Neural Database Model

Moawia Elfaki Yahia[1] Ahmed Mohammed Elsawi[2]

[1]Faculty of Mathematical Science, University of Khartoum, 11115 Khartoum,
P.O.Box 321, Khartoum, Sudan Email: meyahia@hotmail.com
[2]Computer Man College for Computer Studies, 11111 Khartoum,
P.O.Box 12291, Khartoum, Sudan Email:Elsawi76@hotmail.com

Abstract

The work described herein is primarily concerned with recent trends of upgrading Database from its traditional retrieving to that of having a conscious function.

The present study establishes a new model, which constructs and merges the neural networks using the Relational Database (RDB) and Fourth Generation Language (4GL) methods. This is illustrated by neural network model, which is based solely on the RDB and SQL method. This led to a new technique and opened new channels for the use of the RDB and SQL methods in different branches of programming and system analysis, other than their traditional use. A new role is also presented for the concepts of classical storage and active storage. The control structure oriented and data structure oriented have also been considered. The work also deals with a practical example of transforming the source code from Third Generation Language to Fourth Generation Language.

Keywords: Neural Network, Database Models, Forth Generation Languages, Intelligent Database.

1. Introduction

Interest in computer science has developed so rapidly that it covered almost every branch of science. It plays a leading role as an interdisciplinary subject. As an illustration, consider it embracing to both anatomy and artificial intelligence, which led to the creation of artificial neural networks. The rapid development of this branch in the last decade gave it an important role in other branches of computer science such as programming and intelligent machines.

The intermarriage between Artificial Neural Network (ANN) and Database (DB) has led to vast development in research works that led to the foundation of Knowledge discovery and Data Mining; and resulted in transforming the classical database to what is known now as Intelligent Database.

Stored Procedures (SPs) and triggers have been proposed as a means of embedding neural network concepts and techniques into relational database [1]. This method would empower database engines with the ability to resolve several tasks during data manipulation. The engine will INSERT, UPDATE, and DELETE data as instructed by the Data Manipulation language (DML) code residing in the executing application. At the same time, the

engine will use the event to activate a neural network internal to the database.

Embedding neural nets efficiently process each DML transaction as it incurs, adding only a fraction of a second to the completion of the request. The insignificant overhead of each transaction is possible because each engine prepares and optimizes SPs when they are created. After invocation, SPs are loaded by the engine and buffered for future use. Subsequent calls no longer require a disk read to retrieve the object; instead, the SP is executed directly from memory. In fact, when loaded, it may never be retrieved from disk again.

2. Artificial Neural Networks

An Artificial Neural Networks is a model that emulates a biological neural network. As you will see, today's neural computing uses a very limited set of concepts from biological neural system [7].

The concepts are used to implement software simulation of massive parallel processes that involve processing elements (also called artificial neurons or neurodes) interconnected in a network architecture.

The artificial neuron receives inputs that are analogous to the electrochemical impulses that the dendrites of biological neurons receive from other neurons. The output of the artificial neuron corresponds to signals sent out from a biological neuron over its axon. These artificial signals can be changed similarly to change occurring at the synapses.

The state of the art in neural computing rests on our current understanding of biological neural networks. We are, however far from having an artificial, brain-like machine [7].

Despite extensive research in neurobiology and psychology, important questions remain about how the brain and the mind work. This is just one reason why neural computing models are not very close to actual biological systems.

Nevertheless, research and development in the area of ANNs is producing interesting and useful systems that borrow some features from biological systems [6].

2.1 Fundamental of Neural Network…

2.1.1 Components and Structure…

The network is composed of processing elements, organized in different ways to form the network's structure [6].

2.1.1.1 Processing Elements...

The Artificial Neural Network is composed of artificial neurons (to be referred to as neurons); these are the processing elements **(**PEs**).** Each of the neurons receives input(s), processes the input can be raw data or output of other processing elements. The output can be the final product or it can be an input to another neuron.

2.1.1.2 The Network...

Each Artificial Neural Network is composed of collection of neurons that are grouped in layers, which are the Input Layer, The intermediate Layer (called the Hidden Layer), and the Output Layer. Several hidden layers can be placed between the Input and the Output layers.

2.1.2 Structure of the Network…

Similar to the biological networks an Artificial Neural Network can be organized in several different ways (topologies); that is, the neurons can be interconnected in different ways. Therefore, ANNs appear in many configurations.

