Interfacing data destinations and visualizations: A history of database literacy

Rahul Mukherjee
University of California, Santa Barbara, USA

Abstract
I look at journals and popular magazines on computers and information systems from the early 1970s through the early 1990s to see how they construct expertise about databases and address various publics with different “database literacy” levels. During this period, emerging database technologies such as relational database models and menu-driven interfaces made it possible for users to keep a distance from their data. Alongside such technical changes, socially constructed discourse distinguished “information” from “data” and experts (including computer programmers) emphasized that data was too enormous and unwieldy to be handled by common users and prescribed that such users should concentrate on working with information; that is, data processed by the database management systems (DBMSs). By tracing the socio-technical forces that created data–information categorizations and the dynamic interfacing role played by DBMSs, the article attempts to understand how we arrived at notions about where and how our data are stored.

Keywords
Data, database literacy, database management system, datagrammer, information, interface, programmer, relational databases

Introduction
During the late 1960s, the use of “database” as a term to designate large banks of data became prevalent in commercial, technical, and bureaucratic circles. The emergence of computer databases can be attributed to the coeval growth of electronic computing
technologies and the requirement for governmental and commercial institutions to store and retrieve large amounts of data related to consumers and citizens (Elmer, 2002). Since then, developments in databases and database management systems (DBMSs) have been inextricably tied to the growth of digital media (computers and Internet). The database has been hailed as a “genre of New Media” (Manovich, 2000), and database technologies have been considered instrumental in binding multimedia together in an electronic text (Elmer, 2002).

Lisa Gitelman (2006: 155) has asked, “Who knows from where this or that digital content comes from?” Bibliographic and textual approaches have tried to address pertinent questions such as: what does the body of the electronic text consist of, and what is the destination of the data that we “write on” and “read from” such a text (Gitelman, 2006: 96)? Building upon these questions, I ask: in the still nascent digital era, how have people been taught to think about digital data?

Inspired by Carolyn Marvin’s (1988) conceptualization of “technological literacy,” I use the term “database literacy” to understand the construction of expertise about databases and through print media literature and the process by which experts (programmers) and lay users of computers were differentiated based on their programming proficiency and knowledge of databases. I look at journals and popular magazines dealing with computers, information systems, and networks from the early 1970s through the early 1990s to see how they address different publics with different database literacy levels.

I also trace the emergence of relational database models and menu-driven interfaces used to query the database, both of which helped database management systems (DBMSs) develop properties of “data independence” and “data abstraction” (Liu, 2008; Simmons, 1984). The requirement that users know the exact pointer locations for the data they sought to find was no longer necessary because relational databases, unlike hierarchical databases, could implement data independence. Menu-driven interfaces enacted data abstraction by black-boxing program modules – users no longer needed to know how to write programs as long as they were aware of the effects or functions of software modules. The study of the discourses and practices of interfacing is important in understanding the nature of contemporary digital media because, as Lev Manovich (2000: 181) notes, “Creating a work in New Media can be understood as the construction of an interface to a database. In the simplest case, the interface provides a link to the underlying database.” DBMSs did the job of interfacing between users and users’ data destinations (or databases) and this article strives to uncover the different layers in which data is organized.

My discussion of layered data structures does not neatly map onto the hard media materialism approach epitomized by Friedrich Kittler’s insistence on studying voltage differences and logic gates or Wolfgang Ernst’s focus on signal processing and circuit board flows in his research on hardware materiality. However, it does share these scholars’ media archaeological approaches to media history, where digging into the temporal layers of a media object’s history is accompanied by the “excavation of the layered existence of technical media infrastructure” (Parikka, 2011: 65). Digging into the many levels of abstraction over which media technologies operate is important. In their everyday processual workings, DBMSs involve assemblages of users, data, information, code,
interfaces, modules, and representations at various levels of abstractions. Studying DBMSs is thus inevitably an archaeological endeavor: a media history of database technologies must, in some ways, also be a performance of database archaeology.

The story of database literacy is a narrative of two intertwined and mutually reinforcing developments from the 1970s through the 1990s. One occurred in the technical domain, where emerging database technologies – relational databases and menu-driven interfaces – made it possible for users to keep a distance from their data. Another took place in the socio-cultural domain, where socially constructed discourse distinguished information from data and thereby encouraged a distance between users and their data. Experts emphasized that data was too enormous and unwieldy to be handled by common users and prescribed that such users should concentrate on working with information; that is, data processed by the DBMSs.

