Toward Intelligent Representation of Database Content

Joseph D. Oldham and Victor W. Marek

Department of Computer Science, University of Kentucky, Lexington, KY 40506,
{oldham,marek}@cs.engr.uky.edu

Abstract

We address the problem of automating the display of database records in an intelligent way. By this we mean the synthesis of complete multimedia documents from database records. We propose an architecture for mapping diverse data stored in a database to markup language (SGML) programs. These programs are ready for final presentation. The mapping is based on a computational extension of the linguistic concept of *register*. The resulting presentation represents data as information in an intelligent way. General conditions for such a system are discussed. Our own treatment of registers as *rule based* computational structures is offered with some early results on the behavior of rule based registers.

1 Introduction

We address the problem of automating the display of database records in an intelligent way. By this we mean the synthesis of complete multimedia documents from databases which are not otherwise document databases.

Our databases may be object oriented databses with multimedia data. Because our interest is in *populations* of users a suitable database should hold data that is to be used by at least one community of users with strong communication conventions. For example a suitable database might consist of: *medical records* (communities might include physicians, nurses, financial officers of various sorts), *accident reports* (police, insurance adjustors and claimants, courts), *student records* (advisors, deans, admissions officers in other programs, potential employers, students).

"Intelligent display" means a generated display document must satisfy the document conventions of a user community approximately as well as a document built by hand from the same data and for the same purpose. Our only restriction is that the purpose does not include any but very shallow analysis. Making the database system "speak the communal language" facilitates the ability of any member of the community to make sense of the data. So "intelligent" means the ability to "speak the vernacular" of a community of users. [PR87] refers to a text generation system which can do this as a *tailoring* system.

1.1 Approach

For our approach we take advantage of the existence of markup languages, which become the target languages of the kind of systems we describe. The hard prob-

lem is to move from data itself, facts and objects, to complete document content embedded in a markup language program. For this we turn to the language of *register theory*, from descriptive linguistics [GC78]. The control of various registers marks the sophistication of a language user [GC78]. Our "language" is explicitly multimedia. Although registers have been discussed in the literature of natural language processing [BP89] we review the concept briefly: A *register* is a triple $\langle$ *field, mode, tenor* $\rangle$, where:

Field is the domain of discourse. Field understanding provides context for the synthesis of the document. This implies representing the common knowledge of the domain.

Mode is the means of communication, e.g. written vs. spoken, spontaneous vs. planned, or even, *this* plan for written discourse vs. *that* plan for written discourse. Mode is taken here as describing a document *form*.

Tenor describes the purpose of the communication, e.g. to inform, persuade, etc. Tenor informs choice of rhetorical device used within the framework of the mode to represent content.

We claim that this concept of register can be formalized in a way suitable for intelligent display of multimedia database information. In this paper we offer the beginnings of our formalization of registers: *generalized rule defined registers*. Our register is a computational structure which will accept as input a database record and transform that record through a series of steps into a completed document in a target markup language. From now on we assume SGML [HR90] as a target language. Others are possible. The marked up document may be presented to the user via an SGML processor.

Control of registers is a marker of sophistication in language use. Endowing a database with control of a register should increase the user's ability to receive information from the database by making the form of the data intuitive. If the system presentation of data lacks some of the informational coherence of a human presentation then *information* is lost in the human - system interaction. Registers are a way of trying to restore a subtle form of information loss by making the data more readily or better understood by a user, who will be better able to make necessary and correct inferences in less time. We do not believe the problem of intelligent presentation of data from a database as information tailored to a user has been adequately addressed.

1.2 Some Related Work

We begin with some work in text generation. The problem of determining the content of a text document has been studied by McKeown [MC85]. Paris [PR87] introduced the term "tailoring" to represent the process of making text documents user appropriate. Paris and Bateman [BP89] phrase text tailoring in terms of register theory [GC78]. This body of work tells us that with respect to textual components of documents syntactic structure, vocabulary and word order

are key to audience appropriate data presentation. We do not yet know what the multimedia correlates are. In fact we are particularly interested in how the availability of multimedia objects alters text content. In terms borrowed from Devlin [DV91] what is the effect on a document when analog information is included? IVORY [CW93] is a tool developed at Stanford University to assist the physician in writing progress notes. It takes a relatively naive approach to tackle the problem of text document generation in one domain. The impact of global discourse structure on the discourse at sentence level is studied in [RV95] with an eye toward text generation as an application.