In processing information, many of the processing elements perform their computations at the same time. This parallel processing resembles the way the brain works, and it differs from the serial processing of conventional computing.

2.1.3 Processing Information in the Network…

Once the structure of a network is determined, information can be processed. Several major concepts related to the processing must be defined these are:

Inputs: Each one of the inputs corresponds to a single attribute and the numeric value of an attribute is the input to the network. Several types of data can be used as inputs, however preprocessing is needed.

Outputs: The output of the network is the solution to a problem. The ANN assigns numeric values; for example, **1** for "Yes" and **0** for "No" and the purpose of the network is to compute the values of the output.

Weights: A key element in an Artificial Neural Network is the weight. Weights express the relative strength (or mathematical value) of the initial entering data or the various connections that transfer data from layer to layer.

In other words, the weights express the relative importance of each input to a processing element. Weights are crucial; it is through repeated adjustments of weights that the network "learns."

Summation Function: The summation function finds the weighted sum of all the input elements entering each processing element.

Transformation (Transfer) Function: The summation function computes the internal stimulation, of activation level, of the neuron. Based on this level, the neuron may or may not produce an output.

3. Databases

A database is a single organized collection of facts, records and data normally set up to meet the information needs of major parts of an organization and to make specific items easy to find.

Databases are usually organized around an identifier, or key, which can be anything from an account number to a surname. The use of this sort of identifier means that individual data items can be accessed rapidly and efficiently with the minimum of fuss.

Databases are also stored on remote computers and can be accessed by anyone who has access to communications technology [4].

3.1 Active Database Concepts

Rules that specify actions that are automatically triggered by certain events have been considered as important enhancements to a Database system. In fact the concept of triggers is a technique for specifying criteria types of active rules has existed earlier. However, much of research into what a general model for active database should look like has been done since the early models of triggers were proposed [2].

The Generalized Model for Active Database and Oracle Triggers has been used for specifying active database rules and are referred to as the Event-Condition-Action, or ECA model. A rule in the ECA has three components [2]:

1. The event or (events) those trigger the rule.

2. The condition that determines whether the rule action should execute.
3. The action to be taken.

3.2. Intelligent Database

DBMS which performs data validation and processing traditionally done by application program Most DBMSs provide some data validation, e.g. rejecting invalid dates or alphabetic data entered into money fields, but often most processing is done by application programs. There is however no limit to the amount of processing that can be done by an intelligent database as long as the process is a standard function for that data.

Examples of techniques used to implement intelligent databases are Constraints, triggers and SP.

Moving processing to the database aids data integrity because it is guaranteed to be consistent across all uses of the data. Mainframe databases have increasingly become more intelligent and personal computer database systems are rapidly following [8].

3.3. Data Mining

Databases today can range in size into the terabytes — more than 1,000,000,000,000 bytes of data. Within these masses of data lies hidden information of strategic importance. But when there are so many trees, how do you draw meaningful conclusions about the forest?

The newest answer is data mining, which is being used both to increase revenues and to reduce costs. The potential returns are enormous. Innovative organizations worldwide are already using data mining to locate and appeal to higher-value customers, to reconfigure their product offerings to increase sales, and to minimize losses due to error or fraud. Data mining is a process that uses a variety of data analysis tools to discover patterns and relationships in data that may be used to make valid predictions.

3.4 Fourth Generation Language 4GL:

Databases programmed by Fourth Generation Language which defined as a High-level computer languages accessible to people without formal programming skills, with feature such as icons, objects, help facilities, pull down menus and templates which present authors with options for every activity which they are likely to require. The software itself generates the program code required to translate these instructions into the commands, that are required by the operating system and device drivers require [4].

Moving processing to the database aids data integrity because it is guaranteed to be consistent across all uses of the data. Mainframe databases have increasingly become more intelligent and personal computer database systems are rapidly following [8].

4. Neural Database Model

The proposed model consists mainly of two parts:

Part I: This part describes the general idea of transforming the ANN components to RDB components.

Part II: This part is concerned with a model, which transforms a Perceptron Model to a RDB model.

Regards the neuron as the smallest unit in the NN. As explained earlier, its importance was also stressed. The neuron receives a group of inputs multiplied by their corresponding weights, and then discharges them as only one output. This unifying characteristic of a neuron extends to group of neurons, which acts as a layer in

parallel. The group of such layers forms the NN [5]. The process of distribution of the layers and methods of input and output results in a group of networks having different topologies.