Today, data produced by and about users as they provide consumer feedback, submit governmental forms, and interact through social media is increasingly used for marketing and surveillance. Why are most users ready to comply with or unwilling to pay much attention to such invasive practices? Can one of the reasons for such indifference be that users take for granted the role of DBMSs and do not question the distinction that has been made between data and information? Bowker and Star (1999) argue that any category or standard tends to privilege a particular point of view while disregarding another. In the history of databases, ease of data manipulation and the rendering of data in processed form have been privileged over knowing the destination of data or organizing precisely data’s structure. In tracing the socio-technical forces that championed the data–information distinction and the role played by database management systems from the early 1970s through the early 1990s, I want to recover the trajectory of the naturalization of databases. This endeavor should help us to begin answering the question: how did we get to where we are now with regard to thinking about where and how our data is stored?

The quest for efficient DBMSs was not merely about processing queries faster in real time and facilitating ways of finding data. Discovering how the DBMS could be made more user-friendly, how it could render data in the form of actionable information, and how it could enable computers to think more like human beings were all critical goals. These goals can be linked to the broader history of the ways in which a variety of social groups – cyberneticists, information scientists, businessmen, and policy-makers – involved themselves in defining information and “identifying information as the basis of a technological revolution that was creating a new age” (Kline, 2006: 516). Studying co-evolving database technologies and discourses about those technologies can contribute insights into why information as a word, an idea, a concept, and a commodity attained unprecedented currency in many realms, including the political, economic, academic, and social (Schiller, 2007), during the period discussed here.

### Practices of database literacy: research methodologies and theoretical frameworks

Carolyn Marvin (1988: 12) defines “technological literacy” as “a range of professional competencies that at their core valued skill in interpreting technical documents” and that
led to the formation of gradations of textual communities organized around “presumptively shared, but distinctively practiced, epistemology of texts and interpretive procedures that were sanctioned by certified authorities arranged in roughly concentric circles of expertise.” In studying the early social origins and development of electrical media, Marvin found that scientific and popular journals and magazines related to electrical technologies addressed a wide range of publics and communities with different competencies in interpreting scientific and technological literature.

Marvin adopts the term “technological literacy/electrical literacy” to write the history of electrical technologies. In a similar vein, I use the term “database literacy” to describe the practices of designated spokespersons and experts addressing a diverse array of publics about database models and technologies in computer and information systems journals, magazines, and newspapers. The publications I examined included IEEE (Institute of Electrical and Electronic Engineers) and ACM (Association for Computing Machinery) conference proceedings. Articles from these proceedings were essentially technical in content and exclusively written by members of the scientific community for members of that community – computer scientists, database architects, database designers, computer programmers, and information systems specialists, among others. I also studied trade and popular computer magazines and journals (containing varying technical details), including the On-Line Review: An International Journal of On-line Information Systems; BYTE: The Small Computer Journal (which carried special issues on “Database Management Systems” in 1981, “Databases” in 1984, and “Database Software” in 1988); Apple Orchard; Softalk; Creative Computing; and PC San Francisco (which offered independent reviews of IBM products and went on to become PC World and then PC International). This set of publications is by no means exhaustive for that time and thus the scope of my research is selective at best. A further limitation of this archive is that it is restricted to the United States, and thereby this history of database literacy (existing among many other such histories) is specific to that country.

The periodicals I examined included sections discussing the technicalities of different database systems and architectures and ways to improve programming query languages. Many sections were devoted to reviewing computer products for non-programmers, conversations about present trends and future goals of computer (database) technologies, and discussions about the uses of personal computers, online databanks, and stand-alone databases. Topics included the utility of storage spaces in the fields of medicine, science, education, design, and finance, as well as in domestic space (see Poor, 1985). Scattered among these articles were advertisements for new computer components and equipment, letters to the editor, book reviews, database catalogues, etc. The so-called experts in these journals were not just scientists and specialists, but also business consultants, science writers, and sociologists who tinkered with computers in their geekier moments. Sometimes they even included ordinary computer users who had used a certain database for a while and wanted to talk about how satisfied or frustrated they were with it (see the articles by Brent (1981) and O’Rourke (1984)).

The publics addressed in the periodicals I studied were differentiated by their ability to comprehend knowledge about computers, and experts employed different tones and made different suggestions in speaking to each of them. Experts established themselves as such by precisely defining key terms and concepts and adopting a pedagogical tone
when addressing neophyte users, as becomes clear in the following excerpt from one of the first articles in *On-Line Review* in 1977. The article lays out similarities and distinctions among the terms “data base,” “data base system,” “search system,” and “information retrieval system”:

* A word of explanation is in order. This terminological distinction has to do with an all important difference between a data base and a data base system. A data base is simply any computer readable collection of data... data base system is the component which retrieves and analyzes the data... and this can be called a “search system,” or an “information retrieval system” also (emphasis added) (Luedke, 1977: 210).

