currently no mechanism for re-attempting the query generation process with a different match set. Integrating this sort of substitution and correction mechanism, to be undertaken automatically or potentially using correct matches selected by the user, would be an obvious next step for the interface.

2.2.4.3 Tree Transformation

Following entity matching, the parse tree is mapped to an analogous query tree. At the difference of the parse tree, nodes in the query tree loosely correspond to various SQL query components. NaLIR employs a similar tactic, dividing nodes into seven categories—Select nodes, Operator nodes, Function nodes, Name (attribute) nodes, Value nodes, Quantifier nodes, and Logic nodes. My breakdown differs slightly, grouping nodes into 8 types:

1) Entity nodes - These are used to represent tokens that were mapped to database elements (tables, columns, or values) during the matching step.
2) Literal nodes - These are used to represent numeric literals that did not map to specific entities.
3) Operator nodes - These nodes are used to represent both binary operators, such as inequalities, and unary operators like negation.
4) Function nodes - These are used to represent SQL functions, notably aggregations like SUM, COUNT, and AVG.
5) Order nodes - These nodes are used to mark a sort on one or more attributes.
6) Limit nodes - These nodes are added in cases where a token implies that only a limited number of results should be returned.
7) Group nodes - These nodes go hand in hand with aggregations, and designate the groups over which they operate by specifying the attributes used to define the sets.
8) Query nodes - This node type is different from the others in that it is the superclass catch-all used as default type for any token that does not map to any of the other types.

Upon initial construction, every node in the query tree is either an Entity node, if it has one or more potential matches, or a Query node. Following this first attempt at parse-to-query tree mapping, basic optimizations are run to refine the tree.

Unlike parse tree nodes, which are each linked to a single token, Query tree nodes can be tied to one or more input tokens, as entities or keywords may be inferred from collections of tokens (words). The first optimization looks to collapse entity nodes that are part of a multi-word expression into a single entity. This collapsing helps to overcome the challenge of matching to

multi-word expressions that are broken up, notably identified as a significant hurdle by Li & Jagadish,[131] whose eventual handling required all multi-word expressions to be delimited by quotation marks– a solution unsuited to my use case. Concretely, this means, for example, that for the question *Quelle est la saison la plus rentable de Pierre Corneille ?,* the tokens 'Pierre' and 'Corneille' will be collapsed into a single node because they share a match to 'auteurs/nom' and there is at least one shared row[132] between those matches. If more than one match is shared in such a node pair, the collapse preserves all overlapping matches. This functionality further extends to cases where a set of parent and child matches map to different columns in the same table, provided the cardinality of their row set intersection is exactly one. For example, in the context of the question *Quels rôles est-ce que Monsieur Bellecour joue le plus ?*, one of the matches for 'Monsieur' is the value match 'comédiens/titre/Monsieur' and one of the candidate matches for 'Bellecour' is the value match 'comédiens/pseudonyme/Bellecour'.The former's row set includes all actors who used the title 'Monsieur', while the latter's length two row set is comprised of the entries for Monsieur and Mademoiselle Bellecour. Their intersection comprises only one row—the actor table entry for Monsieur Jean-Claude Gilles Colson, dit Bellecour. In this case, as in all cases of node collapsing where there is a single item in the row overlap, the entity is re-mapped to the row's primary key. This means, for instance, that the 'Monsieur' token node matching 'comédiens/titre/Monsieur' and the 'Bellecour' token node matching 'comédiens/pseudonyme/Bellecour' in the above example are collapsed into a single node with a single match, 'comédiens/id/36'.[133]

The final query tree construction refinement is a group of parse course correction measures. French is by no means a significantly under-resourced language in the world of NLP, but spaCy's French models are nonetheless smaller and slightly less accurate in terms of dependency labelling[134]. In light of this, a few structural corrections are applied in cases where parsing is systematically incorrect. This, for example, includes recentering the tree or its subtrees to root

[131] Li & Jagadish, "Constructing an interactive natural language interface for relational databases"
[132] Part of the information stored in the internal representation of each possible match is a list of the row instances to which it corresponds. For example, the value match 'comédiens/pseudonyme/Molé' has a row set of length two, because the literal matches two different rows in the 'pseudonyme'—the entries for actor François-René Molé and his wife, Pierrette-Claudine-Hélène Molé.
[133] The 'id' column is the primary key of the 'comédiens' table.
[134] https://spacy.io/models/fr#fr_core_news_lg-accuracy

them at the appropriate nodes as indicated by the presence of demonstrative or relative pronouns like *dont* or *celui*.

Following preliminary optimizations and repairs, a series of query tree transformation steps add implicit nodes and alter structure based on keywords and relationships. It is worth noting that these structural transformations and the possible scope of the resulting queries are limited, as is the case for all grammar based systems, by the fact that they depend on a predefined list of SQL operation keywords and configurations. Though not beholden in the same way to an external keyword mapping, ML systems are similarly limited in scope by the structural variation of their training sets.[135]

Since one of my principal aims is to allow the end user to follow and understand the underlying logic, I opted to break the steps into logically distinct groups of operations associated with a limited set of linguistically similar keywords. I also see this clear cut set of steps as helping to illustrate to the user the determinism of computing. To that end, it was important to me that the same set of steps be shown for every query and therefore, to avoid the cognitive overload and potential confusion of frequently showing large number of detailed steps which did not alter the tree, some sets of parse corrections or less frequent (but still recurring) structural operations were grouped together under a more general heading. The high level tree transformation steps eventually made visible to the user are negation, aggregations, basic mathematical operations, ordering, logic operations (AND/OR), and structural corrections.

2.2.4.3.1 Negation

Negation is an essential search function, and French negation is theoretically easily recognizable, given the standard required grammatical formula *ne* (verb) *pas*. Further, the parser can tag, with reasonable accuracy, words that have negative polarity, which helps to identify alternative negative adverbs, such as *jamais*. What complicates the handling of negation in the context of an NLIDB is the linguistic register in which the user chooses to address the interface. In both oral and informal French, it is very common to drop the negative particle *ne* without this omission

[135] Kim. et al., "Natural language to SQL"

changing the meaning of the sentence. Conversely, certain more formal structures can include *ne* without it marking negation.

The question *Quelles sont les comédies que Monsieur Préville n'a jamais jouées ?* is an example of the standard case; both parts of the negation are present. The regular handling involves simply replacing the pair of negation markers with a 'not' operator node appended to their shared parent. The fact that both tokens play a part in negation is later made visible to the end user. Though *Quelles soirées rapportent pas plus de 1000 livres?* forgoes the *ne*, the negative adverb *pas* alone qualifies as an instance of negation. Here too an operator node is appended to the parent in place of the negation token query node. In later stages of processing, the added operator nodes are used to define or alter expressions or relationships. in the first example, for instance, where the negation node is a child of an entity node matching Monsieur Préville's numeric id, the presence of the negation node means that the *where* filter acting on the match's column checks for non equality with the associated literal instead of defaulting to equality. In the latter example, the negation node causes the inequality that maps to *plus de* to flip from from greater-than to less-than. *Quelles pièces ne sont jouées que deux fois?* is an example of a floating *ne* that does not not play into any form of negation; the question is functionally the same as *Quelles pièces sont jouées deux fois?*. In this case, no action is taken and the tree node for the *ne* token is simply removed.

2.2.4.3.2 Aggregations

At this stage, aggregations in the guise of function nodes are added to the tree in response to specific structures of keywords. Of the basic functions, COUNT, SUM, and AVG are all handled similarly. MAX and MIN are discussed in section 2.2.4.3.4. COUNT and SUM can both be invoked explicitly or inferred. Additionally, in certain cases where keywords might seem to call for COUNT to be used, SUM is more correctly employed for typing reasons. With the exception of the case where they are operating on all columns, the postfix-style structure sees aggregator nodes inserted as the parent of the nodes on which they are acting.

In the question *Combien de fois est-ce qu'on joue Molière pendant la saison 1765-1766 ?*, for example, *combien* and *fois* explicitly point to a need for COUNT. With a question such as *Quelle est la saison où l'acteur Duval joue le plus de rôles de tragédies ?*, however, the aggregation is

inferred. The token pair *le plus* implies that the results must be sorted according to *rôles* and since *rôles* is not a column nor is it the primary subject of the query, an aggregation is needed to make sort possible. COUNT is used by default, but in the case of numeric columns, the operation is changed to SUM. This mechanism, in the context of the question *Quel est le mois qui rapporte le moins en 1734?*, for example, means that results are correctly ordered by the total of each month's revenue rather than the number of performances that occurred in that month, which would have functionally been the result of applying COUNT to the numeric column (function) match of *rapporte*.

Unlike COUNT and SUM, AVG is never invoked implicitly. It is triggered by the keyword *moyenne*, which can appear before or after the tokens it is acting on. In the question *Quelles sont les comédies en 3 actes ou en 5 actes qui sont jouées moins de 15 fois et ont une recette moyenne de plus de 900 livres ?*, for example, the average is over the performance *recettes*, which precedes *moyenne,* whereas in the question *Quelle est la moyenne du nombre de spectateurs au parterre pendant la saison 1774-1775 ?*, the keyword comes first. In both cases, the resulting function node becomes the parent of the node or subtree on which it is acting.

2.2.4.3.3 Mathematical Operations

The two mathematical operations that naturally emerged from the NL structures of the sample questions are subtraction and division. These operations are functionally treated much the same as aggregations; their addition is triggered by specific keywords or sets of keywords, and they are inserted into the tree as the parent of the nodes they are acting on. A subset of possible inequalities are also handled at this stage.

Division is needed when an input question requires a ratio of some kind. The question *Dans quelles saisons est-ce que le nombre de spectateurs du parterre est particulièrement bas par rapport à la moyenne du nombre de spectateurs du parterre sur l'intégralité de la période 1680-1793 ?,* for example*,* essentially asks to compare the average number of tickets sold in a specific section for each season to the average number sold in that section over the entire period. Though in this specific context the comparison may be expressed differently by a human interlocutor, I chose to map *par rapport* to the division operation as a generalizable interpretation of structures. This question is also an example of the system interpreting and returning a result

reflective of the elements it was able to understand, while dropping others. The system couldn't process *particulièrement bas* and judged that these tokens were sufficiently isolated in the query tree that it was likely safe to discount their influence on the final query. Though the system will throw errors in cases where a lack of interpretable keywords leaves it unable to construct a query, in cases such as this, it is left up to the user to build an understanding of the elements of their question that are reflected in the results via the nl-token-to-query-token mapping that is exposed by the interface. Requests which imply ratios are the other case where division is used. With respect to the question *Quelle est la part des drames par rapport aux autres genres dans la programmation entre 1730 et 1770 ?*, for example, the keyword *part* maps to division such that the request returns the percentage of the repertoire occupied by plays classified as dramas that premiered between the specified dates. Though evidently never specified by the user, a type cast is automatically added to all denominators in order to avoid issues of integer division. As the values of query components are not at any point calculated separately, seemingly valid queries where the realities of data yield in a potentially revelatory 0-denominator are only caught—and the corresponding SQL error displayed in the interface—at execution time.

Subtraction nodes are integrated to the tree in the same way as division operators, their addition triggered by keywords *écart* and *différence*. This was the way I opted to render users asking for the difference between two values, allowing the result to then be used in further operations. In the context of the question, *Quels comédiens ont le plus grand écart entre leur entrée et leur sociétariat ?*, for example, the calculated value to be used for sorting.

Data ranges specified as two comparable tokens preceded by *entre* and separated by *et* as seen in the question *Quelles sont les pièces avec divertissement créées entre 1720 et 1750?* are rendered as a subtree with a logic node at the root having two children—in this case, the nodes corresponding to 1720 and 1750—each with their own operator node child specifying the inequalities defining the node's relationship to its parent. Soft inequalities are used for ranges.[136] Strict inequalities are used in the case of single sided date comparisons triggered by keywords *avant* or *après* used alongside an entity node with a value match on a column with datatype

[136] This decision was semi arbitrary. The choice to include endpoints was informed by previous work with the CFRP team where data requests using this kind of language were intended to be endpoint inclusive.

'date'. For example, the result list generated for the question *Quelles sont les tragédies créées après 1775?* Does not include any plays which premiered in 1775.

2.2.4.3.4 Ordering

Explicit cases of result ordering are inferred from token sets *le plus* or *le moins*. This process is complicated by the fact that, depending on surrounding structures, these keyword sets can also map to MAX/MIN functions or strict inequalities.

With respect to ordering, there are three different parse patterns that require identical rendering. *Qui sont les cinq auteurs féminins les plus joués entre 1730 et 1780 ?*, for example, is functionally equivalent to *Quels cinq auteurs féminins est-ce qu'on représente le plus entre 1730 et 1780 ?*, despite the keywords which invoke ordering appearing respectively before and after the token they are acting on (*joués*/*représente*). An additional formulation adds an extra partitive article *de*, as in the question *Quelle est la saison où l'acteur Duval joue le plus de rôles de tragédies ?*, which again produces a different parse. In all three examples, an order node is inserted as a child of the node it is acting on to indicate that node's relationship to its parent. All three of the above cases further require the addition of an inferred function node in order to make the ordering valid. Without an added aggregation, it is not possible, for instance, to order according to the table match *représentations* which corresponds to the token *joués*. It is, however, possible to sort on a count of the number of *représentations*. In the case of numeric columns, SUM is used, as is the case with the question *Quels cinq auteurs féminins rapportent le plus entre 1730 et 1780?*, where results are ordered according to aggregate revenue.

Whether the order is ascending or descending is understood from the use of *plus* or *moins*. In some cases, the addition of a size-related adjective requires the direction to be flipped. For example, in the question *Quelle saison connaît la recette totale la plus basse ?* the addition of *basse* means that the descending order assumed from *plus* is switched to ascending. That said, the system recognises that not all opposing keyword/adjective pairs require flipping. The adjective *importante* in the context of the question *Quelle saison connaît la recette totale la moins importante ?*, for example, only emphasises rather than alters the direction of the sort.

When a size adjective is present relative to a deeply nested entity node, a function node is added rather than an order node. For example, in the context of the question *Quelle est la saison qui connaisse le plus grand écart entre la recette la plus haute et la recette la plus basse?* the token sets *la plus haute* and *la plus basse* are rendered respectively as the maximum and minimum receipt values.

Finally, when coupled with a subordinating conjunction, keywords that usually denote ordering instead indicate a comparison, which is added to the tree as a strict inequality. Here, too, size adjectives are taken into account when determining direction such that in the case of the query *Lors de quelles saisons est-ce que la recette de Molière est moins que la recette de Voltaire ?*, replacing *moins* with *plus basse* correctly generates an identical result. Comparisons between entity or function nodes and literals are also considered valid. In the context of the question *Lors de quelles soirées est-ce qu'on a joué Athalie mais rapporté moins de 1000 livres?*, for instance, the literal is compared to the numeric column match of *rapporté,* and in the case of *Quels sociétaires jouent moins de quinze fois pendant la saison 1774-1775 ?* the comparison is between the literal *quinze* and a count of performances. As with structures signaling order, implied aggregations are added as needed to form valid queries.

2.2.4.3.5 Logic Operations

At this stage, tokens explicitly denoting logical relationships such as *et*, *ou*, *mais*, and *ni* are either rendered as their boolean operator equivalents or eliminated from the tree if they are found to be superfluous. In the context of the question *Quelles sont les pièces les plus jouées pour les ouvertures ou clôtures de saison ?*, for example, the token *ou* is rendered as a logical OR, joining the nodes specifying the conditions of it being the opening of a season and the closing of a season respectively. The system notably does not correct errors in the user's logic. For example, the input *Quelles sont les comédies en 5 actes et en 3 actes qui sont jouées moins de 15 fois ?* returns zero rows, as it is impossible for a play to have both three and five acts. A lack of stored information around relationship cardinality means that detecting instances of this type of error is non-trivial in the current version of the tool, but a future version might look to explain them to users unfamiliar with boolean operations. The equivalent question which swaps *et* for *ou* correctly returns the list of three or five act plays performed fewer than 15 times.

When *ni* is the child of two related comparable nodes—as is the case for the input *Quels sont les rôles féminins dans les pièces de Voltaire joués par ni Clairon ni Dumesnil ?*— a negation node is added as a child of each operand and the nodes are joined with an AND. I opted to maintain this $\neg a \wedge \neg b$ formulation of conditions as opposed to swapping to the slightly more compact equivalent structure $\neg(a \vee b)$, as the former bears more resemblance to the input query and therefore seems likely to make the mapping easier for a user to follow.

