This paper was presented at PICKLAB'90
Victoria University of Technology, Melbourne, 17-19 July 1990

The Impact of Artificial Intelligence on Database Development

Jacob L. Cybulski
Amdahl Australian Intelligent Tools Program
Department of Computer Science and Computer Engineering
La Trobe University
Bundoora, Vic 3083, Australia
Phone: +613 479 1270, Fax: +613 470 4915

Abstract

This paper discusses the process of integration between Artificial Intelligence and Database Management Systems, two rapidly developing fields of Computer Science. The paper emphasises the issues of database improving representation, reasoning and interface capabilities while still preserving their capacity, efficiency and distributive powers.

1 AI Confusion

Despite the cost and all the reason, throughout the history of Computer Science, two major streams of Information Technology, i.e. Data Base Management Systems (DMBS - Brodie 1986) and Knowledge Based Management Systems (KBMS - Szolovits 1986), developed quite independently of each other and are said to aim at two distinct and very distant goals. The main difference between the two fields is the subject of their study, DBMS directs its attention at *data* and its structure whereas KBMS claims to have studied *knowledge*. Then they differ quite dramatically in their approach to the subject in focus, while DBMS researchers perfected the efficiency of storage, retrieval and sharing of voluminous data, KBMS experts emphasised the representation, manipulation of and reasoning about the information meaning. But then again the two fields are quite similar in their attempt to model and store structured, factual information about the world and its inhabitants.

To have a full understanding of our further elaboration, let's start with the characterisation of and distinction between five abstract notions :- datum, information, inference, knowledge and wisdom. As it may easily be shown by following the infinitely recursive chain of explanations in any dictionary, a clear definition of our concepts is not easy; hence, let us simplify the issue and give a clear-cut description of the five terms, description that would most certainly abhor any practicing epistemologist. A symbol or a value representing an element of reality will be called *datum*, structured and non-redundant data that are free of noise will be referred to as *information*, *inference* is a reasoning procedure interpreting information, thus turning it into *knowledge*, meaningful information. Some would also define *wisdom* as the highest level of knowledge, expanded by experience and ability of its critical use and application. (Cybulski 1987)

A *database*, in this context, is simply a collection of predominantly uniformly structured data (information), and a *knowledge base* is merely its knowledge counterpart. In a knowledge base, however, knowledge items usually cross-reference, negate and re-interpret each other, thus adding significant complexity into the underlying information system. The responsibility for interpretation of data, information and knowledge structures may lay entirely with the information system user or his/her program, but it may also lay with the information system itself, e.g. via a data dictionary and its schema or a knowledge base inference rules interpretable by its inference engine, thus making it a dynamic and a flexible product.

data	10897 SAVINGS Suspect	OK 80999 2451	ACCESS 10979 11879	-500 0000 OK	12987 ACCESS 1477
information	*ATYPE* SAVINGS ACCESS ACCESS	*ACC#* 10897 11879 12987	*ABAL* 10979 -500 80999	*APIN* 0000 1477 2451	*AUDIT* OK Suspect OK
knowledge	*ATYPE* t	*ACC#* x	*ABAL* y	*APIN* z	*AUDIT* ABAL < 0 *or* ACC# < 10000 *or* APIN > 9999 *then* Suspect *else* OK
wisdom	*An account with a high balance and high volume of small transactions is suspect.*				

Figure 1 - Data, information, knowledge and wisdom

An account database, a part of a hypothetical banking system, does illustrate our terminology (Figure 1). According to the previous definitions a set of unstructured facts is our data (SAVING, 10897, -500, OK, etc.), once collected into a structure of identifiable data fields they constitute processable information (data records, e.g. account type ATYPE, its number ACC#, balance ABAL, pin number APIN, and an auditing flag AUDIT), then addition of interpretation, no matter how minuscule, converts it into an element of knowledge (*AUDIT* field - Suspect and OK), wisdom deals with a much wider context and is obviously based on personal experience and judgement (e.g. detection of suspect transactions).

2 AI Potential

For years Artificial Intelligence (AI) used to be compared to the black magic of Computer Science, a secret society of mad scientists working on the Philosopher's Stone transmuting a computer idiot savant into an intelligent and eloquent homunculus. The times have changed, now the governments fund Artificial Intelligence research direct, base some of their high-tech policies on the AI promise, software and hardware manufacturers slowly adopt the fundamental AI concepts, AI products reach the commercial market.