In the same vein, Jonathan Robie (1988), in *BYTE*’s special issue “Database Software, Managing Megabytes”, explains query optimization as a solution for accessing data from large databases. He also signals readers to a separate textbox defining “JOIN” operations and database terminology without which, he states, they would not understand the rest of the essay. In Robie’s and Luedke’s articles, databases are presented as “definitional puzzles” (Peters, 2009: 19). New media, Benjamin Peters (2009: 18) remarks, “are media we do not yet know how to talk about.” It is this kind of uncertainty about the nature, uses, and purposes of new media technologies such as databases, and the terms by which to talk about them, that makes powerful the voices of experts, early adopters, and inventors in the pages of these magazines and journals and makes the construction of database literacy a crucial object of study.

In deploying the term “literacy” and its association with expertise, Marvin emphasizes the interpretation of “technical documents” – being literate most importantly involved being able to fully comprehend the technical nature of texts. I would like to broaden the notion of literacy beyond the interpretation of and access to technology and incorporate the ability to use and manage it. My conceptualization of database literacy helps us to understand how practices of inclusion and exclusion were implemented based on considerations of epistemological credibility. Computer programmers were considered insiders and part of the elite computer science community. They were addressed in a different way from non-programmers, who were perceived as incapable of reading and writing the programming codes/languages written in some of the journals. Experts reviewing database-related computer products positioned themselves as and alongside programmers while at the same time explaining to non-programmers how they could use the database management system efficiently without having to learn programming.

A glimpse of such positioning is evident in a *New York Times* article from 1986, where Erik Sandberg-Diment (1986), an expert reviewing the database manager Interlace, opines:

* Probably the first question the practiced computerist asks when faced with yet another database manager has to do with how good its programming language is. The answer to that question in this case is that Interlace doesn’t have any. The lack, furthermore, is all to the good.

As a “practiced computerist,” Sandberg-Diment asks lay users not to bother with programming language.
Databases and digital media: early politico-economic contexts

Recent scholarship on database technologies notes that relational database models, owing to their de-centered architecture, have fundamentally altered traditional notions of narrative (Hayles, 2005). Stressing the instrumental role played by databases in the creation of hypertexts, digital media scholars have referenced Codd’s foundational 1970 article written on the relational model of the database. However, database history and its political and economic contexts dates back to earlier than 1970, and that history has ramifications for how relational models of databases came to be what they are today.

Greg Elmer (2002: 90–91) locates the origin of databases in Herman Hollerith’s punch-card and tabulating machine, which were used in the 1890 US census and, through electric computation, quickened the process of recording and counting census data. The government, pressed to collect and maintain larger and more complex demographic data, worked in close cooperation with Industrial Business Machine (IBM) as the company started researching faster computational technologies. American Airlines also influenced IBM’s research efforts in its search for efficient ways to schedule customer requests within its centralized reservation system, which required updating and retrieving large amounts of information (Elmer, 2002).

Until the 1960s, IBM enjoyed a monopoly over administratively oriented computers, which performed electronic data processing and were seen as evolving from, and hence replacing, the use of punch cards in mechanical tabulations. Thomas Haigh (2001) charts the history of how corporate systems managers during the 1950s and 1960s redefined computers as “managerial information systems” rather than mere extensions of punch-card machines. For example, American Airlines’ SABRE (enabled by IBM) allowed travel agents to query flight availability and make reservations; system managers promoted SABRE in corporate computing settings, arguing that “real-time access to business data” was required for making efficient corporate management decisions (Haigh, 2001: 40).

Such commercial and industrial contexts shaped E.F. Codd’s work and his 1970 paper, written while he was employed at IBM. Codd constructed a relational model of databases in response to the need to accommodate queries by a large number of people accessing different parts of a shared database – a situation in which, according to Codd, users were not necessarily interested in knowing the exact pointer locations of their data. In Codd’s estimation, users were concerned with processed data, which could be visualized in tabular form. The title of Codd’s (1970) paper, “A Relational Model of Data for Large Shared Data Banks,” itself makes apparent that he was preoccupied with managing very large shared data banks. As he clarifies, these data banks were accessed by multiple users for different purposes. Below, I elaborate on his development of the relational model, beginning with a brief discussion of databases and the different models of organizing data.

Relational databases and data independence

A database should not be conceived of as just a random collection of data, but as a “structured collection of data” from which data can be easily and efficiently searched for and
retrieved (Manovich, 2000: 177). A database management system works toward these operations by structuring the database’s data. In a hierarchical database, a record contains groups of similar objects organized in a cascading hierarchy (Kawan, n.d.: 2). In a relational database, data are parsed in tables consisting of rows and columns, with each record (in a row) consisting of a set of inter-related entities (in columns) (Codd, 1990; Kroenke and Auer, 2007).