Keywords mapping to AND nodes are discarded when they describe the relationship of a direct child to the root, as logical conjunction in the case of filters, and co-selection in other cases, is the assumed relationship of children to the root if not otherwise specified (by an order node, for example). In the question *Quels sont les titres et les genres des pièces de Diderot ?*, for instance, the coordinating conjunction *et* is discarded, as it is not needed to specify the relationship of *genre* to the root. OR nodes are discarded when they represent a tautology. If two identical nodes with no subtrees are joined by a node implying an OR, the query nodes mapped to the logic keyword and to the duplicate operand node are both dropped. One place where this occurs in French is when a user is attempting to write somewhat inclusively rather than defaulting to male for a mixed group of individuals. In the context of the question *Quel acteur ou actrice joue le plus les pièces de Beaumarchais ?*, for example, both *acteur* and *actrice* match to the *comédiens* table, so a subtree composed of both entity nodes joined by an OR operator is functionally equivalent to a single entity node. It should be noted that the system is not currently equipped to handle the more compact style of inclusive writing that utilises interpuncts (e.g. comédien·ne·s).

2.2.4.3.6 Structural Corrections

This final step includes a number of minor keyword-based operations, as well as a collection of tree restructuring measures that add, remove, or relocate nodes to ensure structural validity. In the current version of the interface they are presented to the user as a single—though clearly composite—step. A future instantiation could opt to separate out discrete finer-grained steps, where possible, if the more a nuanced breakdown would be clearer for the user, despite the added complexity of each individual transformation not being as broadly applicable or linguistically obvious as those in previous steps.

Miscellaneous Keywords: autre and différent

Autre and *différent* both map to unique operations—not equal, in the sense of exclusion from a group, and *distinct* respectively—that needed to be applied after the core tree structure was established because they are fundamentally referential.

Consider as an illustration of the former the question *Combien joue-t-on Molière par rapport aux autres auteurs dans la programmation ?*, which asks to compare the number of performances of Molière's plays with those of all plays not penned by the infamous playwright. In this case, *autre* ends up as a child of table match *auteur*; in order to create the comparison, a comparable value-match entity node must be located. If we understand a match as comprising up to three components—a table, a column, and a value, $m = \{m_t, m_c, m_v\}$ —a higher complexity entity node $e_0$is considered comparable to a lower complexity one e_1 if there exists a match pair m_{0i}, m_{1j} such that $m_{1j} \subseteq m_{0i}$. Once such a node is located, the exclusion is expressed by substituting the nodes corresponding to the keyword *autre* and its parent with a clone of the higher complexity node modified by a negation node child.

The specification that only distinct values should be selected is either understood through the use of forms of the keywords *différent* or *distinct,* or implied through structure. The question *Quel est le rôle qui a été joué par le plus de comédiens ou comédiennes différents ?* illustrates the first case where, without the addition of distinct on the entity node matching the actor table that is used in the aggregation by which the results are ordered, the returned row would simply correspond to the role performed the most, rather than to the role performed by the largest number of different actors. Cases where distinct needs to be applied to the root node are inferred by checking to see if the total number of non null instances of the root value returned by an open query on the join path is greater than the number instances in the table corresponding to the root. For instance, with the input *Quelles comédiennes ont incarné Oenone?,* the root matches to the *comédiens* table, but that table must be joined with the *interprétations* table in order to connect actors to their performance of roles. Because an actor can perform many roles many times, there are more non null instances of actors in the joined table than there are rows in the actor table. This condition implies the need to enforce a distinct selection on the root.

Adding Grouping

The addition of group nodes can be triggered by the presence of aggregations in the tree and through the use of certain keywords, as well as inferred from specific structures. The question *Quelles sont les 10 saisons les moins rentables ?* illustrates the basic case. The ordering on column match root node (*séances/saison)* by an entity node that does not match the root (*registres_recettes/recette*) implies the need to add a grouping on the root node, which in turn requires adding aggregation on the node being used for ordering. The input *Comment se répartissent les genres saison par saison ?* invokes two different keywords that trigger the addition of group nodes. *Répartition* implies results that give a breakdown, requiring the addition of three things—first, a group node; second, an entity node inserted as a child of the root such that, if it is not already the case, the subject of the breakdown is included in selection; and finally, a row count to account for the fact that the tree lacks an existing aggregation to use in the breakdown. The keyword *par*, in addition to *chaque*, when directly followed in the input sentence by a token mapping to an entity node and lacking a leading determiner, also implies a grouping. The query resulting from the above input question is therefore grouped by both *séances/saison* and *pièces/genre*.

Adding Limits

Limit nodes are added to the tree in two distinct scenarios. The first is as a means of processing 'floating' literal nodes, which is to say literals that are not involved in any sort of operation. In the case of the question *Quelles trois pièces est-ce qu'on joue le plus entre 1730 et 1740?*, for instance, the token *trois* is unrelated to any comparisons or aggregations, and is not the child of a numeric column-matched entity node to which it could be compared. It is therefore assumed to be a limit. Limit nodes can also be added as a function of root token morphology. If the token happens to be singular, a limit of one is added. This means, for example, that the input *Quelle comédienne joue le rôle d'Agrippine le plus souvent ?* returns a single row, whereas the input *Quelles comédiennes jouent le rôle d'Agrippine le plus souvent ?* is not subjected to any limit, since *comédiennes* is plural.

The Feminine Filter

In addition to potentially adding a limit, I have built in a mechanism whereby root token morphology can also introduce a filter. If the root token is feminine and the root node is matched

to a table with a column named *féminin*, a filter is added to reflect this additional level of precision. This means that the input *Quelles comédiennes jouent le plus ?* returns only women actors, while *Quelles comédiens jouent le plus ?* follows the rule of the inclusive masculine and returns results relating to all actors, men and women alike.

This additional bit of processing is the operation that could be considered the closest to overfitting. However, its more global potential has not yet been explored, as NLIDB systems and literature have so far been limited to English, which is not a gendered language. The overall utility of this feature, including its flaws and the ways in which it might be specified with greater flexibility via the configuration file, would be relevant explorations for future expansions of the system.

2.2.4.3.7 Tree Structure Validity Verification and Adjustments

Before moving on to the query generation stage, the tree is subjected to a series of different checks to address evident structural issues. One frequently occurring correction is splitting a value-match root node into its component parts. The root node is, in the majority of cases, what is selected in the SQL query corresponding to the tree. In SQL, a search for a given row-match manifests as a lower complexity select with a WHERE filter. Therefore, to create a tree that matches the required format, a value-match root node is split into a table-match root and a row-match child that is rendered as an equality filter during query generation. Other small verifications include ensuring that all order nodes are linked to a parent of an appropriate type—not a value-match node, for example—and that non boolean column-match entity nodes are involved in some sort of operation or either removed or signaled as an error, depending on their position in the tree.

To prepare the tree for query generation, nodes whose meaning is unknown or doesn't contribute to query formulation are removed from the tree. This most commonly applies to nodes which I have come to think of as 'floating tables'—table-match entity nodes whose inclusion in the list of tables through which the join path must pass is often essential to properly representing the question, but that are not invoked elsewhere in the query. Individual or self-contained subtrees of nodes whose tokens are considered non-trivial by virtue of their part of speech tag but do not

belong to any of the keyword sets (interrogatives, function words, etc.) and do not map to any entities are also removed. In both cases, the removed nodes are retained to potentially be used in later stages to either help generate join paths or inform the user that the system has failed to understand a central part of their question. Consider the basic example *Combien de fois est-ce qu'on joue Beaumarchais sur l'ensemble de la période ?*. Here, the token *joue* maps to the table *représentations*, but neither that specific token nor the table in general is part of the selection or any of the filters. This means that its node must be removed from the tree so as not to be rendered as part of the query's SELECT clauses or conditions. However, it is essential that it still be included in the join path, because the default shortest path linking *auteurs* and *pièces*—the table components of the entity matches in the rest of the tree—does not include *représentations*. In light of this, without requiring the removed 'floating' table to still be considered when generating the join path, the input would incorrectly simply return the total number of plays written by Beaumarchais, as opposed to the total number of performances of his works.

None of the tokens in the phrase *sur l'ensemble de la période* map to database entities or have any impact on the results.The dependency parse reflects the way in which they are largely disconnected from the other components of the query, allowing the phrase subtree to be removed. However, swapping *Beaumarchais* for *Shakespeare,* for example, causes an error to be thrown, as the node the system cannot understand (*Shakespeare*) is integral to the tree structure. A naive user might think that instead of an error being thrown, this query should simply return no results, as there are no plays credited to Shakespeare in the database; therein lies the value of throwing it. The system has no way of knowing Shakespeare is an author. Even the tree structure could not be used to infer this with certainty, as *joue* can refer to the performance of a play or to the performance of an actor in a role, meaning that an actor match would be an equally valid interpretation. Explicitly raising this kind of error asks that the user reckon with prerequisite knowledge they may unconsciously be assuming of the computer. This is particularly germain with respect to terms with vague or variable definitions. In the case of the input *Lors de quelles séances a-t-on joué une tragédie du répertoire primitif ?*, for instance, the token *répertoire* (if only because it comes before *primitif*, which would do the same) causes an error to be thrown. This is not an indication of the impossibility, within the confines of the interface, to achieve the desired result; a user must simply think through what they mean by *répertoire primitif*; the input

Lors de quelles séances a-t-on joué des tragédies crées avant 1680 ?, would notably return the correct result, as the répertoire primitif is composed of plays that pre-date the foundation of the Comédie-Française.

2.2.4.4 Query Generation

This final stage of question to query translation is primarily based on a postorder traversal, as entailed by the structural similarity of the query tree to a postfix expression tree. However, following tree transformation, three additional steps occur before starting on query generation. The first is a final definitive disambiguation. The traversal assumes that there is only a single match associated with each entity node. Though all potential matches are preserved and exposed to the user in the UI (see section 2.2.5.2), at this stage—as described in section 2.2.4.2—the list is narrowed down to one using basic match similarity and frequency metrics. The second is join-path inference. Once the matches are finalised, it is possible to extract a fixed set of tables that are required by the query. Since there tables represent a subset of the vertices in the database graph, join path inference can be modeled as a minimum Steiner tree problem,[137] which consists of finding the shortest path between a set of nodes S for a graph $G = (V, E)$ where $S \subseteq V$. It is worth noting that the CLAIRON NL to SQL translation process currently only deals in inner joins. The detection of phrases which imply the need for other join types could be an interesting area for future explorations. The third and final preparatory step is foreign key substitution. Though not necessary to produce functional queries, I opted to include this step in order to optimise queries by reducing the number of joins. For each row-match entity node in the tree, if it is referencing a single row and its table is linked by a foreign key to another table already in the path, the match is swapped out for a foreign-key match. Provided it is not being used by any other entity node matches, the table from the original match, now superfluous, is removed from the join path. In addition to producing shorter and, by extension, potentially more readable queries, this serves to highlight important aspects of computational information modeling logics. The intention is to encourage the user to move away from the common monolithic spreadsheet notional machine of a database toward a more accurate networked mental model that affords the construction of more complex queries.

[137] Saha et al., "ATHENA"; Baik et al. "Bridging the semantic gap with SQL query logs in natural language interfaces to databases"

On an abstract level, the final query is composed of a series of nested strings each corresponding to a node, which bubble up the tree until reaching the children of the root. The strings matching the root and each of its subtrees are then combined with a SELECT statement and the pre-calculated join path, in an order reliant on the nature of each child and its relationship to the root. Consider the example input question *Quelles sont les trois pièces de Voltaire les plus jouées entre 1760 et 1775 ?*. Figure 2.2.4.4-1 shows the query tree after the completion of all preprocessing, including foreign key substitution.[138]

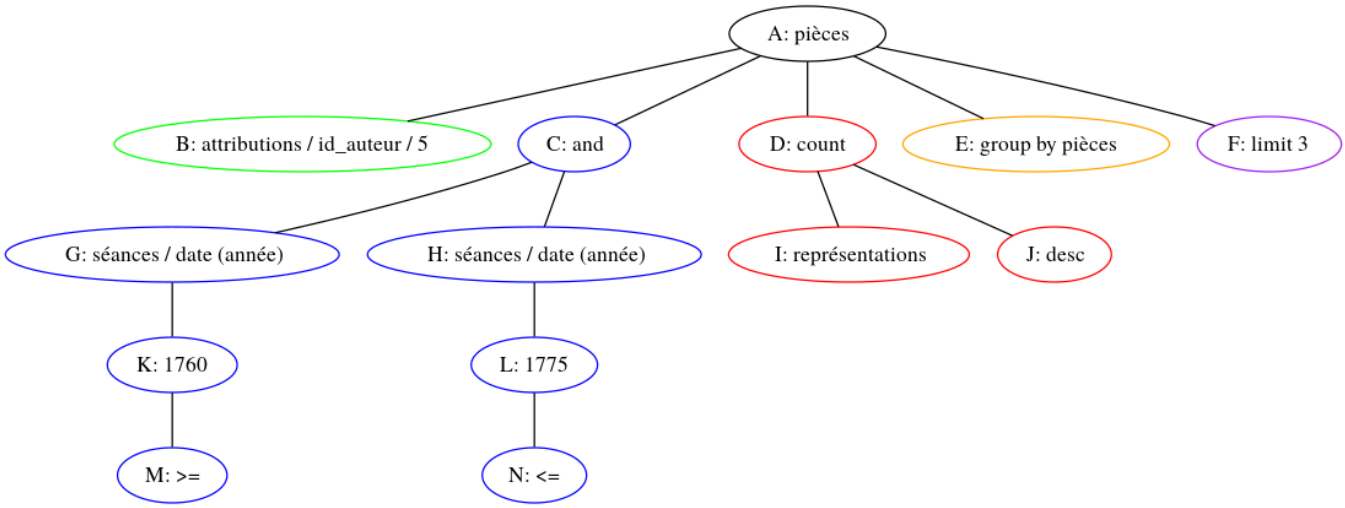

Figure 2.2.4.4-1: Final parse tree for input *Quelles sont les trois pièces de Voltaire les plus jouées entre 1760 et 1775 ?*.

The nodes in the tree are visited in postorder sequence (B, M, K, G, N, L, H, C, I, J, D, E, F, A) and the string corresponding to each subtree is recursively integrated with text of its parent. The string for the leftmost subtree (green) is simply the string for entity node B. The string representation for value-match entity nodes like B is composed of three parts—the database entity specification, the relationship operator, and the literal. Following this pattern, B is rendered as *attributions.id_auteur = 5*. For textual columns, ILIKE is used. If the node additionally has a negation node child, the comparison operator is modified accordingly. Table and column-match entity node strings comprise only the entity specification, with the wildcard used in place of a column for the former. Though the precise *table.column* specification of

[138] The letters have been added for clarity and have no significance to the tree.

database elements is not required by SQL in cases where there is no ambiguity, I have chosen to consistently include it for clarity. The string generation process for the two rightmost subtrees (orange and purple) is a similarly simple combination of SQL keywords and literals or entity node text.The subtree rooted at C (blue) is the best illustration of the recursively defined nature of the subtree strings. The string for node K is simply the literal *1760* with operator node M understood to be K's relationship to its parent. The node G is a special case of an entity node, where an extra SQL operation is added to account for the fact that G is matched to only a portion of a date; it is rendered as *extract(year from séances.date)*. This means that the string corresponding to the entire subtree GKM is *extract(year from séances.date) >= 1760.* This structure is mirrored by the HLN subtree and it follows that the string for the entire blue subtree is *(extract(year from séances.date)>= 1760 and extract(year from séances.date)<= 1775).* In the case of the subtree rooted at D (red), it is order node J that specifies its relationship to its parent. Aiming again for clarity, I have opted to explicitly include the direction keyword in both ascending and descending sorts, despite it only being required in the latter case.

The ways in which subtree query segments are combined depends on the nature of the parent, and whether or not any of the type of relationship is specified somewhere in the subtree. At every level, logical conjunction is the assumed relationship to the parent where no other one is given. This is the case for the green and blue subtrees. The exception to this rule is function nodes which act as filters, as opposed to being selected; these must be introduced by the keyword HAVING. All other subtrees in the example question either have relationship markers or do not require them. The final query produced for the above tree following the addition of the join path that must necessarily pass through attributions, séances, représentations, and pièces is pictured in figure 2.2.4.4-2.