It may be tempting to look at document generation from data as an extension to very dynamic views. [CBR94, DE94] provide examples of how efforts to extend views to Object-Oriented database management systems (OODBMS) are concerned by other issues such as relating the model (or scheme) used by an application to the scheme of a database. Fundamentally a document is more than a view. It is worth noting that if a tailoring system required a data scheme different from that of the DBMS then using a view mechanism to create an intermediate data representation would avoid attempts to alter the DB scheme directly. Such an approach would allow a presentation module to be built on top of an existing database, with no alteration to the underlying data model required. We term such an approach *noninvasive.*

[CACS94] considers the problem of mapping structured (SGML [HR90]) document data type definitions (DTDs) to an OODB schema (in particular the O_2 [BB88] system.) Instances of a DTD (a document) are then mapped to objects and values under the scheme found by the mapping from the DTD. Structured document management and retrieval is facilitated. They explicitly do not consider going the other way which is our problem. Several points are worth remarking: (1) When a class of documents can be generated from stored facts management and retrieval of those documents is no longer different from any other database operation. (2) Even in storing whole documents and taking advantage of their structure one must confront the issue of order. Order matters in structured documents, but not in database schema. This results in an invasive approach with respect to the O_2 data model in [CACS94]. (3) Under our scheme an inverse mapping from document instance to database record does not in general exist. This is compatible with the results in [CACS94] as the database there is a document database while ours is not. In our approach specific facts stored may lead to linguistic or other media expression in the document, i.e. facts may be "forgotten" or blurred. Once blurred only a class of facts is, in general, recoverable. However it is common and often necessary to store "facts" rather than documents.

[EG95a, EG95b] describe *the writer's problem.* If W is a writer with some background knowledge and some facts and W wants to convey with some particular sense the circumstances associated with the facts by creating a document using some *mode* (in our case a multimedia mode), and if W can assume the reader has certain background knowledge, then how does W proceed? [EG95a, EG95b] focuses on the complementary *reader's* problem. It is worth remarking that since

the writer's and reader's problems are complementary, any document created by one system should be understandable by another system. This holds even if a precise inverse mapping to data does not exist.

2 Examples

For the first example we assume the following (non 1-NF) toy scheme and assume HTML rather than SGML:

⟨ NAME, SEX, AGE, PIC, ADDR, DATE, VS, SYMPS ⟩

Example 1. ⟨ Marek,Victor, M, 25, /PtPhoto/VMarek.gif,123/Main/Atown/KY/40512, 8/24/95, 132/84, 92, 99.O, 16, (cough/dry,painful/1/D/nil)⟩

Our register system will deliver a document represented like this:

```
<H1> Patient Victor Marek </H1>
Photo of Mr Marek: <IMG SRC="~/PtPhoto/VMarek.gif", ALT="Mr. Marek">
<H2> Personal Data: </H2>
     Male, 25 years old. <P>
<H3> Mailing Address</H3>
     123 Main St. Anytown, KY 40512<P>
<H2> Presentation of Thursday 8/24/95 </H2>
VS: Within Normal Limits
<P>
Mr. Marek presented with a c/o dry, painful cough. He was afebrile.
The cough began 1 day prior and remained unchanged. No other symptoms
noted.
```

Here is another tuple:
< Oldham/Joseph, M, 82, nil, 217/Cross/Btown/OH/50582, 8/24/95, 100/62, 102, 99.2, 22, (cough/productive, nocturnal/3/D/(dry, constant/2/W)) >

From this we might see:

```
<H1> Patient Joseph Oldham </H1>
<H2> Personal Data: </H2>
     Male, 82 years old. <P>
     <H3> Mailing Address</H3>
     217 Cross St. Sometown, OH 50582<P>
<H2> Presentation of Thursday 8/24/95 </H2>
VS: 100/62, 102, 99.2, 22
<P>
Mr. Oldham presented with mild tachycardia and a temperature of 99.2.
His complaint was 3 days of productive night-time cough. Previously
he had had 2 weeks of dry, constant cough. No other symptoms noted.
```