In their study on the implementation of database systems, Rob and Coronel (2002: 24–33) point out that in a hierarchical database, one searches within the tree-like structure by using a pointer to traverse branches. Commenting on how such traversing pointers needed to know exactly where data is organized in hard storage, Alan Liu (2008: 255) notes that “fundamentally, such lack of logical independence was symptomatic of a lack of physical independence” (emphasis added). Indeed, in his paper, Codd too was interested in providing such a logical independence, which he termed “data independence” in a relational model of the database. For Codd (1970: 377), “data independence” meant the “independence of application programs and terminal activities from growth in data types and changes in data representation.” Therefore, according to Codd (1970), moving towards the relational model from tree or hierarchical structured database models would help database users to avoid the hassle created when the internal representation of the data has changed; that is, when the data has been reorganized in the database without the user’s knowledge.4

In his relational model, Codd (1970) envisioned putting data into tables consisting of entities inter-related with each other and with other tables. Since varied users would be looking for different kinds of information, Codd reasoned that each user did not have to know everything about all tables where data was located, but only about those tables that were relevant to his/her work. Through a DBMS, users could query for a view (or visualization) of those specific tables and manipulate the data they were able to retrieve from them. Since the tables that users could not see were referentially related through common entities, users could make necessary changes, and data would automatically be inserted into those other tables as well.

The way in which tables represented data was independent of its physical location and ordering; this is what Codd (1970) called “data independence.” A user could change data representations through the DBMS without having to know the exact pointer location or access path to the record’s specific entry, as earlier hierarchical models had required.5 The rise of relational databases thus reduced ordinary users’ concern with the intricate details of data storage and dependence upon software programmers.

**Distinctions: data vis-à-vis information and programmer vis-à-vis non-programmer**

Joel Neely and Steve Stewart’s 1981 piece “Fundamentals of Relational Data Organization,” published in the November 1981 issue of *BYTE*, and Rodney O’Rourke’s 1984 article “Foolproof, flexible database management” in the March 1984 issue of *Creative Computing*, echoed Codd’s emphasis on ease of access and retrieval as the benchmarks of effective database design. Such an emphasis prioritized generating precise reports quickly, using certain portions of the data without having to take into account
where the data was stored. This focus resonates with Gitelman’s concern about the lack of questions regarding the final address of digital inscription. In his 1984 article O’Rourke, in an autobiographical vein, eulogizes about how useful the “Data Factory” database has been to his work as a sales representative and is candid about admitting that he is not a programmer. He writes:

I am not a programmer, and I don’t want to be a computer expert. My use for a computer is straightforward: I want it to assist me in the performance of my job, make me more efficient, and help me serve my customers… Data Factory allows me to produce reports customized to my specific requirements (O’Rourke, 1984: 136).

This discourse (propagated by both programmers and non-programmers) about not being a programmer and being content with getting customized reports seems to have been prevalent during that time – the reason cited was that one had a lot of data to sort through and deal with in a very short time, so views of only relevant table data were required. Nobody wanted to deal with data, because either it was the job of the programmer or it was best consigned to the DBMS (which did the interfacing) – after all, that was why users needed to buy it. This quote from an article in the second volume of the *Antic* magazine in 1983 paints a scenario of people hard pressed by data overload and in severe need of DBMSs:

People display a seemingly insatiable desire to keep track of information. My wife is organizing data on 600 Girl Scouts for their summer program. I have a file of participants for a research study. My neighbor’s job requires tracking the court appearances of various miscreants. All these jobs cry out for efficient ways to store and use information, and in answer we have the computerized Data-Base Management System (DBMS) (Harms, 1983: 43).

By the early eighties, DBMSs no longer exclusively catered to big businesses. They had utility for many other users who needed to manage information in their respective professions. The May 1984 issue of BYTE devoted itself to “professional computing,” suggesting ways of integrating microcomputers into professional work environments, including legal and medical practices. Along with word processing capabilities and communications, DBMSs were often recommended in these articles to manage client or patient records, retrieve disease or litigation literature, and keep track of appointment schedules (see Wilkins, 1984; Zucconi, 1984). The intended ubiquity of personal computers with database packages to take care of one’s daily business and activities was famously summed up by Apple Inc.’s then Vice President, Software, John Couch, who stated: “The microcomputer will be to the eighties what the calculator was to the seventies” (Tommervik, 1980: 7). However, if the naturalization of the idea of an “interface” required more pretext, one was provided by distinguishing between “information” and “data.” Consider this 1984 review of the newly released Pick Operating System, with an inbuilt database system, by Rick Cook and John Brandon (1984: 177):

One needs to consider all the times you confuse data with information and end up processing data instead of managing information… Data includes such things as invoice numbers and customer addresses, price list and style codes. Information answers questions: Am I making or
losing money. Have I got enough material to produce what I need for next month… computers should help you sort out relevant data… but on most computer systems, machine needs (and not user needs) predominate. Instead of helping you squeeze information out of the real world, the computer traps you in an unreal world conditioned on its structure (emphasis added).