The information management technology was possibly the first of the computer fields that benefitted from the global AI trend. An object-oriented and semantic data models became accepted as new database standards (e.g. GEMSTONE - Maier et al. 1986, TAXIS - Levesque and Mylopoulos 1979, CG - Sowa 1984), logic databases came out of the research cupboard (PROBE - Dayal and Smith 1986, NAIL! - Morris, Ullman and Van Gelder 1986, POSTGRES - Stonebreaker and Rowe 1986), decision support tools nowadays supplement data management (e.g. commercially available expert systems - ART, NEXPERT, LASER, KEE), natural language interfaces extend the capability of query languages (e.g. INTELLECT - Harris 1984), overall, the emphasis shifts from the domain of data and information processing right into the knowledge manipulation frontier. The impact of knowledge technology on databases could be noted in three major areas, namely knowledge representation, reasoning methods and user interfaces.

Representation. One of the greatest pressures put on current database providers is to improve their product modeling capabilities. The ever popular, commercially available data models, i.e. relational, hierarchical and network, are well equipped for inventory types of tasks suited for everyday business needs. Database technology, however, rapidly moves into new and complex application environments where the sophistication of problems being modeled either exceeds the power and flexibility of currently used tools and techniques or demands prohibitive amounts of design and implementation effort.

For instance, in the area of defense, the battlefield management systems need to represent military formations, procedures and regulations, tactics and strategies, actions and responses, enemy profiles, and weaponry systems. In engineering applications, it may be required to model design, production and maintenance tasks, encode machine part assemblies, describe chemical and physical processes, maintain sensory and control systems. Hospitals also move away from simple stock and registry systems, there is more significance is placed on intelligent decision support, illness diagnosis, treatment planning, medicine prescription and administration, patient and staff management, etc. These and many other enterprises, e.g. building and construction industries, government administration, law enforcement, education and banking to name just a few, have a common information modeling problem. They all utilise and manipulate a very rich collection of data types, structures and inference procedures, the volume of which is comparable to that of the data itself. In these applications, database highly prized traits of uniformity and stability are of detriment to their expressive power and overall efficiency. Thus, introduction of new modeling tools and techniques becomes critical.

The new tools and techniques are available now. The semantic and object-oriented data models (Hull and King 1987), having their roots in AI semantic networks and frames (Woods 1975), allow describing knowledge domain with the use of entity-relationship like diagrams (ER - Chen 1976), they permit describing elaborate relationships between application entities, structuring information into types hierarchies, handling value and structure inheritance, defining new concepts by context, etc. Let's use a simple(-istic!) banking example again (Figure 2).

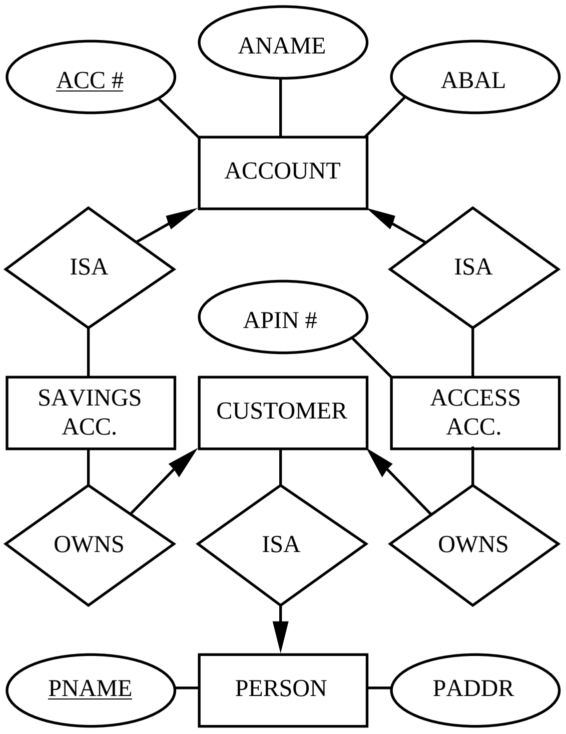

Figure 2 - Semantic data model

A hypothetical banking system must be capable of dealing with bank ACCOUNTs and PERSONs using them (entities - rectangles). Any ACCOUNT will have at least three attributes (ovals) :- account number (ACC# - the key), account name (ANAME) and balance (ABAL). There are two different instances of a generic account, SAVINGS accessible with a cheque book and ACCESS operable from ATM machines, thus requiring a PIN number being its additional attribute (APIN#). Any bank CUSTOMER may OWN (relationships - diamonds) a number of SAVINGS and ACCESS accounts and being a PERSON will certainly have a name (PNAME - a key) and address (PADDR).