Finally, we might see either patient presented through a different register, e.g. as:

```
<H1> Mr Joseph Oldham, age 82</H1>
Mr.Oldham had c/o cough, was tachycardic (102) and febrile (99.2). Had
approximately 17 days of cough. Over the prior 3 days cough became
productive and restricted to night time.
```

Example 2. Here we do not show the raw data, which describes a basketball game between two teams, but we do show intermediate representations following application of field, mode and tenor rules in turn.

Step 1:

A set of *Field Rules* is evaluated on the statistics for the game and all participating entities represented in the scheme. We receive something of the following, which we call: the *field representation*:

```
(University of Arkansas Lost)
(Thurman Top Scorer For Loser)
(Williamson Below Avg Points and Rebounds For Loser)
(University of Arkansas Without Beck)
(Series Tied)
(University of Kentucky Won)
(Mashburn, Delk Top Scorers For Winner)
(Pricket Above Avg Rebound For Winner)
(Rhodes, Ford Injured For Winner)
```

This set of facts is our first *intermediate representation*. These facts are isolated but represented appropriately to the field and goal of the intended presentation. In general at this stage the facts are unordered. Note there may have been a rule such as:

> If . . .
> **and** W plays > X Minutes on average for Y
> **and** W does not play against Z
> . . .
> **then** . . . (Y without W) . . .

This set of rules has most to do with *content selection* for the final document.

Step 2:

We now apply *Mode Instantiation Rules* which decide on the structure of the presentation, within a given set of constraints. (For example two injuries to one team may cause injuries to be mentioned in the lead paragraph.) In this case we may see:

```
(Rhodes, Ford injured for University of Kentucky)
(University of Kentucky Won Over University of Arkansas)
(University of Arkansas Without Beck)
```

```
(Series Tied)
*
(Pricket Above Avg Rebound For University of Kentucky)
(Mashburn, Delk Top Scorers For University of Kentucky)
(Thurman Top Scorer For University of Arkansas)
(Williamson Below Avg Points and Rebounds For University of Arkansas)
*
(Box Score)
(Scoring Graph)
```

Step 3:
The final processing by the register may now occur, resulting in a marked up document, our final *source representation*. *Tenor rules* are applied to give a pleasant surface representation to the selected and ordered facts.

```
<H1> UK Defeats UA </H1>
--Lead Paragraph
Rodrick Rhodes and Travis Ford were injured last night when the
University of Kentucky won over the University of Arkansas by the
score of .... Arkansas played without Corey Beck. The series
is now tied at ....
<P>
Kentucky was aided by a strong rebounding performance by Jared Prickett
and paced in scoring by Mashburn and Delk. Thurman led Arkansas in
points, but it was a quiet night for Corliss Williamson who was
held below his average in both  rebounding and points.
<IMG SRC="BoxScore">
<IMG SRC="ScoringGraphic">
<IMG SRC="~/Kentucky/Photos/mashburn.gif">
```

3 Some formal definitions

We are now ready to look at our approach to computational registers in some detail. Information is represented in *databases*. Databases are defined by means of *schemes*. We expect object-oriented databases see [Ul88, BM93]. The actual content of a database is its *instance*. An instance consists of *records*. In Section 2 we have seen that our databases may consist of objects. Notice that in example 1, the attribute SYMPTOMS was a list of strings. Similarly the attribute NAME was a pair of strings. We will assume (and this always can be enforced) that the domains of attributes are disjoint. The information stored in the database may be text, numerical information or multimedia information (binary large objects, or BLOB fields). In this case the information stored in the field is interpreted as a pointer to the actual location where the information is stored. In our example, the field PtPhoto contained the pointer to the file vmarek.gif with the actual BLOB information. Similar arrangements for aural information, video etc. can be made.

The scheme of the database is a vector of strings called names of attributes. The scheme is supplemented by another vector, of the same length, of *types* of attributes. These could be types such as string, float, BLOB/filename, pairs of strings, lists of strings, etc. We do not define types here.