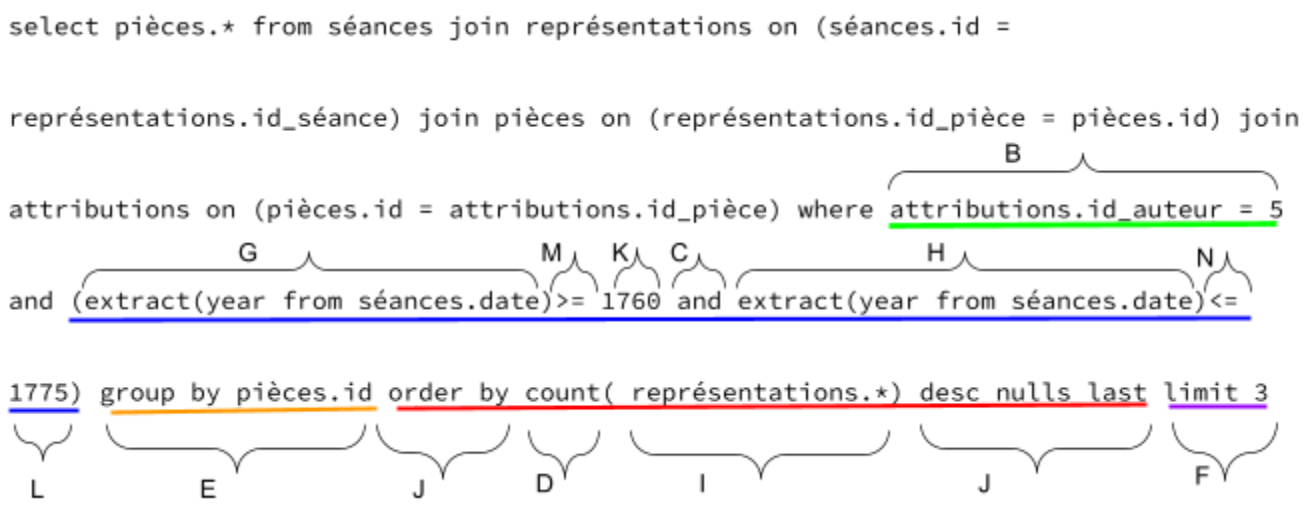

Figure 2.2.4.4-2. Corresponding SQL query for NL input *Quelles sont les trois pièces de Voltaire les plus jouées entre 1760 et 1775 ?*.

The order in which the fragments are encountered in the input and correspondingly, the parse tree, may not always be reflective of the order in which they appear in the resulting query. No matter the original indices of their associated tokens, certain node types — notably groupings, limits, and HAVING clauses — must appear at particular positions in the query. As is the case with natural language translation, there are many ways in which a query representative of any given input can be rendered. The queries generated by the system are not divorced from human intervention but are rather a product of my choices, and I would be remiss not to mention a few of the most significant.

2.2.4.4.1 Non Selection of Calculated Fields Used for Ordering

In cases such as the above where the query is ordered by a calculated field, I have not enforced the selection of this field, though it may oftentimes be of interest to the user. The system currently does not support aliasing, which is common to query realisations where a calculated field is both selected and used to sort the results. That said, in light of the fact that they NL-to-SQL mappings are exposed to the user (see section 2.2.5.2), the idea is that, were they to want to select the calculated field, they would be able to deduce how the query might be altered to do so and test their modified query using the SQL input side of the interface.

2.2.4.4.2 Query Flattening

For the sake of brevity and efficiency, I chose to use CASE statements instead of subqueries wherever possible. This is most commonly seen in cases where mathematical operations or aggregations are acting on multi-node subtrees, or a single value-match or boolean column-match entity node. Consider, for instance, the input *Lors de quelles saisons est-ce que Monsieur Bellemont a joué moins de 100 fois?* whose corresponding tree is pictured in figure 2.2.4.4.2-1.

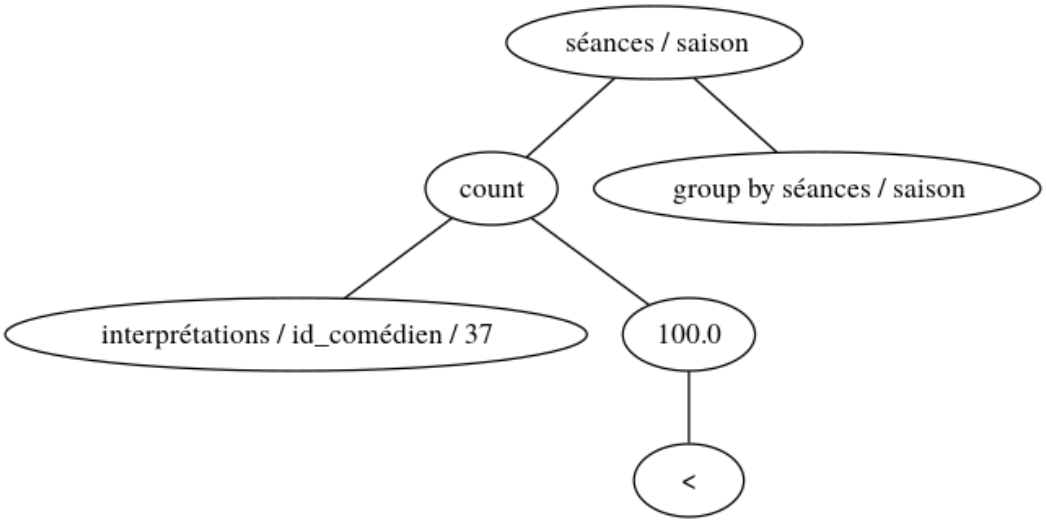

Figure 2.2.4.4.2-1. Query tree for the input *Lors de quelles saisons est-ce que Monsieur Bellemont a joué moins de 100 fois?*.

Here, instead of using a subquery with a WHERE filter to account for the value-match on which the aggregation is acting, CASE is instead used to perform what amounts to an in-place column-wise transformation to produce the following query:

```
select séances.saison from séances join représentations on (séances.id
= représentations.id_séance) join interprétations on
(représentations.id = interprétations.id_représentation) group by
séances.saison having sum(case when interprétations.id_comédien = 37
then 1 else 0 end)< 100.0
```

It should be noted that parse tree corrective measures sometimes fail to identify instances where the structure implies a case statement that is not wholly necessary. In these cases, it does not produce a completely incorrect result, but input translations could have been rendered using a simple filter. Take for instance the question *Quelles sont les pièces les plus jouées de Boursault entre 1774 et 1778 ?* which produces the tree pictured in figure 2.2.4.4.2-2.

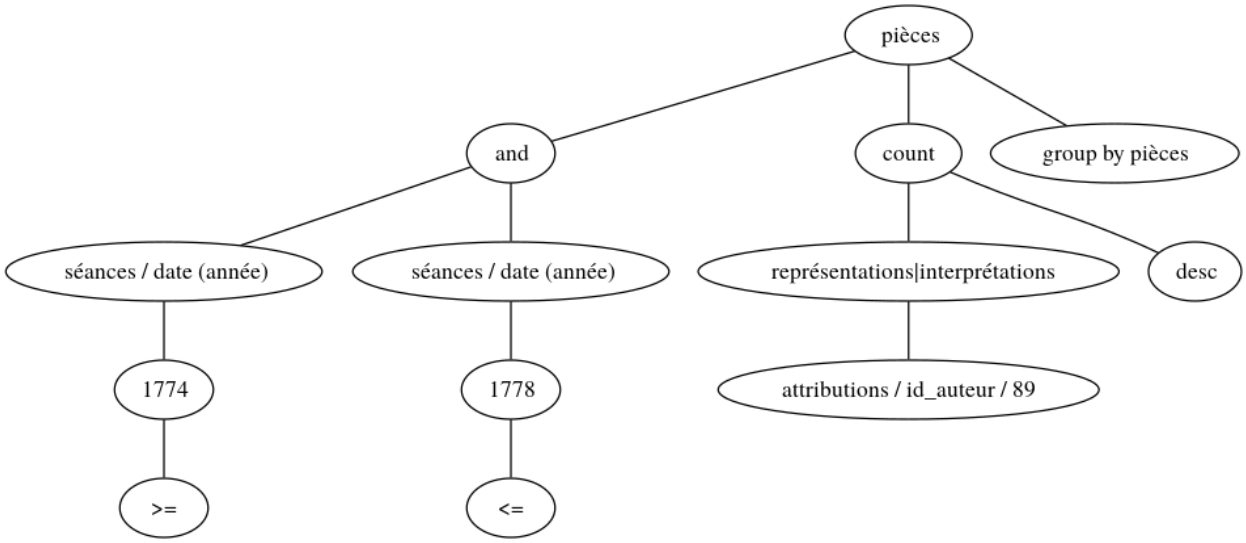

Figure 2.2.4.4.2-2: Query tree for input *Quelles sont les pièces les plus jouées de Boursault entre 1774 et 1778 ?*.

The author filter (attributions / id_auteur / 89) could easily be moved up so that it is a child of the root node. As it stands, the resulting query (below) does return Boursault's plays in descending order of the number of times they were performed during the given window, but, since the filter was treated as part of the ordering, after those rows is also listed every play performed between 1774 and 1778 in no particular order, because all of their order values are functionally 0. Though this formulation still mostly yields the correct result, it is more easily misinterpreted by a user who has limited knowledge of query languages.

```
select pièces.* from attributions join pièces on
(attributions.id_pièce = pièces.id) join représentations on (pièces.id
= représentations.id_pièce) join séances on (représentations.id_séance
= séances.id) where (extract(year from séances.date)>= 1774 and
```

```
extract(year from séances.date)<= 1778) group by pièces.id order by
sum(case when attributions.id_auteur = 89 then 1 else NULL end) desc
nulls last
```

2.2.4.4.3 Mathematical Operation Assumptions

Though not explicitly signaled by the user in any way — I am unsure of how it would manifest — I chose to force floating point division by adding a default cast to the denominator. I additionally wrapped every subtraction in an absolute value. The utility and validity of this is perhaps more debatable than the change to division, but I see it as logical in relation to the keywords that map to subtraction. In the context of the sample questions in which they occured, tokens such as *écart* or *différence* would seem to imply a scalar quantity. Should the keywords or their interpretation by a significant fraction of users change, this addition may need to be reconsidered.

2.2.4.5 Basic Error Handling

The currently implemented error handling is fairly rudimentary. In a handful of cases, such as the lack of mappings evoked in the previous section, errors arising over the course of the natural language to SQL query translation will halt the process before arriving at query generation. This most often occurs when the system encounters outside of vocabulary jargon that it cannot interpret, but which is central to the question. The other major category of this type of error is those relating to structural issues, such as being unable to locate a comparable node for binary operations, or trying to apply SUM or AVERAGE to a table-matched entity node. Nothing in the handling of these errors, which simply consists of returning basic pre-defined error messages, draws a delineation between those introduced by the parser and by the user. Since the user only has control over the latter, working to distinguish user-born errors from parser-born ones would be a valuable future addition with respect to improving user experience. Since errors that come up during the tree transformation halt the process, they do not return any results. To avoid confusion, a different warning is posted for valid queries that do not return any rows. For example, the query generated by the input *Quels comédiens ont joué 1765-02-13 ?* is completely correct, but there is no data to select as the earliest casting register starts in September of 1765.

Some errors are not obvious until query generation, where the tree traversal will fail to produce valid SQL. In these cases, rather than returning an error stating that the query is invalid, the parser's incorrect query is returned and displayed alongside the SQL error message (see section 2.2.5.1). This both gives the user the chance to see what precisely went wrong, as well as to potentially repair the query. It is worth noting that the SQL error messages can sometimes be difficult to parse for novice learners. A valuable future direction for future work might include the implementation of enhanced error messages, which simplify and explain common error patterns and have been shown to help students better understand compiler or interpreter feedback.[139] Raw errors are likewise returned when a user submits an invalid query to the SQL input side of the interface.

2.2.5 CLAIRON User Interface

The UI for CLAIRON is inspired by what Hmelo & Guzdial call 'glassbox scaffolding'.[140] Educational scaffolding looks to enhance learning by providing support to students for working through complex problems rather than exclusively using simplified examples. 'Glassbox' scaffolding—as opposed to 'blackbox'— aims to support both performance and learning, not exclusively the latter, by exposing to the user exactly how extra provided support functions. For instance, in a programming environment that allows users to lean on a library of components, a glassbox system would let them 'see inside' and manipulate the code comprising the pre-built elements. This has proven particularly beneficial in a computing context when the focus is on thought process. In her recent work exploring 'purpose-first' programming, Kathryn Cunningham used glassbox high-level 'program plans' to create a style of instruction that focused on the sort of abstract instruction (program planning, subgoal labelling) that most benefited her target novice learner group without removing the possibility of working at a more granular level.[141] The crux of the case for glassbox scaffolding is that by "seeing implicit processes made explicit, and being encouraged to reflect on those processes the learner is able to

[139] Becker, Brett A. "An effective approach to enhancing compiler error messages." in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (2016): 126-131.
[140] Hmelo, Cindy E., and Mark Guzdial. "Of black and glass boxes: scaffolding for doing and learning." in *Proceedings of the 1996 international conference on Learning sciences* (1996): 128-134.
[141] Cunningham, Kathryn, Barbara J. Ericson, Rahul Agrawal Bejarano, and Mark Guzdial. "Avoiding the Turing Tarpit: Learning Conversational Programming by Starting from Code's Purpose." in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021): 1-15.

construct new knowledge structures and modify existing structures."[142] It is this notion of process-focused reflection that the CLAIRON interface is designed to emphasise. The CLAIRON UI might be best characterised as 'translucent box'. It does not give full access to the code underlying the NL-SQL translation, but that code is not the focus; SQL queries are. By showing the user the steps—and missteps—involved in mapping human logics to computational ones, they are encouraged to critically examine the process and its limitations. They are not required to try their hand at this process independently, but the interface nonetheless offers them the possibility of trying raw SQL query input and offers ample support.

There are four sections to the interface—the question input and result display, the entity mapping display, the sequential parse tree viewer, and the table explorer. Each section aims to support the user in making sense of the SQL mapping and query results as well as come to grips with the differences between computational processing and human cognition. In exposing each step of the process, the interface makes explicit what is typically buried under layers of abstraction. Section-wise documentation helps guide the user through the interface, with external links to provide additional context to the data and further information to support SQL learning without weighing down the UI.

2.2.5.1 Question Input and Results

The design of the question input section with its two side by side input boxes deliberately draws on the visual vocabulary of automatic language translation systems such as DeepL[143] and Google Translate.[144] The box on the left accepts natural language queries, and the one on the right, arbitrary SQL queries.[145] The SQL input box supports syntax highlighting and includes line numbers, but is not a fully fledged editor (e.g. no autocomplete). When a natural language input is successfully translated, the input question and the SQL translation are both displayed above the input boxes with colour-coded highlights to show the correspondence between the two (figure 2.2.5.1-1). Additionally, all tokens that were part of the final query tree are displayed in bold. This allows users to identify which tokens were deemed irrelevant by the system, quickly

[142] Hmelo & Guzdial, "Of black and glass boxes"
[143] https://www.deepl.com/en/translator
[144] https://translate.google.ca/
[145] Though not explicitly advertised, any queries submitted via the interface are executed by a read-only user, so that no changes can be made to the database

catching onto any errors born of dropped terms. The current version doesn't consistently add highlights for operators, though they are bolded, nor does it indicate which of the tables in the join path were explicitly mentioned in the input question and which were inferred. These oversights should be corrected in a future version.

Results of successful inputs are funnelled into a grid located directly below the input boxes. Derived fields currently have non-informative column names; inferring useful labels from query structure would be a nice future addition on the interface. If an NL input fails, the error returned by the back end system is displayed in a temporary banner along the top of the page. Interpreter errors are displayed in cases involving malformed SQL.

Figure 2.2.5.1-1. NL input and query translation with highlight mapping for the input *Quelles sont les comédies en trois acres ou en cinq actes qui sont jouées moins de 15 fois et ont une recette moyenne de plus de 900 livres ?*.

Successful SQL inputs simply fetch results. Additionally generating a natural language approximation for input queries, as NaLIR does, could be an interesting area for future work. I made the deliberate choice not to support SQL-NL back-translation in this first version of the interface, as I believe many users' first instinct would be to attempt to retranslate the

generated—and therefore likely fairly unnatural— language into SQL. Though interesting from an iterative problem solving perspective, this functionality would demand that the system support a much broader range of keywords and structures.