In semantic and object-oriented data models, the relationships between objects and their types take the application rather than set-theoretic semantics (e.g. OWNS as opposed to 1:1, 1:N, or M:N in classical data models). They serve a three-fold role, first they implicitly define the structures of an underlying database (e.g. an ER-diagram may map onto a relational or hierarchical database schema), then they constrain the types of attribute values that may fill-in the record fields (e.g. only a CUSTOMER may OWN bank ACCOUNTs), and finally a complex network of relationships may act as a definitional context for new entity types (e.g. we may wish to define a CUSTOMER as any PERSON that OWNs an ACCOUNT with the bank). A number of built-in relationships is customarily provided to enrich the representational capabilities of the system and to increase its efficiency at the same time - ISA, PART-OF, MEMBER-OF, etc. For instance, ISA relationship defines a hierarchy of data types, thus providing greater economy of description as the object attributes are being inherited to the type instances and need not be defined again at lower levels of the hierarchy (e.g. inheritance of ACCOUNT attributes by SAVINGS and ACCESS accounts).

```
/* Account: Acc# AName ABal */
account(10897, brown_sav, 10979).
account(11879, smith_acc, -500).
account(12987, brown_acc, 899).
account(9867, warren_sav, 111200).
account(10765, lois_acc, 100).

/* Pin: Acc# APin# */
pin(11897, 1477).
pin(12987, 2451).
pin(10765, 86993).

/* Person: PName PAddr */
person(brown, melbourne).
person(smith, sydney).
person(jones, sydney).
person(warren, hobart).
person(lois, canberra).
person(black, melbourne).

/* Owns: PName, Acc# */
owns(brown, 10897).
owns(brown, 12987).
owns(smith, 11879).
owns(warren, 9867).
owns(lois, 10765).
owns(black, bank).
```

QUERY:

```
?-    suspect(AccNo, Person).
```

```
/* Access: Account with a pin */
access(Acc, AName, ABal, APin) :-
      account(Acc, AName, ABal),
      pin(Acc, APin).

/* Savings: Account without a pin */
savings(Acc, AName, ABal) :-
      account(Acc, AName, ABal),
      not pin(Acc, _).

/* Customer: Person owning an account */
customer(PName, PAddr, Acc) :-
      person(PName, PAddr),
      owns(PName, Acc),
      account(Acc, _, _).

/* Audit criteria */
suspect(Acc, PName) :-
      customer(PName, _, Acc),
      Acc < 10000.

suspect(Acc, PName) :-
      account(Acc, _, ABal),
      ABal < 0,
      owns(PName, Acc).

suspect(Acc, PName) :-
      access(Acc, _, _, APin),
      APin > 9999,
      owns(PName, Acc).
```

Figure 3 - A deductive database

Reasoning. Another aspect of DBMS drawing constant attention of database researchers, developers and users is their reasoning capability. A classical approach to data manipulation is via a suite of independent programs, preparing, storing and retrieving database information. An alternative way of encoding procedures interpreting database information is to store them as inference rules in the database itself; this way they could be added, modified and queried as any other data record in a database (thus bringing it to the level of KBMS). Such inference rules may interact with each other, create additional dependencies and functions not stored in the database nor its structure explicitly, they could be applied to take care of defining complex data structures, multiple views, and additional information relating to syntactic and semantic data integrity. Extending databases by application-oriented inferences could certainly improve software development productivity by reducing (or possibly eliminating) the need to recompile application software after schema modification.