In addition to instances (composed of objects called *records* to stress their database provenance) we deal with two more types of entities. These are *source representations* and *final presentations*. Although in the abstract setting the set I of source representations can be arbitrary, we think about it as a set of SGML programs (see Section 2). In contrast, the set of final presentations is the collection of interpreted SGML documents on display. Text formatting provides an analogy: source representations are .dvi files, final presentations are text documents [KN84]. We picture our situation as follows:

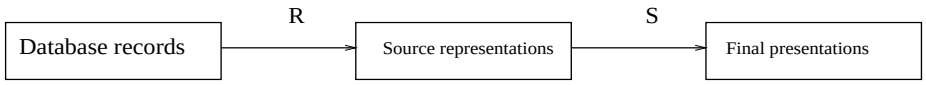

Fig. 1. Transforming the database information into texts

Figure 1 describes the overall approach as exemplified in Example 1. Database records are transformed via the register mapping into finished SGML source representations of documents ready to be processed by an SGML system into a final presentation document.

We will concentrate on the mapping R from database records to source representations. Such mapping, following linguistic terminology, is called a (generalized) *register*.

In the setting of example 2 we see that R is a composition of mappings. Steps 1 through 3 in 2 correspond here to rule based mappings F, M, T, essentially capturing the notions of *field, mode,* and *tenor* in registers. F selects, based on field knowledge, certain facts for translation into expressions appropriate for the field. This is a database mapping. M takes these expression, and using knowledge common to the field and a description of the mode, or form, of the document *orders* these expressions appropriately for intelligent presentation. Again, this is a mapping from one database scheme to another. This captures the "deep" structure of the presentation Finally T delivers a final source representation ready for an SGML or similar system to translate into a polished, reasonably arranged document – a presentation document. We are no longer interested in how the source representation becomes a final presentation as this is a matter for an SGML interpreter, and we have delivered an SGML document. In this setting our picture becomes as in figure 2.

In principle, a register can be any mapping from records to source representations. We are interested, however, in registers defined by means of *rules*. That is rules describe the way in which database information is transformed into the source representation. For instance, in our Example 1 the first field, NAME is a pair of strings (string1/string2). The information contained in that field is

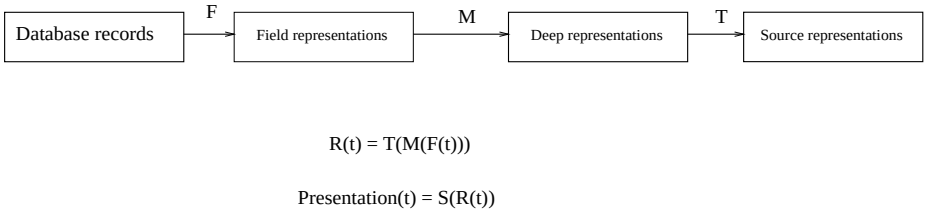

Fig. 2. Transformation with register

interpreted as a line of the resulting SGML program:

<H1> Patient string 2 string1 </H1>

This mode of processing our record is described as a rule:

d: **If** NAME = <string1/string2> **and** ...
Then *i*.firstline = <H1> Patient string2 string 1 </H1> **and** ...

(Rules can be quite complex, and we displayed a fragment of a rule to see some of its salient features).

Let us look carefully at our rule. Among other things this rule tells us that when we process this record and produce the written progress note for a physician, the field NAME is represented as patient information, with the first string represented as surname, and the second string as a given name. Thus the pair of strings *Doe*/*John* is interpreted as a line "Patient John Doe". The additional word "Patient" comes from the context – we talk about patients, and not (for instance) participants in an automobile accident. Notice the fact that the NAME field is primary information is additionally highlighted by the choice of the display information (<H1>). Both features presented above are related to the fact that our source representation assign *meaning* to database information.

Formally, we define a rule as an entity of the form:

d: **If** $p_1(At_1)$ **and** ... $p_k(At_k)$
Then i

Here, $p_1, \ldots, p_k$ are some formulas describing properties of attribute values. For instance, such a property may be that the temperature is between 98.0 and 99.2 (i.e. normal).