All successful and unsuccessful NL or SQL inputs and their corresponding translations or errors are saved to a persistent log (figure 2.2.5.1-2). The purpose of the log is twofold. First, if a user is interested in modifying a previously submitted or translated question or query—erroneous or not— they can simply click the pen icon to automatically repopulate the relevant input box (though, not the results, in the current version). This, I believe, stands to facilitate the sort of iterative problem solving that is essential to research. Second, the backlog of examples is a place where SQL learners can study previous NL-SQL translations and abstract upwards to gain intuition around which types of fragments serve specific purposes. In the NaLIR extension TEMPLAR query translation logs were leveraged to help correctly predict output[146]; there is no reason that this high-level pattern matching could not equally prove useful at a more human level.

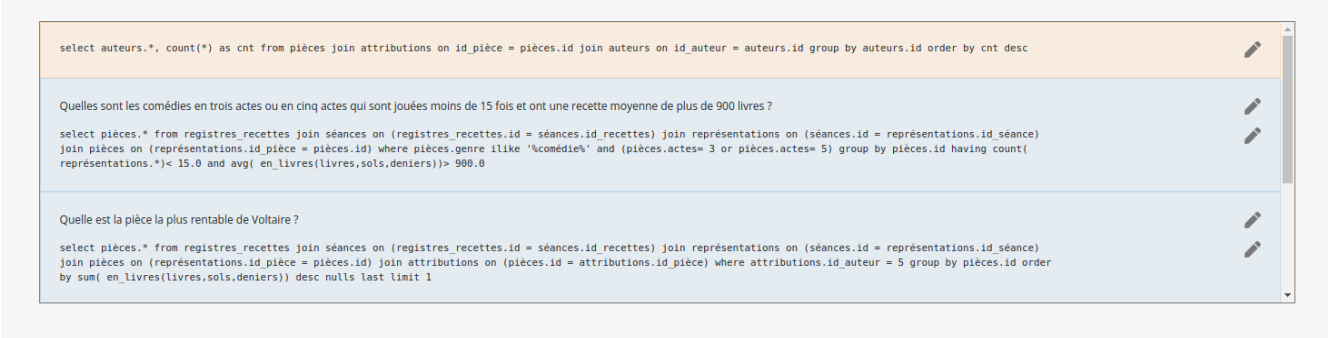

Figure 2.2.5.1-2. CLAIRON query logs

2.2.5.2 Entity Match Display

This section details the disambiguation process for all input tokens that match to database entities. To showcase the difficulty posed by language ambiguity, all possible matches identified during the first round of processing are listed. Using parallel lists positioned from left to right in an attempt to convey the iterative nature of the process, the selected matches and, where

[146] Baik et al., "Bridging the semantic gap with SQL query logs in natural language interfaces to databases"

applicable, foreign key substitutions, are displayed. Figure 2.2.5.2-1 shows the mapping for the input *Quelle est la pièce la plus rentable de Voltaire ?*.

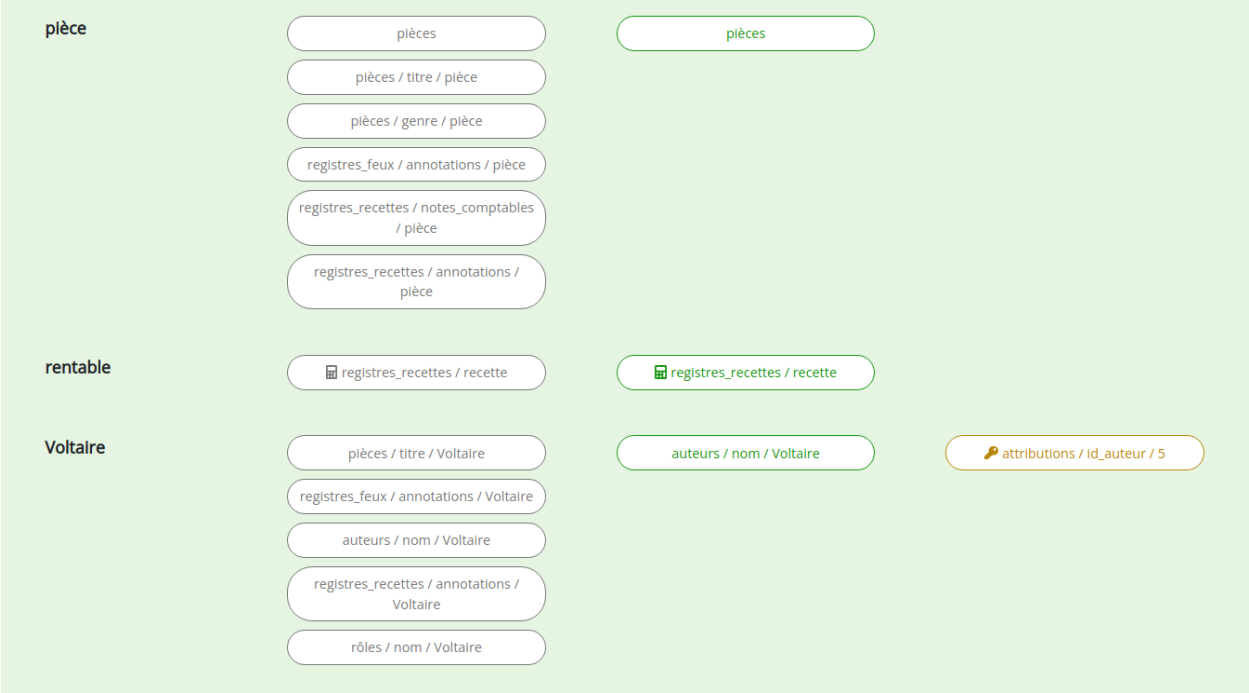

Figure 2.2.5.2-1. Node match lists and selections for the input *Quelle est la pièce la plus rentable de Voltaire ?*.

Specific icons flag fields that are keys or function fields. Currently there is no indicator for when a synonym is introduced via the configuration file is used. For example, the link between the token *Voltaire* and the match *auteurs / nom / voltaire* is evident, but the jump between *rentable* and *registres_recettes / recette*, less so. This is a potential source of confusion for the user, and an indication of these cases would be a valuable future addition to the interface. So as to aid the user in understanding the significance of a match, each can be viewed in context. Clicking on any match will direct the user to the database table viewer (detailed in section 2.2.5.4), adding, in the case or row matches, the column filters needed to display only the relevant rows.

2.2.5.3 Successive Tree Viewer

As described in section 2.2.4.1, a grammatical dependency parse tree is at the heart of CLAIRON's NL to SQL translation process. The series of parse and query tree visualisations displayed in this section of the interface shows how tree nodes are added, collapsed, removed, or

altered as a function of the presence of specific keywords and structural relationships. The indices included in the node labels map to the position of the corresponding tokens in the input question. This allows a user to follow the progress of the transformation and identify the cause of any misinterpretations. Even if an input is erroneous and the query generation process halts before producing a result, the interface will still display all steps up until the failure point. The choice not to hide errors further serves to underscore the fact that the system doesn't truly *understand* language; it has no intuition that would allow it to, for instance, still identify comparable entities even if one were labelled with an incorrect part of speech or erroneously parsed as the child of an unconnected node. Though the tree will not alter on every step for every input, the same set of steps is shown each time. The reason for this is twofold. Including every step every time leaves less space for confusion, from a user experience perspective, but also doesn't create a false impression of machine intuition. A person versed in SQL looking to translate an input question would more than likely focus on only the structures needed to render a solution, not giving a thought to negation or ordering, for instance, if the situation clearly did not call for it. CLAIRON, however, has no other recourse but to deterministically run through every check every time. Figure 2.2.5.3-1 shows the first three steps for the input question *Quelles sont les cinq pièces de Molière qu'on joue le plus entre 1680 et 1700 ?*.

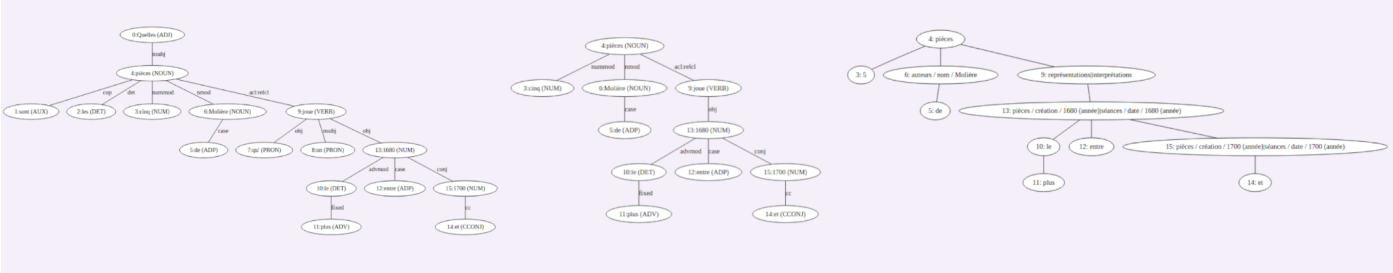

Figure 2.2.5.3-1. The first three parse and query trees produced for the input *Quelles sont les cinq pièces de Molière qu'on joue le plus entre 1680 et 1700 ?*. Note that the token position numbers shown in the query trees are 0-indexed.

2.2.5.4 Database Explorer

Though it is not necessary, when working with an NLIDB system, to have an understanding of the database schema, it does provide useful context. This final section of the interface gives the

user full access to the database tables (figure 2.2.5.4-1) as well as entity-relationship diagrams. This view into the machine interpretation and organisation of the data stands both to help the user reckon with the limits of the system and, on a much more basic level, support them in formulating their own SQL queries.

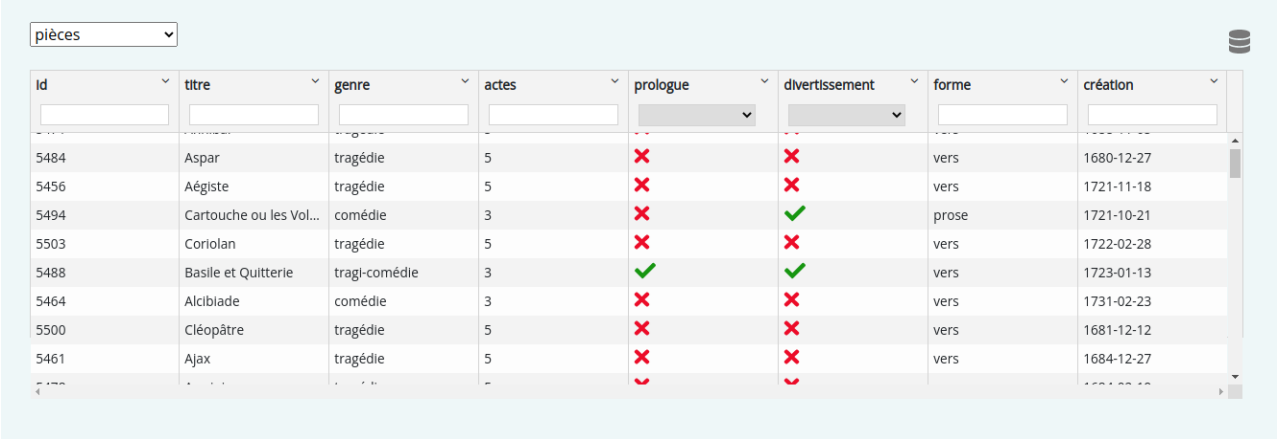

Figure 2.2.5.4-1: CLAIRON database explorer.

Whenever I am introducing the RCF data to a new group, I always start not from the tables, but from the manuscript registers themselves. The intention of this is to highlight the fact that datafication is a subjective and, crucially, lossy layer of abstraction—one that is currently not made visible in the interface. Though the digitised register collections in their entirety are available to the user, a valuable future addition to this section would be links to individual pages in the receipts and casting registers for each corresponding row.

2.2.5.5 Problems and Future Directions

In spite of the mechanisms in place to identify erroneous interpretations before they reach the user, some still slip through. My biggest concern with the CLAIRON UI is that users will uncritically accept the tabular results of these misinterpretations as true, without examining the SQL query, match mappings, or tree parses that might indicate otherwise. Take, for example, the input *Quels comédiens ont participé à le plus de créations ?*. The system is unable to properly interpret the question, returning a list of the actors who performed the most in plays for which

premiere dates are known, rather than in actual premiers. This is evident from reading the resulting SQL, but the results themselves still comprise, as they would have in the correct case, a list of actors. The rate of occurrence and severity of this kind of scenario, and the best ways to ensure a user leverages all the tools at their disposal to verify the results, are difficult to gauge without performing a user study.

Another potential sticking point is with questions where the returned results are not incorrect, per say, but where there is possible dissonance between how CLAIRON interpreted the question, and how the user did. Consider the question *Qui sont les comédiens qui entraînent les recettes les plus importantes ?*, in response to which the system returns a list of actors alongside the revenue of the highest grossing evening they ever performed on. While this is certainly one way of answering the question of who were the most financially successful actors, a user might reasonably have instead expected a list of actors ordered by total or average takings. Though the latter result could be achieved by rephrasing the question, this sort of interpretive disagreement—useful as it is for highlighting machine limitations—could be a source of confusion and frustration for users.

Both of these scenarios might be aided by providing a plain language explanation of the SQL query. I have so far avoided implementing this for two reasons. First, it is an added level of abstraction. Second, and more importantly, to fit within the translation-inspired and iterative logic of the system, this back translated 'fake' natural language would need to be able to then be re-ingested into the NL-SQL cycle. While, as previously mentioned, this sort of gradual convergence of syntax is interesting from a learning perspective, it runs counter to the system's emphasis on drawing attention to the distance and difference between human and machine logics and on pushing the user to navigate that mapping independently—with support from context, sure, but without explicit guidance in any one direction. Finding a way to integrate some sort of plain language query explanation to aid the comprehension of SQL novices without compromising the ethos of the system would be a valuable area for future exploration.

2.2.6 Evaluation

CLAIRON is a proof of concept. I have performed no formal evaluation. However, to provide some context for its scope, strengths, and weaknesses, I have attempted an informal comparison of both the back end NL to SQL translation system, through comparison with other notable NLIDBs, as well as the front end user interface, using the framework specified in section 1.1.

2.2.6.1 Back End

In their 2020 survey,[147] Kim et al. judged NLIDBs according to their performance across eleven benchmark datasets. I am unable to run similar tests on the CLAIRON system, as all of the existing benchmarks are in English, and the creation of a French language test set is outside of the scope of this thesis. As each set is also linked to a large database, the translation of an existing benchmark is likewise infeasible. What's more, evaluation hinging on benchmark accuracy often fails to assess the SQL coverage of a given system. As demonstrated by Affolter et al.,[148] many of the commonly used benchmarks consist of questions whose corresponding queries all map to a small set of SQL structures and formulations. Over 40% of the questions in the Yahoo! L6 corpus,[149] for instance, require only a string filter and potentially a simple JOIN. Similarly, not a single question in the GeoData250[150] corpus requires multiple subqueries, dates, or numerical ranges, and approximately 88% of the examples need only, at most, a JOIN, a string filter, and an ordering statement.

Affolter et al.'s alternative to benchmark-level comparison is evaluation based conceptual coverage. Recognizing that there are often many ways to structure a 'correct' query, they evaluate each system on input questions that, while they could refer to specific operators, generally require different query language concepts. In addition to joins, the ten input question ask that a given system be able to handle string filters, numeric inequality filters,[151] date filters,

[147] Kim. et al., "Natural language to SQL"
[148] Affolter et al., "A comparative survey of recent natural language interfaces for databases"
[149] https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
[150] Tang, Lappoon R., and Raymond J. Mooney. "Using multiple clause constructors in inductive logic programming for semantic parsing." In *European Conference on Machine Learning* (2001): 466-477.
[151] In the original paper, the term 'range' is used. Their example, however, is "All movies with a rating higher than 9", which does not require the two operator comparison that 'range' might imply (e.g. $3 < x < 9$). For this reason, I have opted for the term 'inequality'.

ordering, logic operators,[152] abstract concepts, aggregation,[153] negation, simple subqueries, and complex subqueries. In keeping with the didactic bent of the interface, I have augmented these categories with structure types drawn from concept breakdowns used to evaluate student SQL learning.[154] My evaluation is based on a comparison with three other NLIDBs—NALIR, because it is similarly dependency parsed base; ATHENA, because it is similarly concerned with handling domain specificity; and DBPal, because it is concerned with linguistic diversity, and can also be used to highlight the differences between learning and rules-based models. Figure 2.2.6.1-1 gives an overview of the results. As was the case for Affolter et al, not all of the systems of interest have publicly released their code, meaning the results are entirely based on information made available in related publications. Grey cells indicate a lack of information, green and red indicate supported and unsupported constructs respectively, and yellow flags complicated or marginal cases.