This distinction between the two terms signifies an unwillingness to deal with the purported messiness of data, such as invoice numbers and price lists, and the computations it involves in favor of “information,” which is equated with computer-processed data that enables quick decision-making about profits (“Am I making or losing money”) and investments (“Have I got enough material to produce what I need for next month”). Something more is connoted when it is noted that some computers trap one into the unreal world of machine structures. Dealing with data or thinking about data (and, to borrow from Gitelman, thinking about the nature of electronic texts) would lead to being trapped into understanding how the computer works; through the use of Pick Operating System, one is encouraged to avoid that.

If we shift to the macro-level from the pages of computer magazines, we will find that the distinction between information and data took place because political and economic elites of the time attached great importance to information. During the period beginning in the 1970s, when an unprecedented need for database technologies arose, the US, according to Schiller (2007: 36), witnessed “accelerated information commodification.” With the global share of US manufacturing industries sharply declining, a need was felt to provide importance to the “information industry” as a strategic maneuver to develop new profit sites, and “technological innovations by the military-industrial complex during and after the World War II were incorporated into a succession of systems for capturing, storing, sorting, sharing, and measuring information in real time” (Schiller, 2007: 39).

Analyzed within this broader context, the database articles I discuss here can be read as valorizing information by defining it as something that needs to be processed from “raw data,”6 and at the same time enhancing the credibility and importance of DBMSs because these systems could re-present valuable information in comprehensible and intelligible forms so that users could make decisions based on them in real time. In professional settings, this shift in the perception of computers from efficient data crunchers to intelligent machines aiding decision-making could be seen as shifting the status of computers from mere labor-saving machines to “serving the firm’s senior decision makers” and thereby becoming an “indispensable part of the core management itself”: an attempt for such a shift was made earlier by systems managers from 1950–1968 (Haigh, 2001: 41).

Ronald Kline (2006: 515–516) reminds us that “information” emerged as a keyword to which various social groups attached different meanings during this time. For some, information was semantically critical; for others, such as physical scientists and communication engineers, information continued to be a “mathematically defined, non-semantic quantity related to ‘entropy’.” It is important to contextualize the discursive classification of data and information during the 1970s and 1980s within a wider movement in the fields of information science and management science (including people involved in business and policy-making). Based on such contextualization, one could define information “semantically as the middle term between ‘data’ and ‘knowledge’ in
a hierarchy of cognition” (Kline, 2006: 516): if information was processed data, knowledge was processed information. Michael Buckland offers another formulation of this triad of data–information–knowledge in which the middle term, information, can take both the forms of knowledge and data. According to him, “information-as-knowledge” is subjective and conceptual and thus intangible, but still needs to be represented in some physical way as “information-as-thing”, such as data or signals, which are tangible (Buckland, 1991: 351). Buckland’s alternative formulation suggests that even as the discursive classification of data, information, and knowledge has become stabilized since the 1970s, semantic quirks associated with it continue to be questioned and debated.

To get back to magazines, Cook and Brandon’s review of Pick Operating System in 1984 gestured towards the data–information divide. The writers praised computers’ capabilities for processing data but pointed out that users would need to understand a computer’s structure (computerized data storage) to work on that data. According to the reviewers, Pick Operating System, an information-oriented operating system with “built-in relational database” and “multi-user capabilities” took care of the above-mentioned problem: “The structure of Pick focuses your attention on information rather than workings of the computer” (Cook and Brandon, 1984: 177). Here it is almost acknowledged that an interface in the form of the built-in database system focuses the user’s attention on information and prevents the user from getting distracted by the workings of the computer – put another way, the interface effectively hides the workings of the computer.

This review, like many others, can be seen as reaching for an audience larger than just academics or geeks. In doing so, its authors instructed readers in a certain way about perceiving computers and interacting with computer data. In fact, one can see the ads and reviews I discuss here as creating a cultural perception of human–computer interaction. If the review and ad for Pick Operating System attempted to hide the computer’s workings, an advertisement for the database system Sequitur in the November 1981 BYTE magazine sought to make the computer think like a human (Sequitur, 1981).

The ad’s punchline reads “A Database System that thinks like you do” and shows a sketch of a man with a thought bubble reading “$1/rong/wrong/p” – signifying that he is thinking in computer program language – while the computer has a thought bubble saying, “Couldn’t we be a little friendlier” (refer to Figure 1). The computer’s friendliness and human-ness connoted its increased (artificial) intelligence.