Definition 1. A (generalized) register R is rule-based if R is defined by a collection of rules of the form:

d: **If** $At_1 \in X_1$ and ... $At_k \in X_k$
Then i

Here $i \in I$ and $X_1 \subseteq Dom(At_1), \ldots, X_k \subseteq Dom(At_k)$ Further, X_j is called the j^{th} condition in the rule d and a record t matches rule d if for all j, $1 \leq j \leq k$, $t(j) \in X_j$.

Notice that if the rule d does not explicitly impose conditions on attribute At_j then $X_j = Dom(At_j)$. The condition of the form $At_j = x$ is equivalent to one of the form $At_j \in \{x\}$ More generally, if $\varphi(\cdot)$ is any formula, then the condition of the form $\varphi(At_j)$ is equivalent to one of the form $At_j \in X_j$ where $X_j = \{x : \varphi(x)\}$. For instance, if one of our attributes is called *Temp* and the condition on the attribute *Temp* is: $100.0 \leq Temp \leq 100.5$ (and the step is .1) then our condition is equivalent to $Temp \in \{100.0, 100.1, \ldots, 101.5\}$.

Once the register is defined by use of rules, the order in which the rules are processed usually matters. Thus registers based on rules are, in general, nondeterministic.

We define now some classes of registers.

Definition 2. (a) A rule-based register R is consistent if for every record t and for every pair of rules d_1, d_2 in R if t matches both rules d_1 and d_2 then d_1, d_2 have the same conclusion.
(b) A register R is *complete* if every record matches at least one rule in R.
(c) A register that is both consistent and complete is called *functional*.
(d) A register R is called *partitional* if for every two rules d_1, d_2 in R and for every attribute At_j, the j^{th} conditions X_j^1 and X_j^2 in d_1 and d_2 respectively satisfy the property:

$$X_j^1 = X_j^2 \quad \text{or} \quad X_j^1 \cap X_j^2 = \emptyset$$

(e) Two registers R_0, R_1 are called *equivalent* if for $j = 0, 1$, whenever a record t matches a rule d in R_j then there is a rule d' in R_{1-j} such that t matches d' and the conclusion of d' is the same as that of d.

4 Properties of registers defined by rules

In this section we report on some technical results obtained while working on the properties of registers. These technical results allow us to understand better how registers behave.

Proposition 3. *(a) For every register R there is an equivalent partitional register R'*
(b) When R is functional then so is R'.

Now we will characterize registers in which a record can match only one rule.

Definition 4. A functional register R is called decompositional if for every $i \in I$, $R^{-1}(i)$ is empty or there exist nonempty sets $Z_1 \subseteq Dom(At_1), \ldots, Z_k \subseteq Dom(At_k)$ such that

$$R^{-1}(i) = Z_1 \times \ldots \times Z_k$$

Since for $i_1 \neq i_2$, $R^{-1}(i_1) \cap R^{-1}(i_2) = \emptyset$, decomposability means that that for each i, $R^{-1}(i)$ is a "hyperrectangle", and that those hyperrectangles cover the entire universal instance $Dom(At_1) \times \ldots \times Dom(At_k)$.

We have this characterization of decompositional registers.

Proposition 5. *A functional register R is decompositional if and only if there exists register R' such that R' is equivalent to R, and for every $i \in I$, i is the conclusion of at most one rule in R'.*

We will characterize now registers that are both decompositional and partitional. These are particularly simple registers which are generated by very simple rule sets. We start with a definition.

Definition 6. 1. A register R is called *simple* if R is both decompositional and partitional.

2. Let $D = (At_1, \ldots, At_k)$ and $D' = (At'_1, \ldots, At'_k)$ be database schemes. Let $U = \bigcup\{Dom(At_j) : 1 \leq j \leq k\}$ and $U' = \bigcup\{Dom(At'_j) : 1 \leq j \leq k\}$. A mapping $f : U \to U'$ is a database map if for every $x \in U$, and every j, $1 \leq j \leq k$, $x \in Dom(A_j)$ implies that $f(x) \in Dom(At'_j)$.

3. A register R is called *invertible* if all its rules have different conclusions and each rule is matched by exactly one record.