[152] Though the paper labels this operation category as 'union', it is specified that a potential solution could use either logical conjunction or disjunction.
[153] It is understood through their example—"What is the best movie of each genre?"— that this category also includes a system's capacity to deal with grouping.
[154] Migler, Andrew, and Alex Dekhtyar. "Mapping the SQL learning process in introductory database courses." in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (2020): 619-625; Ahadi, Alireza, Julia Prior, Vahid Behbood, and Raymond Lister. "A quantitative study of the relative difficulty for novices of writing seven different types of SQL queries." in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (2015): 201-206.

		ATHENA	DBPal	NaLIR	CLAIRON
Join	Simple join				
	Complex join				
Filter	String filter				
	Inequality filter				
	Date filter				
Logic	Conjunction				
	Disjunction				
Negation					
Distinct					
Limit					
Ordering					
Aggregation					
Grouping	Single grouping				
	Multiple grouping				
	Having				
Subquery	Simple subquery				
	Complex subquery				
Concept	Abstract concept				
	Derived field				

Figure 2.2.6.1-1. : NLIDB system comparison

2.2.6.1.1 Join

The ability to handle simple multi-table joins is essential for any functional NLIDB, as very rarely is all sought information confined to a single table. The syntax used for joins differs from system to system. NaLIR, ATHENA and DBPal all fully cross the tables and subsequently add filters with the relevant keys, whereas CLAIRON uses the ON keyword to specify inner joins. It is unclear if any of the comparison systems support left or right joins. CLAIRON currently does not—none of the sample questions which informed the system scope required special joins, and I am unsure of how the need for one would be expressed in natural language input. CLAIRON,

like ATHENA, cannot handle self joins, though it makes the point of informing the user when an input query contains specific keywords words implying that one should be used, so that they might try implementing it on their own using the SQL input side of the interface. It is unclear whether NaLIR or DBPal can handle self joins but, in the case of the latter, similarly performing learning-based systems such as GNN[155] cannot. The main difficulty I encountered when considering the implementation of self joins, was knowing which properties to enforce equality on, especially in the case of tables or views that include multiple foreign keys. Unlike NaLIR, there is no possibility of human-in-the-loop structural correction, but I do see investigating the ways in which this could reliably be inferred from the parse tree and the addition of support for self-joins as a feasible future addition.

2.2.6.1.2 Filters and Logic

As the most basic of operations, filters of all types are well supported. It should be noted that all published examples relating to dates deal only in years. It is unknown whether the comparison systems can handle partial dates, as CLAIRON can. Similarly, none of the published examples confirm that the different systems can interpret written numbers. Logic operators are similarly fairly well supported, though ATHENA is unable to handle expressions where both the left hand side and right hand side map to instances of the same ontology element.

2.2.6.1.3 Negation, Distinct, Limit, and Ordering

The authors of ATHENA explicitly state that the system is unable to handle negation. LIMIT and DISTINCT also seem to be missing from the grammar of the intermediate representation that dictates its query construction. It is unclear whether DBPal is able to handle either structure and, while NaLIR can handle DISTINCT, nothing in the published examples, test sets, or public source code suggest that it can render arbitrary numeric limits. All systems are able to handle basic ordering, though it is unclear whether the comparison systems can order on multiple properties, potentially in opposing directions. CLAIRON currently only supports ordering by a single element, though it can be complex (i.e. an aggregation, a mathematical operation etc.).

[155] Bogin, Ben, Jonathan Berant, and Matt Gardner. "Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing." in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019): 4560-4565.

2.2.6.1.4 Aggregation and Grouping

All systems can handle aggregation when it is correctly detected. However, detecting aggregation is one of the notable weaknesses of parse-based systems, as it relies on a generic set of 'trigger words'.[156] For example, given the input *Display the longest hospitilization period*, NaLIR fails to identify the token 'longest' as indicating an aggregation. As Kim et al. note,[157] this sort of error can be mitigated by extending token-keyword mappings for each dataset the system is used on, but this implies a certain amount of overhead. Though I have not performed any formal comparison, I believe it is possible that CLAIRON may be less vulnerable to this issue because of the lower degree of linguistic variation in structures pointing to an aggregation. In French, adverbs like 'hottest' or 'longest' do not exist; they must be written as *le plus chaud* or *le plus long* (*the most hot/long*). This greatly reduces the number of 'trigger' tokens that must be considered when trying to detect aggregations. However, the language does introduce different issues. In response to the input *Y a-t-il des auteurs dont la soirée au cours de laquelle ils sont joués ne rapporte jamais moins de 4000 livres ?,* for instance, it simply sees *jamais* as a negative adverb, rather than the aggregation that the idea of *never* implies. This produces the query—

```
select distinct auteurs.* from registres_recettes join séances on
(registres_recettes.id = séances.id_recettes) join représentations on
(séances.id = représentations.id_séance) join pièces on
(représentations.id_pièce = pièces.id) join attributions on (pièces.id
= attributions.id_pièce) join auteurs on (attributions.id_auteur =
auteurs.id) where en_livres(livres,sols,deniers)> 4000.0
```

—which returns all authors whose plays were at some point performed on an evening grossing more than 4000 livres. The correct query, returning authors whose plays were only ever performed on high revenue evenings is

```
select distinct auteurs.* from registres_recettes join séances on
(registres_recettes.id = séances.id_recettes) join représentations on
(séances.id = représentations.id_séance) join pièces on
(représentations.id_pièce = pièces.id) join attributions on (pièces.id
```

[156] Affolter et al., "A comparative survey of recent natural language interfaces for databases"
[157] Kim. et al., "Natural language to SQL"

```
= attributions.id_pièce) join auteurs on (attributions.id_auteur =
auteurs.id) group by auteurs.id having
min(en_livres(livres,sols,deniers))> 4000.0
```

ATHENA notably does not allow grouping on multiple properties, but, like CLAIRON, it can handle aggregation based filters, as HAVING is included in its grammar.

2.2.6.1.5 Subqueries

All systems are able to render singly nested queries, and all of the comparison systems support aliasing, though CLAIRON does not. Support for more complex formulations—correlated nested queries, for example— is, like other elements, limited by the structures of which systems are 'aware'. ATHENA's intermediate representation grammar only allows it to handle some forms of nesting, though this set was expanded in the later iteration, ATHENA++.[158] DBpal can only cope with query structures seen in the training data, and as the slot filling algorithm templates used to generate the dataset only covered one form of basic subquery, the system is unable to produce more complex forms. In DBPal's case, expanding coverage to more complex nesting means the diversification of the training data.

2.2.6.1.6 Concepts

Given that they do not map as directly or consistently to specific query language operations, concepts are particularly difficult for NLIDBs.[159] There are two distinct types of input that fit under this umbrella— abstract and oftentimes subjective concepts, and derived fields that are the product of calculation possibly combining several fields. Affolter et al's test question *List all great movies* is an example of the first type, as are the notions of 'major city' and 'major river' crop up in commonly used GeoData250 benchmark. A concept like 'population density', from that same dataset falls into the latter category.

ATHENA is easily able to handle tokens mapping to abstract concepts as long as they are encoded into the domain-specific ontology provided to the system. Because the system is premised on a user only having access to the ontology and not to the mapping between ontology

[158] Sen, Jaydeep, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. "ATHENA++ natural language querying for complex nested sql queries." in *Proceedings of the VLDB Endowment* 13, no. 12 (2020): 2747-2759.
[159] Ibid

elements and query language expressions, it is up to the expert who creates the mapping to provide a concrete definition. As NaLIR looks to establish a direct mapping between tokens and database elements rather than passing through an intermediate representation, it cannot handle this type of token. Given that one of the primary goals of the system is to deconstruct the encoded judgements and biases that underpin interfaces which present this kind of field as objective fact, and to communicate the limits of computing, CLAIRON handles this style of inputs by deliberately rejecting them. If asked, for instance, to list *les meilleures pièces de Molière*, CLAIRON responds that it cannot understand the token *meilleures*, which hopefully might prod the user to recognize the judgement and ambiguity of their question, and to reformulate it in a more specific way.

With calculated fields, learning-based systems like DBPal have an advantage over parse-based ones, as they do not depend on distinct keyword and structural encodings. Provided examples of a given type of field derivation have appeared in the training data, DL systems are able to handle a variety of complex structures. CLAIRON handles this kind of input by allowing a user to configure pseudocolumns which map to pre-created database functions. This is in some ways similar to ATHENA's handling, as it is effectively adding an element to the data ontology. It is worth noting that this functionality could be used to handle some cases that the system typically rejects, undermining its aim for transparency. Consider, for example, the concept of 'major city' introduced in the GeoData250 benchmark. It would be trivial to create a 'major' boolean pseudocolumn on a cities table which is the result of a comparison of the population against an arbitrary literal value and the mechanics of the function would remain hidden from the end user. Extending the user interface to render the calculations underlying these pseudocolumn-function mappings as part of the NL-SQL mapping displayed in the interface, or developing other strategies to mitigate this issue would be a valuable avenue for future work.

2.2.6.1.7 Limitations and Future Directions

Though I have not performed a formal error analysis, experimentally, the two most significant sources of error for CLAIRON are matching errors and incorrect parses—issues common to many NLIDBs.

Linguistic ambiguity means that there is more often than not no clear one-to-one mapping between input tokens and database elements. NaLIR's human-in-the-loop system approaches this problem by placing the onus of disambiguation on the user; evaluations of NaLIR run in non-interactive mode universally observe a significant drop in performance. One of the important factors in the success of ATHENA is that its intermediate domain-ontology imposes important constraints on possible mappings. Though CLAIRON employs metrics similar to NaLIR's match ranking system, unlike that system, it picks a single match before generating a query, as opposed to trying all potential matches. What's more, because this token-to-element mapping happens independently of the user, there is no way for them to correct the selection, at least from the natural language input side of the interface. A useful future extension to the system would be to generate and test all potential parse tree permutations, if the most probable one appears to be incorrect. The difficulty would be to do this in such a way that the volume of information does not become overwhelming for the user and the decision making process—and its faults—remain at the forefront.

CLAIRON, like all parse-based systems, also struggles with out-of-vocabulary words. Currently, only the input token and its lemmatized form are compared to the database entities. This means that a query such as *Quels rôles de tragédie sont joués par Monsieur Préville ?* works as expected, but the equivalent *Quels rôles tragiques sont joués par Monsieur Préville ?* does not produce the correct result as the match 'pièces/genre/tragédie' is not considered a potential for the token *tragiques*. Many systems utilise digital linguistic resources to supplement their vocabularies by considering synonyms and variants. CLAIRON allows basic synonym specification via the configuration file, but does not probe any external resources. This could be a valuable future addition, though important consideration would need to be given to how to expose this step of the process to the user, given that this would make the link between token and database entity name may be less plain. Interestingly, though using embeddings to represent and infer relationships between tokens means that DL systems tend to cope better than parse-based systems with out-of-vocabulary terms, they can fare worse on domain-specific terminology than mechanisms like ATHENA's ontology or CLAIRON's configuration. Because embeddings used to help identify matches are oftentimes not contextual, the systems are tripped up by terms that

are strongly conceptually related in a given context, but semantically distant in most contexts—'star' and 'rating' as used in a movie database, for example. [160]

The other primary source of error is bad parsing and tagging, which most often occurs when inputs are not quite as explicit as they might be. Consider the question *Combien de fois joue Bellemont au cours de la saison 1789-1790 ? (*figure 2.2.6.1.7-1). Here, the parser misinterprets *Bellemont* as a determiner, which means that CLAIRON does not consider it as a candidate for an entity match, consequently missing an essential part of the question.

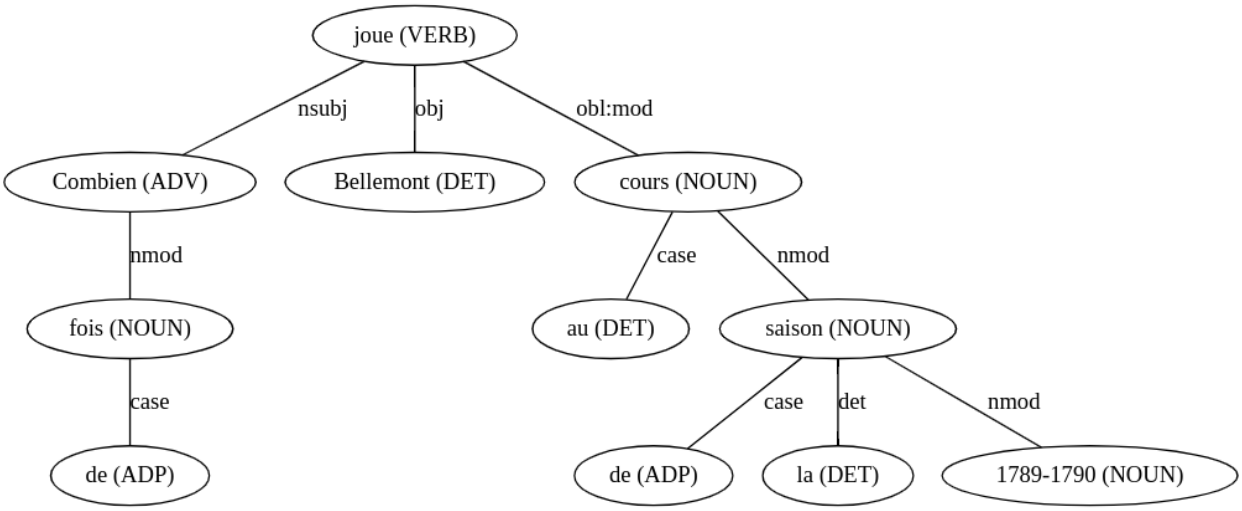

Figure 2.2.6.1.7-1. Parse tree for input *Combien de fois joue Bellemont au cours de la saison 1789-1790 ?*.

Unlike NaLIR, CLAIRON does not filter out invalid parse trees based on a grammar, though it does, as addressed in section 2.2.4.3.6, attempt some correction of invalid configurations. This means that, while it may be open to a more diverse variety of query structures, it is also more vulnerable to parsing errors. Even if they parse correctly, CLAIRON also struggles with questions that imply parallel structures but do not state them explicitly. For example, the current iteration of the system cannot correctly interpret the input *Lors de quelles saisons est-ce que Pierre Corneille rapporte plus que Racine ?* but has no issue with the equivalent question *Lors de quelles saisons est-ce que Pierre Corneille rapporte plus que rapporte Racine ?*. Improving the detection and handling of cases that call for the insertion of implicit nodes would be a

[160] Yaghmazadeh et al., "SQLizer"

valuable future direction, as this style of utterance is more natural. CLAIRON also does not do well with yes/no questions. It also does not select the aggregations it orders by. It most often opts to select binary comparisons as derived boolean fields instead of using them as filters. All of these things could be accounted for, but there is a never ending list of increasingly specific operations. With respect to CLAIRON in particular, the aim is not for the system to be able to handle all human inputs. Human logic and processes are not computational logic and processes; the system is designed to highlight this reality, not obfuscate it.

For parse-based systems, matching and parsing errors alike often come back to vocabulary limitations, and I see vocabulary expansion as a particularly interesting area for future work. When I prompted the CFRP team for the example input questions around which CLAIRON is based, I specifically requested that they ignore everything they may know about the database, its structures and the kind of queries that are possible. This means that their responses did not sound like they were designed with SQL in mind, as I find some of the example questions used by other systems can tend to — the question *What is the average length of stay of patients where age is 80?* generated for the DBPal training set, for instance, or the input *return me all the organizations in "North America"* included the widely used Microsoft Academic Search benchmark dataset.[161] Consequently, tokens that might be considered more 'SQL-esque' — *décroissant* or *croissant* for ordering, for example—are not included in the CLAIRON triggers or keyword mappings. Though this is in line with the notion of distinguishing human logics from computational ones that is central to the system, a future version supporting more hybrid inputs might be an interesting area for exploration, specifically in the context of instruction. The recently developed Hedy teaching programming language[162] scaffolds learning by gradually adding in syntactic elements such as brackets, indenting and quotations. This serves to create a throughline between human-like inputs, and the sometimes daunting syntax of formal programming languages. In the current iteration of CLAIRON, though the mapping NL-SQL is shown, it is left up to the user to infer the specifics of how query language syntax works if they

[161] Roy, Senjuti Basu, Martine De Cock, Vani Mandava, Swapna Savanna, Brian Dalessandro, Claudia Perlich, William Cukierski, and Ben Hamner. "The microsoft academic search dataset and kdd cup 2013." in *Proceedings of the 2013 KDD cup 2013 workshop* (2013): 1-6.
[162] Hermans, Felienne. "Hedy: a gradual language for programming education." in *Proceedings of the 2020 ACM conference on international computing education research* (2020): 259-270.

wish to input an SQL query of their own. Hedy style incremental support could both help make this process seem less daunting, and avoid misconceptions.