4.1 Translation Process

To translate the above scenario to the world of RDB, imagine that there is a central unit with satellite groups of components that interact simultaneously with mutual responses. Figure 4.1 is a database table called Nets_Table that consists of two fields. The first *NET_Id* is a primary key, while the latter *NET* is the field concerned with the network and its properties. This is considered here as there information.

*Net_Id	Net
1	Net 1
2	Net 2
3	Net 3
|	|

Nets Table Fiaure 4.1

Every net has a specific topology. It consists of several layers. Figure 4.2 gives a table for the structure of layers. It consists of three fields, starting with the *Net_Id* as a Foreign key from the Nets_Table. *Layer_Id* is the second Field and in combination with the *Net_Id*, form a Primary Key for this table. The *Layer_Type* is the third field and its function is to specify the type of layer (Input, Hidden, or Output).

Net_Id	*Layer_Id	Layer_type
1	1	Input
2	2	Hidden
3	3	Output
|	|	

Layer Table Figure 4.2

A table of more interest is the Neuron Table. It is illustrated in Figure 4.3. It is the object concerned with the role of the fundamental unit in forming the ANN.

This table called Neuron_table consists of five fields, the *Net_ID*, and *Layer_ID* form the Foreign Key from the layer table. These two fields form with the *Neuron_ID* the Primary Key for this table. Also there is a *Neuron_Output* field which collects the output from the neuron. The last field in this table is the *Neuron_Biase* form a special neuron input ranging from (-1, 1).

Net_Id	Layer_Id	Neuron_Id	Neuron_Output	Neuron_Bias
1	1	1		
2	2	2		
3	3	3		
|	|	|		

Neuron Table Figure 4.3

As explained earlier, every input is a complained a weight. This is presented in the table in figure 4.4, which is known as the Weight table. All the fields of this table are considered as Primary Key. The first three of which (*Net_Id, Layer_Id, Neuron_Id*) are regarded as a Foreign Key from the Neuron_table.

Pair of two other tables shown in figures 4.5, 4.6 is the Activation_Function table and the Learning_Rules table. These two tables play leading role in the process of learning and subsequently by the substitution after learning.

*Function_Id	Activation Functions	Functions
1	Unity or Identity	F(x) = x
2	Linear Function	F(x) = μx + c
3	Threshold Function	F(x) = 0 | 1; x < θ | x > θ
|	|	|

Activation Function Table Figure 4.4

*Rule_Id	Learning Rules	Rule
1	Simple Hebb Rule	dij = L * Outi * Inpj
2	Cressberg Rule	dij = L * Outi * (Inpj – wij)
3	Delta Rule	dij = L * (goali – Outi) * Inpj
|	|	|

Activation Function Table Figure 4.5

A question poses itself here is that these Database Objects are designed to store the data; and that all the fields particularly those in the above two tables, contain a simple fact that does not reach the standard of an instruction. So how can this model deal with such a problem, particularly that an example in ANN requires a high-grade language that is capable of taking it. The next section shows how to resolve such a problem.

4.2. Data Activation

As has been pointed earlier, the new model treats the ANN simply as a type of dummy data. This poses the question of how to activate this data. Consider the following example:

```
SQL> select &x from tab;
Enter value for x: *
old   1: select &x from tab
new   1: select * from tab

TNAME                          TABTYPE CLUSTERID
------------------------------ ------- ---------
INPUT                          TABLE
LAYER                          TABLE
NET                            TABLE
NEURON                         TABLE
OUTPUT_TABLE                   TABLE
TP                             TABLE
WEIGHT                         TABLE

7 rows selected.
SQL>
```

In this example we found that the "*" has replaced the variable "&x"; but hardly any progress has been made in the sense of dealing with data and its replacement with another form. However, the following example explains how to activate the data to control. Consider the table in figure 4.7.

In this table we notice the presence of two records. The first one is designated by the number *1*together with *SYSDATE* function. Now on applying a simple modification to the example above, we find that if, in the above table, Select statement is applied and the output value is stored in encryption and then replace this variable by "&x" as a new variable. now as the

Id	Control Functions
1	Sysdate
2	*
|	|

Control Functions Table Figure 4.7

table contain
commands n te
a symbol table in compilers. So, we can easily activate all the data stored in previous tables. This completes the process of activating the data. This model named Neural Database.

The above model is faced with some difficulties when dealing with SINGLE PERCEPTRON. That is why part II of this paper is devoted to a special model for the PERCEPTRON.