It turns out that database maps can be composed with rules.

Definition 7. Let $D = (At_1, \ldots, At_k)$ and $D' = (At'_1, \ldots, At'_k)$ be database schemes and let $f : U \to U'$ be a database map as above. Let R' be a register over D'. The composition of register R' and database map f is a register R consisting of rules over D (but with the same set of source representations I) :

d: **If** $At_1 \in f^{-1}(X_1)$ **and** $\ldots$ $At_k \in f^{-1}(X_k)$
Then i

such that the rule

d': **If** $At'_1 \in X_1$ **and** $\ldots$ $At'_k \in X_k$
Then i

belongs to R'. This composition is denoted $f \circ R'$.

Proposition 8. *A register R over a database scheme D is simple if and only if there exists a database scheme D' and a database map $f : U \to U'$ and an invertible register R' over D' such that $R = f \circ R'$.*

Proposition 9. *For every register R there is a database map f and a register R' such that:*
(a) R is equivalent to $f \circ R'$, and
(b) All rules in R' are matched by precisely one record.

Acknowledgments

The second author gratefully acknowledges a partial support by NSF grant IRI-9400568.

References

[BB88] F. Banchilon, G. Barbedette, V. Benzaken, C. Delobel, S. Gamerman, C. Lecluse, P. Pfeffer, P. Richard and R. Velez. The Design and Implementation of O_2, an Object–Oriented Database System, Proceedings of the OODBS II Workshop, Bad Munster, FRG, September 1988.

[BP89] J. A. Bateman and C. L. Paris. Phrasing a Text in Terms the User Can Understand, Proceedings of IJCAI 89.

[BM93] E. Bertino and L. Martino. *Object-Oriented Database Systems: Concepts and Architectures*. Addison-Wesley Reading, MA, 1993.

[CACS94] V. Christophides , S. Abiteboul , S. Cluet and M. Scholl. From Structured Documents to Novel Query Facilities, Proceedings of SIGMOD 94, ACM, 1994.

[CBR94] A. B. Cremers, O. T. Balovnev and W. Reddig. Views in Object–Oriented Databases, Proceedings of ADBIS'94, 1994.

[DE94] M. Dobrovnik and J. Eder. Adding View Support to ODMG–93, Proceedings of ADBIS'94, 1994.

[DV91] K. Devlin. *Logic and Information*. Cambridge University Press, 1991.

[EG95a] D. Estival and F. Gayral. A Study of the Context(s) in a Specific Type of Texts: Car Accident Reports, Computation and Language EPrint Archive, paper 9502032, 1995.

[EG95b] D. Estival and F. Gayral. An NLP Approach to a Specific Type of Texts: Car Accident reports, Proceedings of Workshop on "Context in Natural Language Processing", IJCAI 95.

[GC78] M. Gregory and S. Carroll. *Language and Situation: Language Varieties and Their Social Contexts*. Routledge and Kegan Paul Ltd., 1978.

[HR90] E. van Herwijen. *Practical SGML*. Kluwer Academic Publishers, 1990.

[CW93] K.E. Campbell, K. Wickert, L.M. Fagan and M.A. Musen. A Computer–based tool for generation of pregress notes, Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC, November, 1993, pp. 284–284.

[KN84] D.E. Knuth. *The* Tex *Book*. Addison Wesley, Reading, MA, 1984.

[MC85] K. R. McKeown. *Text Generation*. Cambridge University Press, 1985.

[PR87] C. L. Paris. *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise*. Ph.D. Thesis, Columbia University, 1987, also Technical report CUCS–309–87, Columbia University.

[PA83] Z. Pawlak. Rough Sets, *International Journal of Information and Computer Sciences* 11:145-172, 1982.

[RV95] W. Ramm and C. Villiger. Global Text organization and Sentence–Grammatical Realization: Discourse Level Constraints on Theme Selection, Proceedings RANLP 95, to appear.

[Si64] R. Sikorski. *Boolean Algebras*. Springer Verlag, Heidelberg, 1964.

[Ul88] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, Rockville, MD, 1988.

This article was processed using the LaTeX macro package with LLNCS style