2.2.6.2 Front end

The evaluations of the Discovery and Graph Tools, and of the Faceted and Cross-Tab browsers in chapter one showed that they all struggled to some extent with data transformation hiding and were not necessarily very explicit about how their structure influenced knowledge construction. Placing the emphasis very squarely on the constructionist elements of data interrogation as it does, CLAIRON does not necessarily share many of the challenges common to the other CFRP interfaces, but its lack of layers of abstraction to aid with data understanding adds different challenges.

2.2.6.2.1 Transparency

Interface opacity, per Loup and Masure, is a product of layers of deliberately invisibilized computational abstraction.[163] CLAIRON aims for transparency by stripping away and deconstructing these layers. Because the interface works directly with the database tables—to which the user is given unfettered access—questions of data construction are limited to derived fields. Though icons flag these elements, the calculations underpinning them are not exposed. Issues of bias and situatedness as they relate to datafication practices are not addressed; this might be mitigated through the addition of links to the registers as seen in the Faceted browser.

The interface is constructed in such a way as to deliberately draw attention to the nuance and challenges of mapping human questions to database queries. It particularly focuses on the ambiguity inherent to this process, in direct opposition to the "reification of information"[164] performed by, and arguably required by, many common forms of visualisations. The carousel of tree iterations walks the user through the transformation process, and the choice to show the same steps every time emphasises the deterministic limitations of computation. In a similar vein, displaying all potential index choices serves to highlight data nuance and complexity, as well as draw attention to the difficulty the computer has with making decisions that would be obvious to

[163] Cellard & Masure, "Le design de la transparence"
[164] Drucker, "Humanistic theory and digital scholarship", 86

a human, because of its lack of deep contextual knowledge. Computers never have knowledge, only information.

2.2.6.2.2 Generativity

Like the Faceted browser, CLAIRON also creates space for interpretive work through multiplicity, but with the added potential for data manipulation and transformation. Exposing the database tables presents the full range of entry points into the data without imposed hierarchies or categories that expand beyond table structure. Lev Manovich[165] positions the database as the antithesis of narrative. Interfaces like the discovery tool or cross-tab browser that respond to a specific set of questions are beholden to an internal narrative, which is not the case for CLAIRON. Proximity to the database allows CLAIRON to maintain a flexibility and focus on plurality that must necessarily be simplified and collapsed in the case of more guided interfaces. It could be argued that CLAIRON's tabular results are less compelling, less likely to spark new ideas, than more narratively driven results. Not imposing limits on entry points or process types, however, does create more space for the user to curate an individualised experience.

The display mechanisms of the interface invite the user to question and deconstruct computational processes. Though they are unable to influence them, the index choice and parse tree evolution displays create the potential for error identification and critical reflection on the dissonance between human and machine conceptions of problems. The configuration file with its specifically customisable parse rules and function-to-field mappings further facilitates the interrogation and (re)definition of ontological elements and relationships.

Finally, CLARON's workflow affords iterative and reciprocally fed explorations. The query logs help users to mutate and build on previous queries; by explicitly showing the logical breakdown of input question components, they notably feed combinatorial explorations. Highlight mappings further scaffold inductive SQL learning.

2.2.6.2.3 Interpretability

The interaction design of CLAIRON is based on natural language translation tools. The aim of this is that the familiar interface configuration will cue users as to its use, but also potentially

[165] Manovich, "Database as a symbolic form"

encourage them to take a critical eye to the results—those who have experience with automatic translation systems will likely have some level of awareness of the ways in which they struggle with ambiguity and context. If the user is not able to intuit how to use the interface, details of the system's functionality are outlined in the embedded documentation.

Like the Graph tool, CLAIRON integrates two different modes of interaction to support users with different degrees of knowledge. Those who lack knowledge of SQL are able to use the plaintext input, and users comfortable with SQL are able to directly query the database in order to find answers to questions whose format or complexity exceeds the current capabilities of the translation system. The addition of the query log helps novice users to gradually build their query language skills by incrementally modifying previously successful queries. The single page scroll navigation and linking between elements such as index options and table rows further aims to help manage cognitive load by making it easy for a user to locate additional context and details while they iterate on a problem without having to exit their current frame.

Chapter 3 and Concluding Remarks: The Third Wave

Abstraction, formalisation, and modelling. These are the three mainstays of computational work identified by DH scholar Pierre Mounier that stand, he argues, to lead researchers to see their objects of study from a different perspective, to pursue alternatively focused explorations, and to define new avenues of exploration.[166] In these three consecutive phases I see a parallel to the first two so called 'waves' of Digital Humanities and a third of which I believe we might be at the precipice. In the first chapter of this thesis I explored existing approaches to the creation of foundationally humanistic interfaces and identified the core facets of transparency, generativity, and interpretability. I further proposed an alternative method of enacting these principles that looked to engage with the complexity of humanities problems and data by stripping away layers of abstraction to expose the computational processes whose implicit logics define digitally mediated knowledge production. The second chapter presented CLAIRON, an instantiation of this new style of interface. In this final chapter, I will expand on what I believe could constitute an already emergent third wave of DH, and provide some conclusions regarding how my work on CLAIRON aligns with the kind of paradigmatic and, crucially, pedagogical, shift that it represents and that is needed to move DH from alternative foci to new avenues of exploration.

The first wave of DH "looked backward as it moved forward",[167] replicating on a different scale existing scholarly—largely literary—techniques. As Mounier points out, Roberto Busa's *index thomisticus*, which many position as the genesis of DH, did not introduce any novel processes. The digital facilitated his explorations but did not shape them. The same could be said of the RCF interfaces discussed in chapter one. As evidenced by the work of Johannidès and Lancaster, no performance data accessible via the Discovery tool, for instance, could not be collated by hand, given enough time and resources. Much like in the case of Busa's concordance of the commonplace term 'in', it is the scale, and the distance from the source that this entails, that sets the digital work apart. We might consider Franco Moretti's elaboration of 'distant reading'[168] to be the distillation of this early DH focus on quantification and statistical analysis, a pivot emblematic of the first wave that aligns with Mounier's idea of an initial change of perspective.

[166] Mounier, Pierre. "Les Humanités numériques, gadget ou progrès?." *Revue du Crieur* 2 (2017): 144-159.
[167] "The Digital Humanities Manifesto 2.0". *Humanities Blast* (2009).
[168] Moretti, Franco. *Distant reading*. (Verso Books, 2013).

The second wave of DH is characterised by a shift in focus from the quantitative to the qualitative, giving way to the "interpretive, experiential, emotive, [and] generative"[169] and calling into question the supremacy of the arithmetic sublime. The rise in black, indigenous, and feminist scholarship in particular has brought greater attention to elevating voices at the margins, those who are most vulnerable to erasure or exploitation by aggregative practices. These communities have additionally promoted a greater scholarly consideration for the value of affect and embodiment. The transition from the first to the second wave has further been marked by a move away from digital environments rooted in patriarchal power structures—say, a European clergyman partnered with a computing company strongly tied to the military-industrial complex[170]—to critical contexts where the oppressive and emancipatory potential of the digital is at the forefront. This necessarily demands a recognition of the fact that computation is not merely a tool which we can exploit, but a force that fundamentally shapes our modern society.[171]

Milad Doueihi argues that this degree of enmeshment with the social, the political, and the ethical requires that we elaborate a broader theory of digital humanism. He asserts that the essential point of inflection is passing from "penser avec le numérique" to "penser le numérique."[172] For me, this is one of those powerful yet difficult to translate turns of phrase. I don't see 'thinking with the digital to thinking digitally" or even 'thinking with computers to thinking computationally' as adequately expressing the shift Doheidi identifies from using digital tools as just that—tools—to critically engaging with, though crucially not submitting to, digital logics and conceptualizations of the world. Stephen Ramsey's theorisation of 'humane computing' echoes this emphasis on "allow[ing] digital objects…to participate in humanistic discussions."[173] I believe that this focus on the "underlying computationality" of digital forms will, as predicted by David Berry in 2011, form the basis of a third wave of DH.[174] A shift towards greater domain equality on a theoretical level is needed to be able to recognise, deconstruct, and re-imagine the ways in which the digital alters humanities work and, equally, the undeniable humanity of computation. It is centering this not only methodological but

[169] "The Digital Humanities Manifesto 2.0"
[170] Mounier, "Les Humanités numériques, gadget ou progrès?."
[171] Doueihi, Milad. "Un humanisme numérique." *Communication langages* 1 (2011): 3-15
[172] ibid
[173] Ramsay, Stephen. "Humane Computation." *Debates in the Digital Humanities* (2016): 527-529.
[174] Berry, David M. "The computational turn: Thinking about the digital humanities." *Culture machine* 12 (2011).

epistemological dialogue that stands to afford, more than previously, the creation of novel paths of inquiry.

Berry's prediction, however, appears to have been premature. Of the 245 definitions of Digital Humanities[175] crowdsourced from DH students and scholars during the following year's 'Day of DH', only 37—in very generous terms—in any way mention engaging with computational logics from a humanistic perspective. For me, only two reflect the sort of bidirectional exchange that I had expected when I first came to DH—Francesca Benatti's "Researching the Humanities through digital perspectives, researching digital technologies from the perspective of the Humanities" and Geoffrey Rockwell's similar definition "The thoughtful use of computing in humanistic inquiry and the thinking through of computing from the perspective of the traditions of the humanities". More frequently, the definitions, even those that speak of working at the 'intersection' of domains, invoke the *application* of digital *tools* to the humanities, and insist on the humanistic *tradition*, ignoring that computation has a tradition and a corresponding worldview of its own. This is still thinking *with* the digital.

This is not to say that the history of DH lacks an awareness of, or interest in, thinking through the digital. Back when the Digital Humanities was still Humanities Computing, Willard McCarty identified "the questions raised by [...] algorithmic thinking, especially by the inevitable mismatch between any algorithm and data of the sort normal to the humanities" as the field's primary interest. In this, I read the necessity to cultivate an understanding of the logic, procedures, and epistemology of computation as a medium for the modelling of phenomena. Yet, an engagement with a bidirectional theoretical grounding that would facilitate this work seems to me to be missing from the majority of definitions of DH scholarship.[176] In her strikingly (and I would argue fittingly) titled essay 'The Radical, Unrealized Potential of Digital Humanities', Miriam Posner affirms that "it is not only about shifting the focus of projects so that they feature marginalized communities more prominently; it is about ripping apart and rebuilding the machinery of the archive and database so that it does not reproduce the logic that got us here in

[175] Data available at https://github.com/hepplerj/whatisdigitalhumanities/blob/master/dayofquotes_full.csv. This list includes data from 2010, 2011, 2012, and 2014. As many of the same individuals submitted definitions year to year, I selected only the subset corresponding to the 2012 event.
[176] Berra, Aurélien. "Faire des humanités numériques." *Read/Write Book* 2 (2012): 25-43.

the first place."[177] In a similar vein, the 2020 Turing Institute whitepaper 'The challenges and prospects of the intersection of humanities and data science'[178] stresses the importance of moving beyond the simple application of computational methods to humanities problems to the point of being able to reshape and subvert its rules and norms to enact a practice of computing that aligns with humanistic principles. This "radical unrealized potential" is third wave DH. This is Berry's vision for the evolution of the field that I would claim has so far failed to come to fruition. In the search for what is stunting the development of DH, I cannot help but to look to its pedagogy.

In their 2017 article, DH educators David Birnbaum and Alison Langmead state that it is crucial for humanists working with digital tools and methods to have some initiation into computing as otherwise "the risk of missed opportunity is great because the humanist may not know *what is possible computationally* and the programmer may not understand what is interesting to a humanities scholar—that is, it may be that neither knows how to ask the questions that would bridge the divide."[179] They then outline an approach to humanities computer science education that aims to emulate oral-proficiency-oriented foreign language pedagogy. Birnbaum and Langmead specifically advocate for instruction that actively avoids explicitly teaching "abstractions like numeric datatypes or control structures", focusing instead on task-specific programming exercises to build 'fluency' rather than computational abstractions that they equate to natural language grammars.[180] In addition to disregarding parity they initially invoke, I would argue that this approach is precisely the opposite of what is needed to prepare students to explore *what is possible computationally*. For all that we draw parallels between them, programming languages are not natural languages. There are no native Python speakers whose linguistic innovations define grammaticality. The grammar of a programming language dictates what is

[177] Posner, Miriam. "What's next: The radical, unrealized potential of digital humanities." *Miriam Posner's Blog* 27 (2015).
[178] McGillivray, Barbara, Beatrice Alex, Sarah Ames, Guyda Armstrong, David Beavan, Arianna Ciula, Giovanni Colavizza et al. "The challenges and prospects of the intersection of humanities and data science: A white paper from The Alan Turing Institute." (2020).
[179] Birnbaum, David J., and Alison Langmead. "Task-driven programming pedagogy in the digital humanities." in *New directions for computing education* (Springer, Cham, 2017): 63-85, emphasis mine.
[180] It bears mentioning that Birnbaum and Langmead are part of the group of digital humanists who principally view DH as the application of digital tools to humanities problems. For them "digital humanists perform humanities research, not computer science research and not, except as a means to an end, software development". This is markedly different from the picture of the field I am attempting to draw in this chapter.

possible as a 'speaker', not the other way around. To be able to push the limits of computational possibility, therefore, it is crucial to understand the abstract.

"The purpose of a course in programming is to teach people how to construct and analyse processes",[181] argued computing pioneer Alan Perlis. For him, programming was a medium particularly well suited to expressing processes. Correspondingly, he believed that computing curriculum should focus on understanding the process of mapping human understandings of systems and processes to a machine interpretable form, and interrogating the conceptual 'friction' that arises out of the imperfect translation between these cognitively distant representations. Additionally, Perlis, who was an early champion of what we now call 'CSforAll', advocated for wide reaching computing instruction, insisting on its potential to change the way in which *anyone* views, analyses, and describes the world.[182] Computing education in the humanities has taken rather an opposite turn to Perlis' imaginings, developing a fixation on 'building'[183] that backburners instruction in computational abstraction, seeing it as an unnecessary level of complexity that hinders the mastery of tools. This approach, however, is insufficient with respect to conveying the scope of computation and, by extension, the ways in which it can be transformed and adapted to humanistic inquiry. In response to this concern, Micheal Mateas proposes an alternative computing curriculum for 'new media' students, inspired by Perlis, that is premised on the development of procedural literacy (PL), which he defines as "the ability to read and write processes, to engage procedural representation and aesthetics, to understand the interplay between the culturally-embedded practices of human meaning-making and technically-mediated processes."[184] Contained in this definition I see two key facets of humanistic computing education that I believe are essential to moving DH forward into the third wave: computational thinking (CT) and critical computing.

---

[181] Greenberger, Martin. *Management and the Computer of the Future* (Wiley, 1962): 206.
[182] Guzdial, Mark. "Computer Science was always supposed to be taught to everyone, and it wasn't about getting a job: A historical perspective", *Computing Education Research Blog* (blog). 26 November, 2021. https://computinged.wordpress.com/2021/11/26/computer-science-was-always-supposed-to-be-taught-to-everyone-but-not-about-getting-a-job-a-historical-perspective/
[183] Ramsay, Stephen. "Who's in and who's out." in *Defining digital humanities* (Routledge, 2016): 255-258.
[184] Mateas, Michael. "Procedural literacy"

Given the focus on reading procedure, Mark Guzdial suggests that one conception of PL might be as computational thinking.[185] Defined by Cuny et al. as *"*the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent*"*,[186] CT can be viewed as the act of thinking through and managing Perlis' cognitive 'friction'. One could argue that the conceptual sandpaper of the nuance, heterogeneity, and ambiguity that characterises humanities sources with regard to the precise, discrete, and systematised data and problem descriptions required by the digital makes CT all the more important in the digital humanities. Research in a CS context has shown that emphasising CT through explicit scaffolding of problem solving not only benefits concrete skill acquisition, but also improved self-efficacy and reinforced the growth mindset of novice learners.[187] In her work on conversational programmers, who are in many ways similar to DH practitioners in that they may not be the ones writing the solutions but they need to be able to discuss computational procedures on an abstract level within a domain specific context, Kathryn Cunningham found that they were best served by an approach that forefronted high level 'program plans' which prioritised procedures over implementations.[188] There's a reason that CSUnplugged[189], which teaches computer science without computers, is one of the most widely used resources in k-12 computing curriculum—as (probably not[190]) Edgar Djikstra argued, computer science is no more about computers than astronomy is about telescopes. Knowing the syntax for a Java array is not essential. Knowing that lists exist in the abstract and are a way of organising information that imposes specific constraints and assumptions about the data they contain, is. Understanding how to use pointers to implement a network graph isn't nearly as important as being able to recognize problems that might be suited to being modelled as a network as well as what loss might be incurred in translating the source data and questions to

---

[185] Guzdial, "Computer Science was always supposed to be taught to everyone, and it wasn't about getting a job: A historical perspective"

[186] Cuny, Jan, Larry Snyder, and Jeannette M. Wing. "Demystifying computational thinking for non-computer scientists." *Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/~ CompThink/resources/TheLinkWing. pdf* (2010).