A debate on whether computers were not only fast and reliable machines, but also intelligent machines, intensified during this period in circles wider than those of cyberneticists. Indeed, changes in database technologies took place concurrently with studies in artificial intelligence (AI). Before the Sequitur ad appeared, Steven Roberts wrote an article in AI’s September issue of the same year suggesting that computers had gained greater data-processing abilities but would become “intelligence amplifiers” only when communication with them was not formal and restricted to programming languages, when their “internal world” would not be alien to him, and when they would yield “information, not just data” (Roberts, 1981: 178). Sequitur’s DBMS did not fulfill all the criteria laid down by Roberts for computers to be intelligent, especially the one about the incompatibility between the internal worlds of humans and computers. However, the Sequitur ad certainly seems to be affectively catalyzing the desire of humans to interact with machines as if those machines were intelligent humans.
In the ad, due to the intervention of the Sequitur DBMS as an interface or mediator, the computer has apparently been humanized and is suggesting to the human that he need not become computerized. Under the subheading “It [Sequitur] speaks your language,” it is noted that the user no longer has to write a code like “INSERT INTO EMP (EMPNO, ENAME, JOB, SAL): (7989, “CARTER”, “SALES MAN”, 1500);” and instead just needs to type in “ADD” and fill in the required fields. However, what it does not say is that providing a shortcut for adding data to tables also results in being less informed about the data characteristics of the EMP or employee table. This shortage of information would prevent the user from performing operations more complex than “INSERT” or “ADD.”

Data interfacing as a design strategy enabled by DBMSs thus involves “data abstraction,” which D.L. Paranas (1972) calls a form of information hiding where the behavior of software modules is specified in terms of external effects. Put another way, one does not have to know the code inside a module or the internal processes which make the module behave the way it does; it is enough to know the external function the module performs when called or invoked. In the Sequitur ad, the behavior of the “INSERT SQL”
query statement, which the database management system now performs instead of a programmer (when the user writes “ADD” and fills in the record), is a data abstraction in practice. The imperative of programming to make data interfacing easier for lay computer users leads to the entry of the “rule of modularity” (McPherson, 2010) into the database logic, and thus into the culture of human–computer interaction.

Such a “rule of modularity” also accords a special role to the software programmer. The programmer can have knowledge at different levels of abstraction – from viewing tables to understanding the relations between entities and from entering data to finally knowing its actual physical (storage) location. Lay users only get to view the tabular reports. The tabular report is an external view or outer visualization that presents data to users at the external layer of a database. Unlike lay users, programmers’ knowledge of the relation between entities comes from their understanding of the conceptual schema of the database that operates at the conceptual layer and is interfaced with both the external layer at the top and the physical layer at the bottom where the actual storage of data takes place (see Figure 2). This diagrammatic layout of the DBMS, consisting of multiple layers and interfaces between the layers, provides an idea of how DBMSs actually work, and digging through these layers – that is, performing database archaeology – is key to pointing out how the data-versus-information classification continues to operate between external and conceptual layers while obfuscating any attempts to discuss the tasks undertaken at the conceptual and physical layers. Such a media archaeological approach (Parikka, 2011) then helps to show how the discursive classification leads to infrastructural invisibility (Bowker and Star, 1999), for it focuses on some particular layers and interfaces and not others. What is fascinating about the periodicals I discuss here is that, because they were fostering communication not only between experts and lay users but also among experts, discussions of techniques of data abstraction that would lead to infrastructural invisibility also took place.

While magazines such as BYTE continued to run ads and reviews about the utility of DBMSs for users who did not know how to program, there were also sections for computer scientists and programming enthusiasts to encourage them to develop programs that could simulate data abstraction in a better way. An article by Gary F. Simmons (1984: 130) notes that data abstraction is a “strategy of ignoring certain details about the original problem so as to transform it into a simpler and more general one.” One can therefore see the parallels in the distinctions made between treating digital data as complex data vis-à-vis general information depending on whether one is a programmer or non-programmer. If a DBMS was menu-driven, a layer of data abstraction got added when compared to a command-driven DBMS since one now did not need to know how to write a SQL statement, but could just invoke one through inputs on the menu.

The rift between menu-driven non-programming modes and only-for-programmers command-driven database management systems continued. While menu-driven DBMSs were praised for their ease of use, command-driven ones were said to provide sophisticated manipulation of data. The then (1984) recently launched menu-driven DBMS “DataEase” claimed to provide both ease and sophistication. An ad comparing DataEase with the command-driven DBMS “dBASE” (popular among programmers) declared: “‘DataEase’ has the power of dBASE without dBOTHER” (Jacobson, 1984: 289), where “without dBOTHER” indicated ease of use and “power” indicated increased sophistication in
manipulating data. However, programmer Bill Jacobson, writing in the Software Review section of a 1984 *BYTE* issue, created an insider (programmer) versus outsider (non-programmer) distinction when he compared mode-driven “DataEase” with command-driven “Condor” and “dBASE II” DBMSs and concluded that DataEase was easy to use but did not quite match up to the sophistication of dBASEII or Condor. In his evaluation, Jacobson gave high points to DataEase for its interactive capabilities and report generation features and praised dBASE II for its versatility, remarking that “for someone who wants the ultimate in

**Figure 2.** Three layers of a DBMS.
Source: Bergholt et al., 1998.
DBMS flexibility, has or wishes to develop good programming skills…dBASE may be a good choice” (Jacobson, 1984: 302). dBASE, by virtue of its having a built-in programming language, provided a vast array of special commands, including specifying the particular records to be listed and the capability to modify the structure of the database.