This tendency to enrich data dictionaries by addition of application-oriented inferences, lead to the development of logical data models, of which deductive databases and rule-based expert systems are the most popular. To illustrate reasoning capabilities of such data models, let us consider a deductive database based on the previously discussed semantic schema (Figure 2). Let us assume that an appropriate mechanism found a correct mapping of a semantic data model into a deductive database consisting of a number of PROLOG-like rules (Figure 3). Two data and two index files were created :- ACCOUNT holding all the necessary information of customer accounts and PERSON describing people the bank deals with (to include customers), PIN extends some of the ACCOUNTs by additional attribute APIN#, and OWNS assigns ACCOUNTs to PERSONs. The remaining elements of our semantic schema are represented as database inference rules, thus defining ACCESS and SAVINGS accounts as ACCOUNTs with or without APIN# respectively, and a CUSTOMER as a PERSON who OWNS a bank ACCOUNT. We have also provided a definition of SUSPECT rules reflecting the auditing mini-criteria from Figure 1. From a classical perspective ACCESS, SAVINGS and CUSTOMER may be seen as views, whereas SUSPECT as data integrity constraints. A sample audit query (listed in the table) will find all suspect account, i.e. Smith's as having negative balance, Lois' using an illegal PIN, and Warren's as having an incorrect account number. Additional audit trails, views and constraints may be added to this database with a great ease and flexibility without ever touching the application programs using it.

#	**NLP Query**	**KBMS Query**
1	List all bad accounts.	print(Account | suspect(Account, _)).
2	Print names of all bank customers.	print(Name | customer(Name, _, _)).
3	Give me a list of customers using an access account.	print(Name | customer(Name, _, Account), access(Account, _, _, _)).
4	Show me all the people having an overdraft.	print(Name | account(Account, _, Balance), Balance < 0, owns(Name, Account)).
5	Now, give me their addresses.	print(Address | person(Context.Name, Address)).

Figure 4 - Natural language queries

Interfacing. Quality and flexibility of user interfaces frequently decides of a software product failure or success. In database management systems an introduction of stand-alone query languages elevated them from unfriendly programmer-dependent environments to powerful user-driven applications. Addition of AI-based knowledge manipulation techniques to DBMS allowed the users to enter new realms of independence - the use of natural language in database processing (NL - Allen 1987, Gazdar and Mellish 1989).

It is often surprising that an impressive NL front-end to a deductive database of restricted domain could be built very quickly. Such systems are usually based on a simple parser using a form of semantic grammar constructed almost entirely of a database schema, synonym lists and concept taxonomies, all providing a mapping between sentence structures and logical rules, thus allowing translation of user natural language input into a series of database queries. Figure 4 illustrates the potential of the NL technology, it shows a number of English phrases and their corresponding database form (cf. Figure 3). From the examples, it can be seen that the system dictionary must be able to describe the correlation between a number of synonym words and their equivalents DB terms (e.g. commands "list", "show" and "give" vs "print", or entity "people" vs "persons", then constraints "bad" vs "suspect", a more difficult is to relate "people" vs "person's names" and "overdraft" as "balance < 0" - #4). In parts a lexicon simply includes entity and relationship names defined in a database schema (e.g. "suspect", "account" and "customer"), others must be additionally provided as they are related to the entity attributes normally not represented in PROLOG rules (but listed in ER diagrams, Figure 2, e.g. "name", "address" in "person"). NL parser is then responsible to determine the clausal form of our DB query (simple variable query - #1 and #2, a conjunction of existing entities and relations - #3, insertion of ad hoc constraints into the query - #4, or context-sensitive queries - #5).

NL extensions to computer software have already met users enthusiastic approval and were shown to have a very positive influence on their productivity (Napier et al. 1989). Over the years English was successfully applied to the variety of software tools, e.g. program generators (Heidorn 1976), on-line help systems (Wilensky 1984) and of course database query languages (Harris 1984).

3 AI Practice

This paper described and illustrated a number of AI methodologies recently adopted by database researchers, developers and users. The paper focuses on the benefits of applying knowledge representation, reasoning and interfacing techniques to DBMS. It would be a totally false perception, however, to think that the reverse flow of ideas and methods does not exist. On the contrary, most of the recent commercial AI tools lean heavily on experience learnt by DBMS researchers and developers.

Semantic data modeling, reasoning, and natural language interfaces provide uncomparable intelligence to data and knowledge bases. In the past, however, software "artificial" intelligence was achieved only at the price of increased representational and processing complexity limiting AI and KBMS systems to toy-like applications of research rather than practical value. Only recently, due to the integration of Computer Sciences, KBMS and DBMS in particular, the results of Artificial Intelligence pursuits could turn into real-life products. Database researchers helped improving previously unmanageable or unscalable problems, and *optimised* KBMS processing, increased KBMS *capacity* and *efficiency*, provided them with *distributive* powers, etc. (It is beyond the scope of this paper to go into depth of these problems, nevertheless, interested readers should refer to Ullman 1988 for further details.)