[187] Loksa, Dastyni, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. "Programming, problem solving, and self-awareness: Effects of explicit guidance." in *Proceedings of the 2016 CHI conference on human factors in computing systems* (2016): 1449-1461.

[188] Cunningham et al, "Avoiding the Turing Tarpit"

[189] Bell, Tim, Jason Alexander, Isaac Freeman, and Mick Grimley. "Computer science unplugged: School students doing real computing without computers." *The New Zealand Journal of Applied Computing and Information Technology* 13, no. 1 (2009): 20-29.

[190] The attribution of this quote is debated.

that form. DH computing curriculum appears to so far have been rather telescope heavy, but it is skills in computational thinking that are needed to elaborate computational solutions to novel humanities problems. McCarty argues that "research in humanities computing begins…when tools become models"; moving from a practice-focused to CT-focused curriculum is effecting precisely this transformation.

The second essential facet–critical computing–demands a recognition of computing as a political practice.[191] This asks that we consider the larger context of the systems of power and oppression from which it emerged and which it perpetuates. Critical CS instruction must crucially bring to light computing's limits and non-neutrality, and help students build the skills needed to dismantle systems of oppression rather than reinforce them.[192] As addressed in chapter one, there is a pervasive and longstanding association between deterministic computation and rational objectivity. Mateas counters this by positioning computing as "a universal representational medium for describing structure and process", a deliberate ontological choice that recognizes the need to examine the rhetoric embodied by different instantiations of a process.[193] Procedural literacy, it follows, includes the ability to deconstruct different encodings, peeling through the layers of abstraction to come to grips with the ways in which computational representations and tools encode and propagate specific values and worldviews reflective of the contexts of their development. Berry speaks of redefining what "reading and writing actually should mean in a computational age."[194] If computational thinking is a new way of thinking about writing, or perhaps more justly, translation, then PL as applied to critically conscious computing is reading for the third wave DH practitioner. It is those who are able to read and deconstruct the procedures of frequency distribution or neural network based sentiment analysis who will be able to understand the biases underpinning what may seem, from their productions, to be indistinguishable systems. It is those who are able to read the gaps and bias in datification

[191] Ko, Amy J. "Programming as Cognition, Programming as Politics" University of Colorado Boulder, Institute for Cognitive Science. Virtual. November 19, 2021
[192] Ko, Amy J., Alannah Oleson, Neil Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. "It is time for more critical CS education." *Communications of the ACM* 63, no. 11 (2020): 31-33.
[193] Mateas, "Procedural literacy"
[194] Berry, "The computational turn"

procedures and imagine alternative encodings that will be able to identify and counter the harm done by policies based on deficit data.[195]

Though not primarily intended as a teaching tool, emphasising procedural literacy was at the forefront of my mind in designing the CLAIRON interface. As addressed in chapter two, the NL to SQL translation at its centre looks to bring to the fore the process of mapping human conceptions of problems to their corresponding—though not necessarily equivalent—computational expressions. Usually discussed more abstractly in a postsecondary context, decomposition, abstraction and pattern recognition, and procedural problem solving are often used as the four cornerstones of practical CT instruction in recently adopted k-12 computing curriculum[196]; the various sections of the interface each look to address one or more of these facets. The highlight mapping and tree structure both help show the breakdown of problems and aid users to abstract upwards and identify their common parts. Pattern-oriented and procedural thinking are further scaffolded by the query logs. All sections, most notably the tree transformation and match sets, invite a critical look at the computational processes underlying the interface. Users are able to navigate through each step of the process to find errors or ambiguous cases, and the interface makes no effort to gloss over the limits of computation—quite the opposite. Having been around since 2008, the trajectory of the Comédie-Française Registers Project has aligned, to an extent, with the waves of DH. The creation of the initial databases and early tools correspond to the first wave focus on mass datafication, distant reading and statistical calculation. The expansion of the data assemblage to include more diverse and qualitative sources, as well as the research-creation workshops where students explored questions of data bias and remediation, among others, echo the characteristics of the second wave. CLAIRON is a foray into the intersection of the CFRP and what I believe to be the emergent third wave of DH.

Why now, though? is of course the obvious question with respect to third wave DH. Since Berry's prediction in 2011, and even since Posner's speculative article in 2015, I maintain there

[195] Walter, Maggie. "The voice of indigenous data: beyond the markers of disadvantage." *Griffith Review* 60 (2018): 256-263.
[196] See, for example, the British Columbia curriculum rolled out in 2016 https://codebc.ca/wp-content/uploads/2017/01/TTP_22pages_LighthouseLabs-v3.pdf

has been a significant shift, not in the humanities, but in the CS community, that is conducive to the kind of stronger and more egalitarian partnership that DH has been missing. There has been growing concern over the past few years that current CS education is taught in too much of a vacuum. For all that CS programs may make an effort to valorise the sort of high-level abstract systems thinking that would facilitate navigating the nuances of mapping human logics to computational ones, the development of these skills is built around toy problems and de-contextualized datasets. These resources, in an effort to simplify learning, eliminate any sort of Perlisian critical friction by minimising to the point of invisibility the conceptual distance between the problem and a computationally expressed solution. This decontextualized approach has the effect of naturalising and erasing the worldview encoded in the transformation in a global context where computing is more intimately intertwined with the diverse societal facets than ever before.[197] Momentum is building for curriculum reform anchored in interdisciplinarity.[198]

On one level, this refers to an increasing focus on the *mise en scène* of CS learning. Despite the recognition that students learn better when they feel more connected to the material, CS educators have historically shied away from contextualised problems out of concern for cognitive load.[199] What this fails to recognize, however, is that context is not the same as framing. Truly contextualised pedagogy does not view the CS problem as the core of the exercise and conceive of the contextualization as an extra barrier that needs to be overcome. Rather, it sees it as another source of knowledge—something which has only just begun to be understood and leveraged. For example, Guzdial and Shreiner's recent work in task specific programming languages (tsp languages) derives from the notion that students learn computation most effectively when they are also engaged in domain specific learning.[200] One of the tsp languages

[197] Connolly, Randy. "Why computing belongs within the social sciences." *Communications of the ACM* 63, no. 8 (2020): 54-59.
[198] ibid; Ko er. Al, "It is time for more critical CS education."; Guzdial, Computer Science was always supposed to be taught to everyone, and it wasn't about getting a job: A historical perspective"
[199] Craig, Michelle, Jacqueline Smith, and Andrew Petersen. "Familiar contexts and the difficulty of programming problems." in *Proceedings of the 17th Koli calling international conference on computing education research* (2017): 123-127.
[200] Guzdial, Mark, and Tamara Shreiner. "Integrating computing through task-specific programming for disciplinary relevance: Considerations and examples." in *Computational Thinking in Compulsory Education: A Pedagogical Perspective,* Aman Yadav and Ulf Dalvad Berthelsen (Eds.). Routledge Taylor & Francis Group (2021): 171-190

and corresponding tools they developed, for example, focuses on visualising historical data.[201] The impetus for its creation was the fact that Schreiner's previous work revealed that the vast majority of students struggle to make sense of visualisations of social sciences data and largely did not consider, in their evaluation, important humanities questions of sourcing, collection, processing, and display methodologies.[202] Much like CLAIRON, Guzdial and Schreiner's tool allows users to interact with data using familiar controls that don't require programming knowledge, but also grants access to manipulable underlying information representation—in this case, the JSON corresponding to the displayed bar graphs and their data. Importantly, the shape of the tool is context-driven; it is premised on the binary comparison of graphs "because historical inquiry often begins with two pieces of data or accounts that do not agree."[203] It is not a tool to be *applied* to the humanities, but one driven by domain specific theory that also looks to engage with computational constructedness such that domain and CS learning advance in parallel, neither a hindrance to the other.

On a second broader and more abstract level, the drive to move toward a more integrated computing education is motivated by the need to redress the historical emphasis that has been placed on *what* is taught rather than *why* or *how*.[204] In her 2019 Koli Calling keynote "21st Century Grand Challenges for Computing Education",[205] Amy Ko points to the fact that society is increasingly looking to technology to solve some of its biggest issues—climate change, fake news, minority discrimination, etc.—without addressing the role it had in creating them, or equipping technologists with the skills needed to approach these kinds of problems holistically. Ko emphasises that CS education has a tendency to fetishize the machine, valuing power and efficiency over humanity to a degree that promotes a skewed perspective of what is possible in the realm of computing–misinterpretations that have regrettably spilled over into the public consciousness. To prevent this sort of thing from continuing to occur, what is needed, says Ko, is an increased focus on recognizing and respecting the limits of computing—something which

[201] Shreiner, Tamara L., Mark Guzdial, and Bahare Naimipour. "Using participatory design research to support the teaching and learning of data literacy in social studies." in *Paper Presented at the College and University Faculty Assembly of the National Council for the Social Studies 2021 Annual Conference Virtual Conference* (2021).
[202] Shreiner, Tamara L. *Framing a model of democratic thinking to inform teaching and learning in civic education* (University of Michigan, 2009).
[203] Guzdial & Shreiner, "Integrating computing through task-specific programming for disciplinary relevance": 181
[204] Ko et al., "It is time for more critical CS education"
[205] Ko, Amy J. 2019 "21st Century Grand Challenges for Computing Education". Koli Calling Computing Education Research Conference. Koli, Finland. November 22, 2019

messy and ambiguous humanities contexts excel at bringing to light. It is likewise essential to bring ethics to the forefront as a topic that should be a constant consideration rather than an afterthought. As Doueihi rightly points out, computation is inextricably embedded in societal operations. We can no longer afford to forget that some problems are undecidable not because they are NP hard but because they are morally hard. Existing interventions within CS have tended to introduce individual human-centric examples, but it is the introduction of overarching humanistic epistemologies and methodologies that provide the robust theoretical framework needed to counter a *system*. It is the deep integration of the parallel abstractions of humanities formalisms and computational procedures that are needed to facilitate the deconstruction of harmful practices and the development of alternative procedures for the treatment of complex, ambiguous, and nuanced data and problems.

Students are not being taught to think critically about how the technology they produce concretely impacts the wider world, and educators see the scattered introduction of isolated ethics courses as an insufficient means of addressing the problem.[206] The push is for long term interventions and an integrated curriculum that promotes an awareness of context and worldview. Randy Conolly argues that computing needs to "move to the edge[s of the bounds of the discipline] and to participate in the rich academic biodiversity that happens where computing interacts with other disciplines." He insists on the fact that this shift cannot be an "exotic vacation" but rather must constitute a permanent relocation. Further, this shift must avoid "a colonizing ideology that sees computational thinking as the best way to understand and inhabit this world" and instead focus on embracing methodological, theoretical, and epistemological pluralism.[207] Much of DH's approach to computing education had been calqued from CS and has remained similarly decontextualized and siloed. DH courses addressing visualisation tend to focus on the mastery of tools that are embedded in a particular information display logic without either addressing it or discussing alternatives. Courses in programming languages do not tend to undertake the same sort of media archaeology oriented explorations of the development and evolution of control structures and algorithms that might be applied to various textual forms. But they could. Achieving a tradition of scholarship that engages with 'computationality', I would

[206] ibid; Bruce, Kim B. "Five big open questions in computing education." *ACM Inroads* 9, no. 4 (2018): 77-80.
[207] Connolly, "Why computing belongs within the social sciences"

argue, requires a degree of domain equality on the level of theoretical engagement. This is what guided the development of CLAIRON. This is what DH pedagogy is missing. This is third wave Digital Humanities.

Bibliography

Affolter, Katrin, Kurt Stockinger, and Abraham Bernstein. "A comparative survey of recent natural language interfaces for databases." *The VLDB Journal* 28, no. 5 (2019): 793-819.

Ahadi, Alireza, Julia Prior, Vahid Behbood, and Raymond Lister. "A quantitative study of the relative difficulty for novices of writing seven different types of SQL queries." In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (2015): 201-206.

Baik, Christopher, Hosagrahar V. Jagadish, and Yunyao Li. "Bridging the semantic gap with SQL query logs in natural language interfaces to databases." *2019 IEEE 35th International Conference on Data Engineering (ICDE)* (2019): 374-385.

Bardzell, Jeffrey, and Shaowen Bardzell. "Humanistic Hci." *Interactions* 23, no. 2 (2016): 20-29.

Becker, Brett A. "An effective approach to enhancing compiler error messages." In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (2016): 126-131.

Bell, Tim, Jason Alexander, Isaac Freeman, and Mick Grimley. "Computer science unplugged: School students doing real computing without computers." *The New Zealand Journal of Applied Computing and Information Technology* 13, no. 1 (2009): 20-29.

Berra, Aurélien. "Faire des humanités numériques." *Read/Write Book* 2 (2012): 25-43.

Berry, David M. "The computational turn: Thinking about the digital humanities." *Culture machine* 12 (2011).

Bertin, Jacques. *Sémiologie graphique: les diagrammes, les réseaux, les cartes*. De Gruyter Mouton, 1973.

Beyer, Kurt W. *Grace Hopper and the invention of the information age*. Mit Press, 2012.

Biet, Christian, Sara Harvey, and Agathe Sanjuan. "Postface–Le Programme RCF, de l'archéologie à la futurologie." In *Données, recettes & répertoire: La scène en ligne (1680-1793)*. MIT Press, 2020.

Birnbaum, David J., and Alison Langmead. "Task-driven programming pedagogy in the digital humanities." In *New directions for computing education*, 63-85. Springer, Cham, 2017.

Bogin, Ben, Jonathan Berant, and Matt Gardner. "Representing Schema Structure with Graph Neural Networks for Text-to-SQL Parsing." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019): 4560-4565.

Bogost, Ian. *The rhetoric of video games*. MacArthur Foundation Digital Media and Learning Initiative, 2008.

Borges, Jorge Luis. "Of exactitude in science." In *A Universal History of Infamy*, 31. Penguin Books, 1975.

Bruce, Kim B. "Five big open questions in computing education." *ACM Inroads* 9, no. 4 (2018): 77-80.

Burdick, Anne, Johanna Drucker, Peter Lunenfeld, Todd Presner, and Jeffrey Schnapp, "A Short Guide to the Digital_Humanities" In *Digital_Humanities*, 121-136. Mit Press, 2016.

Burnett, Margaret, Anicia Peters, Charles Hill, and Noha Elarief. "Finding gender-inclusiveness software issues with GenderMag: A field investigation." *Proceedings of the 2016 CHI conference on human factors in computing systems* (2016): 2586-2598.

Cellard, Loup, and Anthony Masure. "Le design de la transparence." *Multitudes* 4 (2018): 100-111.

Chamberlin, Donald D., and Raymond F. Boyce. "SEQUEL: A structured English query language." In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control* (1974): 249-264.

Clairon, Hippolyte *Mémoires d'Hyppolite Clairon, et réflexions sur l'art dramatique.*F. Buisson: Paris, 1799.

Connolly, Randy. "Why computing belongs within the social sciences." *Communications of the ACM* 63, no. 8 (2020): 54-59.

Craig, Michelle, Jacqueline Smith, and Andrew Petersen. "Familiar contexts and the difficulty of programming problems." In *Proceedings of the 17th Koli calling international conference on computing education research* (2017): 123-127.

Crawford, Kate. "Power." In *The Atlas of AI*, 217-227, Yale University Press, 2021

Cunningham, Kathryn, Barbara J. Ericson, Rahul Agrawal Bejarano, and Mark Guzdial. "Avoiding the Turing tarpit: Learning conversational programming by starting from code's purpose." In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021): 1-15.

Cuny, Jan, Larry Snyder, and Jeannette M. Wing. "Demystifying computational thinking for non-computer scientists." *Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/~ CompThink/resources/TheLinkWing. pdf* (2010).