4.3. Perceptron Model

The PERCEPTRON MODEL or also called the ONE LAYER PERCEPTRON. In this model we practically transform it to RDB model.

As an illustrative example to this model we shall consider the (OR GATE). This model consist of number of tables as shown below:

1. INPUT TABLE:

This is concerned with the input quantities as explained in the table in figure 4.8.

X1	X2	Tp
0 (1,1)	0 (1,2)	0
0 (2,1)	1 (2,2)	1
1 (3,1)	0 (3,2)	1
1 (4,1)	1 (4,2)	1

OR GATE Table Figure 4.8

In the above truth table we notice that the number of (X1 ,X2) to evade the pitfall of future change in the number of inputs, we resort to the concepts of RDB which save us the trouble of resolving this difficulty.

If we regard the INPUT_TABLE as a table with three fields but instead of representing (X1 ,X2,Tp), it represents (Row_Id, Col_Id, Value) where (Row_Id, Col_Id) represents a PRIMARY KEY. The

INPUT_TABLE is then seen to look like a matrix, as in figure 4.9.

Row_Id	Col_Id	Value
1	1	0
1	2	0
2	1	0
2	2	1
3	1	1
3	2	0
4	1	1
4	2	1

OR Gate Matrix Figure 4.9

Using this model is would have acquired a big space for flexibility in both the vertical and horizontal directions for data input. This type of flexibility con not found in 3GL where the upper limit for an array is fixed and limited and its RUN-TIME cannot be varied easily, so that a programmer is forced to use the LINKED LIST and thus resulting in a further complication to the problem.

Table in figure 4.10 is concerned with the transfer of the INPUT PROCEDURE from 3GL to 4GL.

Pascal Input/Calculate Procedures

```
Procedure Inputs; (*patterns Input*)
begin     (*Inputs*)
  For j:= 1 to 4 do
   begin   (*Loop J*)
     Writeln('Pattern (',j,')');
     Writeln('---------------');
     For i := 1 to 2 do
      begin (*Loop i*)
        Write('X(',i,') = '); Read(X[j,i]);
      end;  (*Loop i*)
     Write('Desired = '); Read(Tp[j]);
   end;   (*Loop J*)
end;  (*Inputs*)
(*****************************************)
Procedure calculate;
begin
 For j := 1 to 4 do
  begin
    Neti := 0;
    For i:= 1 to 2 do
     begin
     Neti := Neti + (X[j,i] * W[i]);
     end;
    If Neti >= Theta then
      Oi[j] := 1
    else
    if Neti < Theta then
      Oi[j] := 0;
   Error :=  error + (Tp[j] - Oi[j]);
   writeln('error',error);
   end;
end;
```

PL/SQL Input Procedure

```
declare
vcol_id number(1);
vrow_id number(1);
vtp number(1);
voi number(1);
dummy number(1);
input_value number(1);
weight_Value number(32,30);
begin
:Global.neti := 0;
:Global.theta := 0.5;
:Global.error := 0;
for vrow_id in 1 .. 4 loop
for vcol_id in 1 .. 2 loop
select value into input_value from input
where
row_id = vrow_id
and
col_id = vcol_id;

select weight.wvalue into weight_value
from
 weight
where
wrow_id = vrow_id
and
wcol_id = vcol_id;

:Global.Neti := :Global.Neti + nvl(input_value *
weight_value,0);
end loop;

if :Global.Neti >= :global.theta then
insert into tp values(vrow_id,1,1);
else
insert into tp values(vrow_id,1,0);
end if;
select distinct(tp.orow_id),tp.ovalue into dummy,vTp from
tp where tp.OROW_ID = vrow_id
and tp.ocol_id = 1;
Select distinct(output_table.orow_id),output_table.ovalue
into  dummy, vOi from output_table where
output_table.orow_id = vrow_id
and output_table.ocol_id = 1;
:global.error := :global.error + (vtp - voi);
message(:Global.Error);
Message(' ');
message(nvl(:global.neti,0));
message(' ');
end loop;


end;
```

Transfer INPUT Procedure from 4GL to 3GL Figure 4.10

2. WEIGHT TABLE:

As Pointed in [3], every input is multiplied by a specific weight. The adjustment procedure of the weights is really the crucial trick in learning the ANN.