In sync with these dichotomous categorizations of data versus information, programmers versus non-programmers, and command-driven versus menu-driven DBMSs was John Couch’s (1981/1982: 19) proposition for machines to have “specification languages” that would allow people to solve problems “by specifying what they wanted in terms of inputs, outputs, relationships etc.” Couch (1981/1982) proposed the term “datagrammer” for those who would work only with specification languages without having to program in algorithmic languages. He explained the distinction between programmer and datagrammer with a diagram (refer to Figure 3). He also suggested a temporal demarcation based on the growth of technology by noting that before 1980, only programmers could work on data, but after 1980, data would have to evolve into “data defined via specification language” (data amenable to menu-driven inputs and readily reproducible reports, in other words), so that datagrammers could work on it.

The idea of a database system acting as an interface and the dual functions of an interface to make something available and at the same time to abstract or hide something also emerged because some of the governmental and military databases were considered private and needed to be protected. A database management software needed to guarantee data privacy and data security, which would prevent data loss in the event of several users accessing the same database.

Reading James L. Luedke (1977: 207), it becomes apparent that availability and accessibility were important considerations while thinking of databases:

By accessibility of data bases is meant the mode of retrieving the data from them, namely interactively, or in batch, and whether or not, they are remotely accessible over a computer network. By availability is meant the intended user-audience: public, restricted, or in-house…
Many of in-house databases have this status by virtue of a lack of awareness on the part of potential user-communities, rather than by outright fiat.

Luedke, an information systems researcher and consultant to a number of library and information sciences organizations, lays out the concepts of accessibility and availability and provides expert opinion that user-communities exist that are unaware of the online databases they can use. Many articles (Curtis, 1977; Monsen, 1977; Nash, 1982) in On-Line Review concerned themselves with helping researchers to efficiently search specialized bibliographical and numerical databases pertaining to specific academic disciplines. Here again, an interface was necessary to give access only to selected researchers who had permission to use certain restricted databases – in Luedke’s words, the “intended user-audience.”

Conclusion

Database literacy is helpful in understanding how discourse about database technologies circulated by experts and non-experts perpetuated classificatory regimes of programmers and non-programmers and of data and information. DBMSs have acted as interfaces, helping users visualize relevant data while at the same time hiding data destinations and the workings of the computer. Alexander Galloway (2012: 33) proposes considering interfaces not as surfaces but effects: “…an interface is not a thing, an interface is always an effect. It is always a process or a translation.” In studying DBMSs, I have focused on the act of interfacing, which entails tracing the dynamic interface processes of translating data and information across varied levels and in manifold visualizations.

boyd and Crawford (2012) write that the era of “Big Data” has begun and getting access to massive amounts of data is becoming crucial for professionals and academicians ranging from computer scientists and bioinformaticists to sociologists and political scientists. They express concern that those who get to analyze “Big Data” are a small group of experts with computational skills who then also determine the rules governing the usage of data. This kind of institutional inequality in usage and participation is similar to the maintenance of the programmer and non-programmer divide discussed in this paper.

If the 1970s saw the emergence of relational databases and the 1980s their eventual dominance, the early 1990s roughly mark the point where object-oriented models, especially for applications involving complicated relationships between data objects, begin to challenge relational models. With the development of object-oriented languages, object-oriented databases emerged. BYTE, which had already devoted issues to object-oriented programming in 1986 and 1989, carried an article by Daniel Rasmus in 1992 titled “Relating to Objects.” Rasmus argued that object-oriented database management systems (OODBMSs) were better able to handle complex data structures and data types than two-dimensional arrays of relational database management systems (RDBMSs) and were thereby more suitable for supporting applications like CAD and multimedia. Rasmus does however acknowledge that RDBMSSs cannot be replaced. OODBMSs carry over the qualities of data independence (including structural independence) and data abstraction from relational databases, although they are implemented differently. Currently, relational, object-oriented, and object-relational databases co-exist and compete with each other.
In 2000, experts began pointing out scalability issues while implementing relational database models for distributed systems. In relational databases, each action performed needs to pass consistency checks. With a wide set of connected databases across space, carrying out consistency checks each time takes up a lot of time and may not be the most efficient process. Some other database models and algorithms, based on NoSQL (Not only SQL), have through the implementation of CAP (Consistency, Availability, Partition Tolerance) traded consistency for availability; that is, for them, meeting a query request quickly is more important than keeping it pending until one completes consistency checks (Anderson et al., 2010: 12–13). Recent articles in Wired suggest that experts at Amazon, Google, and Facebook believe NoSQL to be a future-oriented solution towards building databases to store the “unstructured information” that comes spilling from the Internet (Garling, 2012).