Diderot, Denis, and Jean Le Rond d'Alembert. *Encyclopédie, ou, Dictionnaire raisonné des sciences, des arts et des métiers*. Vol. 1. Pergamon Press, 1776.

Diderot, Denis. *Paradoxe sur le comédien: ouvrage posthume*. A. Sautelet, 1830.

"The Digital Humanities Manifesto 2.0". *Humanities Blast* (2009).

D'Ignazio, Catherine, and Lauren F. Klein. "Feminist data visualization." *Workshop on Visualization for the Digital Humanities (VIS4DH)*, 2016.

D'Ignazio, Catherine. "Creative data literacy: Bridging the gap between the data-haves and data-have nots." *Information Design Journal* 23, no. 1 (2017): 6-18.

D'Ignazio, Catherine, and Lauren F. Klein. *Data feminism*. MIT press, 2020.

Dijkstra, Edsger W. "How do we tell truths that might hurt?." *ACM Sigplan Notices* 17, no. 5 (1982): 13-15.

Doueihi, Milad. "Un humanisme numérique." *Communication langages* 1 (2011): 3-15.

Drucker, Johanna. "Humanities approaches to interface theory." *Culture machine* 12 (2011).

Drucker, Johanna. "Humanities approaches to graphical display." *Digital Humanities Quarterly* 5, no. 1 (2011): 1-21.

Drucker, Johanna. "Humanistic theory and digital scholarship." *Debates in the digital humanities* 150 (2012): 85-95.

Drucker, Johanna. *Graphesis: Visual forms of knowledge production*. Cambridge, MA: Harvard University Press, 2014.

Dumesnil, Marie-Françoise. *Mémoires de Mlle Dumesnil, en réponse aux mémoires d'Hippolyte Clairon*. Ponthieu: Paris, 1823.

Escobar Varela, Miguel. "Introduction: Pursuit of Theater's Digital Traces" In *Theater as Data: Computational Journeys into Theater Research*, 1-20. Ann Arbor: University of Michigan Press, 2021.

Essinger, James. *Ada's algorithm: How Lord Byron's daughter Ada Lovelace launched the digital age*. Melville House, 2014.

Ferré, Xavier, Natalia Juristo, Helmut Windl, and Larry Constantine. "Usability basics for software developers." *IEEE software* 18, no. 1 (2001): 22-29.

de Fieux Mouhy, Charles. *Tablettes dramatiques contenant l'abrege de l'histoire du theatre francois, l'etablissement des theatres a Paris, un dictionnaire des pieces... Avec 3 Supplem*. Sebast. Jorry, 1752.

Filippi, Florence and Sara Harvey. "Émergence du vedettariat théâtral en France (xviie-xixe siècles)" In *Le Sacre de l'acteur. Émergence du vedettariat théâtral de Molière à Sarah Bernhardt,* 11-26. Paris, Armand Colin, « Collection U », 2017.

Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. "Datasheets for datasets." *Communications of the ACM* 64, no. 12 (2021): 86-92.

Greenberger, Martin. *Management and the Computer of the Future*. Wiley, 1962.

Guzdial, Mark, and Tamara Shreiner. "Integrating computing through task-specific programming for disciplinary relevance: Considerations and examples." In *Computational Thinking in Compulsory Education: A Pedagogical Perspective,* Aman Yadav and Ulf Dalvad Berthelsen (Eds.), 171-190. Routledge Taylor & Francis Group, 2021.

Guzdial, Mark. "Computer Science was always supposed to be taught to everyone, and it wasn't about getting a job: A historical perspective", *Computing Education Research Blog* (blog). 26 November, 2021. https://computinged.wordpress.com/2021/11/26/computer-science-was-always-supposed-to-be-taught-to-everyone-but-not-about-getting-a-job-a-historical-perspective/

Hall, Kyle Wm, Adam J. Bradley, Uta Hinrichs, Samuel Huron, Jo Wood, Christopher Collins, and Sheelagh Carpendale. "Design by immersion: A transdisciplinary approach to problem-driven visualizations." *IEEE transactions on visualization and computer graphics* 26, no. 1 (2019): 109-118.

Haraway, Donna. "Situated knowledges: The science question in feminism and the privilege of partial perspective." In *Feminist theory reader*, 303-310. Routledge, 2020.

Harvey, Sara, and Agathe Sanjuan. "Le projet des registres journaliers de la Comédie-Française: Les humanités numériques, dialogue entre les mondes de la recherche et de la documentation." *Bulletin des bibliothèques de France* 9 (2016): 102-109.

Hermans, Felienne. "Hedy: a gradual language for programming education." In *Proceedings of the 2020 ACM conference on international computing education research* (2020): 259-270.

Hmelo, Cindy E., and Mark Guzdial. "Of black and glass boxes: scaffolding for doing and learning." In *Proceedings of the 1996 international conference on Learning sciences* (1996): 128-134.

Isaacson, Walter. "The Intersection of the Humanities and the Sciences." *43rd Jefferson Lecture in the Humanities, National Endowment for the Humanities*. 2014.

Jammi, Manasa, Jaydeep Sen, Ashish R. Mittal, Sagar Verma, Vardaan Pahuja, Rema Ananthanarayanan, Pranay Lohia, Hima Karanam, Diptikalyan Saha, and Karthik Sankaranarayanan. "Tooling framework for instantiating natural language querying system." In *Proceedings of the VLDB Endowment* 11, no. 12 (2018): 2014-2017.

Joannidès, Alexandre. *La Comédie-Française de 1680 à 1900: dictionnaire général des pièces et des auteurs*. Plon-Nourrit, 1901.

de Jong, Ton. "Scaffolds for scientific discovery learning." *Handling complexity in learning environments: Theory and research* (2006): 107-128.

Kim, Hyeonji, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. "Natural language to SQL: where are we today?." *Proceedings of the VLDB Endowment* 13, no. 10 (2020): 1737-1750.

Klein, Lauren. "What Data Visualization Reveals: Elizabeth Palmer Peabody and the Work of Knowledge Production." (2022).

Ko, Amy J. 2019 “21st Century Grand Challenges for Computing Education”. Koli Calling Computing Education Research Conference. Koli, Finland. November 22, 2019

Ko, Amy J., Alannah Oleson, Neil Ryan, Yim Register, Benjamin Xie, Mina Tari, Matthew Davidson, Stefania Druga, and Dastyni Loksa. "It is time for more critical CS education." *Communications of the ACM* 63, no. 11 (2020): 31-33.

Ko, Amy J. “Programming as Cognition, Programming as Politics” University of Colorado Boulder, Institute for Cognitive Science. Virtual. November 19, 2021

Lancaster, Henry Carrington. *The Comédie Française, 1680-1701: Plays, Actors, Spectators, Finances*. No. 17. Johns Hopkins Press, 1941.

Li, Fei, and Hosagrahar V. Jagadish. "Constructing an interactive natural language interface for relational databases." In *Proceedings of the VLDB Endowment* 8, no. 1 (2014): 73-84.

Loksa, Dastyni, Amy J. Ko, Will Jernigan, Alannah Oleson, Christopher J. Mendez, and Margaret M. Burnett. "Programming, problem solving, and self-awareness: Effects of explicit guidance." In *Proceedings of the 2016 CHI conference on human factors in computing systems* (2016):1449-1461 .

Manovich, Lev. "Database as a symbolic form." *Museums in a digital age* (1998): 64-71

Masure, Anthony. "Vers des humanités numériques «critiques»." *Repéré à dlis. hypotheses. org/2088* (2018)

Mateas, Michael. "Procedural literacy: educating the new media practitioner." *On the Horizon* 13, no. 2 (2005): 101-111.

McGillivray, Barbara, Beatrice Alex, Sarah Ames, Guyda Armstrong, David Beavan, Arianna Ciula, Giovanni Colavizza et al. "The challenges and prospects of the intersection of humanities and data science: A white paper from The Alan Turing Institute." (2020).

Migler, Andrew, and Alex Dekhtyar. "Mapping the SQL learning process in introductory database courses." In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (2020): 619-625.

Miller, George A. "WordNet: a lexical database for English." *Communications of the ACM* 38, no. 11 (1995): 39-41.

Moretti, Franco. *Distant reading*. Verso Books, 2013.

Mounier, Pierre. "Les Humanités numériques, gadget ou progrès?." Revue du Crieur 7, no. 2 (2017): 144-159.

Munzner, Tamara. “Rules of Thumb” In *Visualization analysis and design*, 116-141. CRC press, 2014.

Noble, Safiya Umoja. "Algorithms of oppression." In *Algorithms of Oppression*. New York University Press, 2018.

Perez, Caroline Criado. *Invisible women: Data bias in a world designed for men*. Abrams, 2019.

Popescu, Ana-Maria, Oren Etzioni, and Henry Kautz. "Towards a theory of natural language interfaces to databases." *Proceedings of the 8th international conference on Intelligent user interfaces* (2003): 149-157.

Posner, Miriam. "What's next: The radical, unrealized potential of digital humanities." *Miriam Posner's Blog* 27 (2015).

Quamen, Harvey, and Jon Bath. "Databases." In *Doing Digital Humanities*, 181-198. Routledge, 2016.

Ramsay, Stephen. "Humane computation." *Debates in the digital humanities* (2016): 527-529.

Ramsay, Stephen. "Who's in and who's out." In *Defining digital humanities*, 255-258. Routledge, 2016.

Ravel, Jeffrey S. "The Comédie-Française by the Numbers, 1752–2020." In *Données, recettes & répertoire: La scène en ligne (1680-1793)*. MIT Press, 2020.

Roy, Senjuti Basu, Martine De Cock, Vani Mandava, Swapna Savanna, Brian Dalessandro, Claudia Perlich, William Cukierski, and Ben Hamner. "The microsoft academic search dataset and kdd cup 2013." In *Proceedings of the 2013 KDD cup 2013 workshop* (2013): 1-6.

Saha, Diptikalyan, Avrilia Floratou, Karthik Sankaranarayanan, Umar Farooq Minhas, Ashish R. Mittal, and Fatma Özcan. "ATHENA: an ontology-driven system for natural language querying over relational data stores." *Proceedings of the VLDB Endowment* 9, no. 12 (2016): 1209-1220.

Sanjuan, Agathe, and Martial Poirson. *Comédie-Française: une histoire du théâtre*. Seuil, 2018.

Schuwey, Christophe. "Humanités numériques et études littéraires : une question d'interfaces" *La lettre de l'InSHS* (2018): 25-27

Sen, Jaydeep, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. "ATHENA++ natural language querying for complex nested sql queries." In *Proceedings of the VLDB Endowment* 13, no. 12 (2020): 2747-2759.

Schieber, Philip. "The wit and wisdom of Grace Hopper." *The OCLC Newsletter* 167 (1987).

Shneiderman, Ben. "The eyes have it: A task by data type taxonomy for information visualizations." In *The craft of information visualization*, 364-371. Morgan Kaufmann, 2003.

Shreiner, Tamara L. *Framing a model of democratic thinking to inform teaching and learning in civic education*. University of Michigan, 2009.

Shreiner, Tamara L., Mark Guzdial, and Bahare Naimipour. "Using participatory design research to support the teaching and learning of data literacy in social studies." In *Paper Presented*

at the College and University Faculty Assembly of the National Council for the Social Studies 2021 Annual Conference Virtual Conference. 2021.

Sinclair, Stéfan, Stan Ruecker, Milena Radzikowska, and I. N. K. E. Inke. "Information visualization for humanities scholars." *Literary Studies in the Digital Age-An Evolving Anthology* (2013).

Stevens-Martinez, Kristin and Mark Guzdial. “Live Coding”. The CE-ED Podcast. February 3, 2020, https://csedpodcast.org/blog/season1_episode4/

Tang, Lappoon R., and Raymond J. Mooney. "Using multiple clause constructors in inductive logic programming for semantic parsing." In *European Conference on Machine Learning*, 466-477. Springer, Berlin, Heidelberg, 2001.

Tufte, Edward R. *Visual and statistical thinking: Displays of evidence for making decisions*. Vol. 12. Cheshire, CT: Graphics Press, 1997.

Utama, Prasetya, Nathaniel Weir, Fuat Basik, Carsten Binnig, Ugur Cetintemel, Benjamin Hättasch, Amir Ilkhechi, Shekar Ramaswamy, and Arif Usta. "An end-to-end neural natural language interface for databases." *arXiv preprint arXiv:1804.00401* (2018).

Velde, François R. "An Analysis of Revenues at the Comédie française, 1680-1793." In *Données, recettes & répertoire: La scène en ligne (1680-1793)*. MIT Press, 2020.

Walter, Maggie. "The voice of Indigenous data: Beyond the markers of disadvantage." *Griffith Review* 60 (2018): 256-263.

Walter, Maggie, and Michele Suina. "Indigenous data, indigenous methodologies and indigenous data sovereignty." *International Journal of Social Research Methodology* 22, no. 3 (2019): 233-243.

Yaghmazadeh, Navid, Yuepeng Wang, Isil Dillig, and Thomas Dillig. "SQLizer: query synthesis from natural language." *Proceedings of the ACM on Programming Languages* 1, no. OOPSLA (2017): 1-26.

Appendix A: Sample Configuration File

```
{
   "connection":{
      "database":"rcf_thesis",
      "host":"localhost",
      "user":"dbuser",
      "readonly_user":"clairon",
      "password":"MDzf*4rN6!SMG@Rk",
      "readonly_password":"phèdre",
      "schema":"public"
   },
   "parse_patterns":{
      "date":{
         "parts":[{"IS_DIGIT": true, "LENGTH":4}, {"IS_PUNCT": true}, {"TEXT":
{"REGEX":"^0?[1-9]|1[012]$"}}, {"IS_PUNCT": true}, {"TEXT":
{"REGEX":"^0?[1-9]|[12][0-9]|3[01]$"}}]
      },
      "season":{
         "parts":[
            {
               "IS_DIGIT":true,
               "LENGTH":4
            },
            {
               "TEXT":"-"
            },
            {
               "IS_DIGIT":true,
               "LENGTH":4
            }
         ],
         "valid":"lambda x:int(x[2])==int(x[0])+1"
      },
      "compound": {
        "parts": [
          {
             "IS_DIGIT":false
          },
          {
             "TEXT":"-"
          },
          {
             "IS_DIGIT":false
          }
        ]
      }
   },
```

```
"tables":{
   "auteurs":{
      "synonyms":[
         "écrivains",
         "dramaturges"
      ],
      "attributes":{
         "nom":{
            "synonyms":[
               "pseudonyme"
            ]
         }
      }
   },
   "pièces":{
      "attributes":{
         "création":{
            "synonyms":[
               "créer"
            ]
         },
         "divertissement":{
            "synonyms":[
               "danse",
               "musique",
               "machine"
            ]
         }
      }
   },
   "comédiens":{
      "synonyms":[
         "acteurs",
         "membres"
      ],
      "attributes":{
         "titre":{
            "synonyms":[
               "honorifique"
            ]
         }
      }
   },
   "lieux":{
      "synonyms":[
         "endroit",
         "théâtre",
         "scène"
      ]
   },
   "rôles":{
```

```
      "synonyms":[
         "personnages"
      ]
   },
   "registres_recettes":{
      "attributes":{
         "semainier":{
            "synonyms":[
               "signataire"
            ]
         },
         "recette":{
            "function":{
               "signature":"en_livres",
               "arguments":[
                  "livres",
                  "sols",
                  "deniers"
               ]
            },
            "synonyms":[
               "rapporter",
               "rapporte",
               "rentable"
            ]
         }
      }
   },
   "séances":{
      "synonyms":[
         "soirées"
      ]
   },
   "attributions":{
      "synonyms":[
         "écrit"
      ]
   },
   "interprétations":{
      "synonyms":[
         "interpréter",
         "interprète",
         "jouer",
         "joue"
      ]
   },
   "représentations":{
      "synonyms":[
         "répétitions",
         "jouer",
         "joue",
```

```
            "représenter",
            "représente",
            "répéter",
            "répète",
            "donne",
            "donner"
         ]
      },
      "ventes":{
         "attributes":{
            "prix":{
               "function":{
                  "signature":"en_livres",
                  "arguments":[
                     "prix_livres",
                     "prix_sols",
                     "prix_deniers"
                  ]
               }
            },
            "recette_totale":{
               "function":{
                  "signature":"en_livres",
                  "arguments":[
                     "recette_livres",
                     "recette_sols",
                     "recette_deniers"
                  ]
               }
            },
            "billets_vendus":{
               "synonyms":[
                  "places",
                  "spectateurs"
               ]
            }
         }
      }
   }
}
```