In the same manner followed in the INPUT_TABLE the WEIGHT_TABLE represents (wrow_Id, Wcol_Id, Wvalue) where (wrow_Id, Wcol_Id) represents a PRIMARY KEY. Next we have the weight table in figure 4.11.

wRow_Id	wCol_Id	wValue
1	1	0.5
1	2	0.3
2	1	0.5
2	2	0.3
3	1	0.5
3	2	0.3
4	1	0.5
4	2	0.3

OR GATE weight matrix Figure 4.11

In this table we notice so redundancy in the *wvalue* filed. This redundancy can be rectified by an appropriate normalization techniques. For the present problem, being pressed for time, I have not been able to achieve this. I hope that some future researchers in this field may be able to resolve the problem of normalized models.

The screen output shown in figure 4.12 shows some redundancy in the *TARGET_OUTPUT* although the problem dealt with is a non-database item, but the nature of the program led to this effect.

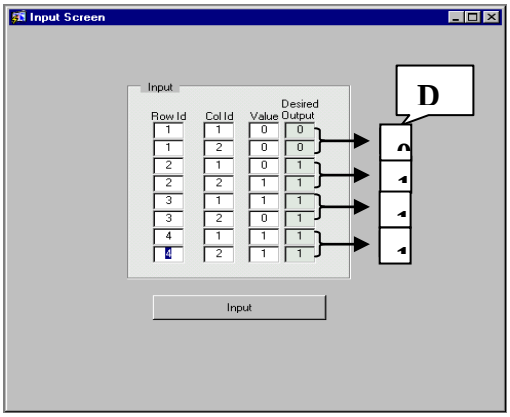
The operation of multiplying the input by the weight results in the *NETi* as shown in table 4.10. The INSERT statement fills the Actual output table, which called *OUTPUT_TABLE*. There is another table, which forms the Desired Output filled by firing the KEY_NEXT_ITEM trigger in the DESIRED_OUTPUT field using another INSERT statement.

Conclusion

The novel feature of this work is the design of a new model, called Neural Database. Which introduces the ideas of RDB into the ANN treatment. Another achievement is that of a practical example whereby PERCEPTRON MODEL was converted into RDB MODEL, giving excellent results, comparable with those when using a 3GL for solving the model.

This work is an initiation for using the concepts of RDB in postulating a variety of models in the language of this approach. There are several ideas that can be pursued and thus opening new research channels. For example, the concepts of Compiler Design can be dealt with using the same approach we used for the ANN in which the data was transformed into control, as dealt with in the symbol table.

It can also be used in security by denying access to any unauthorized person who is trying to fiddle with the database.



OR GATE input screen Figure 4.12

The Neural Database model has some advantages shown below:

1. **Active Storage**

   We have used this terminology on the assumption that the classical method of storage is inert since it is only concerned with store and retrieve; while the present method introduces the idea of thinking, investigating, and recognizing the material being stored. Active storage can be very practical in the environment of DISTRIBUTED DATABASE, where it is assumed that the NEURAL DATABASE is linked to a storage server of its own and which communicates with the servers of the other units of the network.

2. **Development of Researches**

   This new model will certainly enhance research in different fields such as DATA

MINING, KNOWLEDGE DISCOVERY and INTELLIGENT DATABASE.

3. Dually of two methods

The dual matching of the two methods will certainly reveal a lot of the qualities of each of them.

4. The 4GL language

The present work revealed the effectiveness of the 4GL and that it can be utilized in fields other than that of the Database.

5. Trap Door

The method showed that a TRAP DOOR could be planted in Database while its functioning.

6. Networks

It is now possible through the present method, to deal simultaneously with many neural network connections in parallel. Also it is possible for the NN to exchange operations during the process of learning.

7.Database Engines

It makes possible the programming of INTELLIGENT DATABASE ENGINES.

References

[1] The conscious Database, Michael L. Gonzalies, WWW.dbpd.com.
[2] Database Programming & Design, volume 10 – Number 7, July 1997.
[3] Introduction to Data Mining and Knowledge Discovery, Third Edition, By: Two Crows Corporation.
[4]Fundamental of Database Systems, Third Edition, Ramiz Emasri,Shamkant B. Nacathe, Addison-Wesley 2000.
[5]Encyclopedia of Computer Science, Third Edition, Ralston Reilly, Chapman & Hall Inc, 1993.
[6] Neural Networks 'Comprehensive Foundation', Send Edition, Simon Haykin, Mc Master University, Prentice Hall Inc, 1999.
[7] Neural Computing 'The Basics', Larry Medskr, IEEE Press, 1997.
[8] http://foldoc.doc.ic.ac.uk.