Although I have restricted my historical enquiry to end in the early 1990s, I must add that computer magazines have significant online presence now, experts continue to provide advice, and interaction with these experts often takes place in online discussion forums and blogs. Experts emphasize the changing nature of data storage in the contemporary data ecology of Big Data and cloud computing. For example, John Cook’s (2012) personal blog post republished in DZone explains that relational databases were meant for small, expensive storage media, but now storing data is cheap and applications no longer share data through tables but via APIs, and Jake Robinson (2013) advises enterprises and application owners through Wired magazine’s community blogs on the best practices of scaling their relational databases for the cloud.

It is crucial to keep DBMSs’ ability to provide interfaced visualizations in mind when digital media users think of their own data, their friends’ data, their privacy settings, and their security on social networking sites, which are, among other things, networked databases. Admittedly, one cannot possibly make sense of so much data, and interfaces do render relevant data visualizable. Yet one must ask: based on what parameters and protocols do they tend to do so, and what kinds of data abstractions result from that? Users should be cognizant of these questions as they willingly or unwillingly risk the consequences of the unknowability of their data’s destinations and visualizations. In conclusion, I would like to argue for a critical pedagogy of databases that attempts to acknowledge and decipher how, in facilitating seamless information visualization, DBMS interfacing processes require users to cede control over their own data.

Acknowledgements
I am indebted to Rita Raley, Nicole Starosielski, Jennifer Holt, Sarah Harris, and Athena Tan for their careful reading and insightful suggestions. Discussions at the ICA Virtual conference in 2011 and the Skip Intro meetings (organized by Utrecht University’s New Media Studies Program), where drafts of this paper were presented, were extremely beneficial. The editors and two anonymous reviewers at New Media & Society helped to enormously improve the paper through their critical incisiveness, rigorous feedback, and kind encouragement.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.
Notes

1. Roger Clarke (1988: 499) wrote about surveillance strategies carried out through collection of personal data in 1988 and defined the term “dataveillance” as “the systematic use of personal data systems in the investigation or monitoring of the actions or communications of one or more persons.”

2. It must be noted that James Beniger (1986) convincingly shifts the origins of the computing technologies and data-processing systems from the twentieth to the nineteenth century, when American society was rapidly industrializing, leading to a “crisis of control” over the manufacturing and distribution of products and services. A whole information apparatus developed which could regain control by regulating the movement of means of transportation of goods and people through careful measurement and co-ordination. This theme of loss of control – albeit for the slightly inflected reason of having to deal with too much information, which Ronald Kline (2006: 517) dubs the “perceived crisis of an information explosion” – gave rise to the need for databases that would help to keep track of it.

3. The close relationship between databases and airline companies during this time is also evident from the many ads put up by these companies looking for database managers capable of operating large database systems. For example, an ad in the New York Times looked for executive programmers who could work with large database systems as part of the aircraft ground systems group (The New York Times, 1972).

4. Others, such as S.K. Singh, suggest that hierarchical databases do provide data independence by isolating programs from storage details, and yet such data independence is compromised by lack of “structural independence.” Singh (2009: 76) clarifies that in the hierarchical models, changes in data characteristics do not abruptly transform application programs but “modifications or changes to the physical structure can lead to the problems with application programs, which will also have to be modified.”

5. Automatic updating has to be done to maintain referential integrity, which means that data relations have to be made consistent across updates. As far as the common entity for the two tables is concerned, for one table it is called the “primary key” and for the other it is referred to as the “foreign key.” See Codd (1970).

6. Even though scholars such as Geoffrey Bowker and Lisa Gitelman have argued that “data are always already cooked” and “the imagination of data entails an interpretive base” (Gitelman and Jackson, 2013: 2–3), the tendency to attach “rawness” as a quality of “data” continues to be ubiquitous in our contemporary times.

7. Studies of the hailing and addressing of diverse publics and potential consumers through advertisements are crucial in comprehending the history of database literacy, because these ads aid in comprehending more fully the role of discourse in shaping the social imaginaries of database technologies.

References


Garling C (2012) Amazon goes back to the future with NoSQL Database. *WIRED*. Available at: http://www.wired.com/wiredenterprise/2012/01/amazon-dynamodb/


**Author biography**

Rahul Mukherjee is a PhD candidate in the Department of Film and Media Studies at the University of California, Santa Barbara. His research interests include translocality, critical/cultural theory, and development discourses. Rahul’s dissertation explores mediated technoscience debates in contemporary India.