So, do we really witness an impact of AI on DBMS? No, it is rather an integration of Computer Sciences that has a tremendous influence on the form and function of contemporary software. Hopefully it will be the computer end-user who benefits most out of the integration of AI and commercial concepts in the near future.

References

Allen, J. (1987): *Natural Language Understanding*, The Benjamin/Cummings Pub. Co., Menlo Park, CA.

Brodie, M.L. (1986): "Database Management: A Survey," in Brodie, M.L. and Mylopoulos, J. (Eds), *On Knowledge Base Management Systems*, New York: Springer-Verlag, pp 201-218.

Chen, P.P.S. (1976): "The Entity-Relationship Model: Towards a Unified View of Data," *ACM Transactions on Database Systems*, Vol 1, No 1.

Cybulski, J.L. (1987): *Development of Concepts and Methodologies for the Representation of Contextual Information in Knowledge Based Systems*, Msc Thesis, Royal Melbourne Institute of Technology, Department of Computer Science, Ch 2.

Dayal, U. and Smith, J.M. (1986): "PROBE: A Knowledge-Oriented Database Management System," in Brodie, M.L. and Mylopoulos, J. (Eds), *On Knowledge Base Management Systems*, New York: Springer-Verlag, pp 227-258.

Gazdar, G. and Mellish, C. (1989): *Natural Language Processing in Lisp*, Addison-Wesley, Wokingham, England.

Harris, L.R. (1984): "Experience with INTELLECT: Artificial Intelligence Technology Transfer," *The AI Magazine*, Vol 5, No 2, 43-50.

Heidorn, G.E. (1976): "Automatic Programming Through Natural Language Dialogue: A Survey," in Rich, C. and Waters, W.C. (Eds) *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann, pp 203-214.

Hull, R. and King, R. (1987): "Semantic Database Modeling: Survey, Applications, and Research Issues," CRI-87-20, Computer Research Inst., USC.

Levesque, H. and Mylopoulos, J. (1979): "A Procedural Semantics for Semantic Networks," in N.V. Findler (Ed) *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, pp 93-120.

Maier, D., Stein, J., Otis, A. and Purdy, A. (1986): "Development of an Object-Oriented DBMS," *OOPSLA'86*, ACM, New York, pp 472-482.

Morris, K., Ullman, J.D. and Van Gelder, A. (1986): "Design Overview of the NAIL! System," *Proc. Third Intl. Conf. on Logic Programming*, pp 554-568.

Napier, A.H., Lane, D.M., Batsell, R.R., and Guadango, N.S. (1989): "Impact of Restricted Natural Language Interface on Ease of Learning and Productivity," *CACM*, Vol. 32, No. 10, pp 1190-1198.

Sholovits, P. (1986): "Knowledge-Based Systems: A Survey," in Brodie, M.L. and Mylopoulos, J. (Eds), *On Knowledge Base Management Systems*, New York: Springer-Verlag, pp 339-352.

Sowa, J.F. (1984): *Conceptual Structures*, Addison-Wesley, Reading, MA.

Stonebreaker, M. and Rowe, L.A. (1986): "The Design of Postgres," *ACM SIGMOD Intl. Conf. on Management of Data*, pp 340-355.

Ullman, J.D. (1988): *Principles of Database and Knowledge-Base Systems*, Vol. 1 and 2, Computer Science Press.

Wilensky, R. (1984): "Talking to Unix in English: An Overview of an On-Line UNIX Consultant," *The AI Magazine*, Vol 5, No 1, pp 29-39.

Woods, W.A. (1975): "What's in a Link: Foundations for Semantic Networks," in Bobrow, D.A. and Collins, A. (Eds) *Representation and Understanding: Studies in CognitiveScience*, Academic Press, New York, pp 35-82.

Biographical Note

Mr. Jacob L. Cybulski graduated with B.App.Sci. and M.App.Sci. in Computer Science from RMIT. His interests include Artificial Intelligence, Software Engineering and the commercial applications thereof. His previous work experience involved him as a lecturer, scientist, consultant and commercial software developer at various organisations in Australia and overseas, currently Mr. Cybulski works in the capacity of a deputy director of Amdahl Australian Intelligent Tools Program at La Trobe University.