Semantic Encoding and Compression of Database Tables

D. Paul Benjamin, Pace University, benjamin@pace.edu
Adrian Walker, Reengineering LLC, adrianw@snet.net

This working paper is being circulated as a basis for possible joint exploitation of the underlying US patent and international patents pending. Version 20040308.

Abstract

A method is described to allow the encoding and compression, of one or more tables of data, by splitting each table into two or more subtables, to allow the splitting to be followed by permutation of the subtables, or by construction of an interconnection table using a collection of permutations and keys or key numbers, and to optionally allow the permuted tables to be padded with additional data. This is done in such a way that the original tables cannot be queried or reconstructed from the subtables without knowledge of the permutations and keys. The method also allows efficient querying of the subtables to retrieve information that was in the original tables, optionally over a network, and efficient reconstruction of the original tables, given knowledge of the permutations and key numbers. The method also supports billing for authorized access to tables, and rapid changes in encoding to guard against key theft.

Introduction

A database is a collection of logically related data, usually stored on computers, that is set up either to be questioned directly, or to provide data to one or more applications.

Data in a database is logically represented as a collection of one or more tables. Each table is composed of a series of rows and columns. Each row in the table represents a collection of related data. Each column in the table represents a particular type of data. Thus, each row is composed of a series of data values, one data value from each column.

A database and its applications may reside on one computer, or the database may be distributed over a number of computers that are connected by a network, such as a local area network, a virtual private network, or the internet. Duplicates of part or all of a database may be stored on different computers, for performance, availability, or other reasons. The applications that use a database may reside on one of the computers where the database resides, or may reside on other computers connected to the database over a network.

Semantic Encoding is a novel way of securing the contents of a database, and of making those contents available only to authorized individuals, or groups of individuals, or programs. Authorization is given by making known a collection of keys or key numbers and/or permutations. SE is different from
previous methods for securing databases. It can be used in isolation, and it can also be used to complement the previous methods.

A database that is logically organized into tables of data is typically managed by a Relational Database Management System (RDBMS). The RDBMS provides commonly needed services, such as: means to retrieve data in a related way from more than one table in response to a question; means to update data; means to ensure integrity of the data with respect to constraints; means to control access to the data; means to index the data for rapid access.

Typically, a question put to a database will be written in a notation that is based on a mathematical construct called the Relational Algebra. The answer to a question is itself a table. There are three main operations in the Relational Algebra that can be used together to construct an answer table for a question: Projection of a table on some of its columns results in a new table consisting of the set of rows obtained by omitting the remaining columns; Selection of rows meeting a certain criterion from a table results in a new table consisting of only those rows of the original table that meet the criterion; Joining two tables results, conceptually, in a new table having rows formed by appending a row of the first table to a row of the second table, and Selecting only such rows that have the same values in certain designated columns in the two tables. The constructs of the Relational Algebra are easily recognizable in the Structured Query Language (SQL) that is a common means of accessing and manipulating data in an RDBMS.

The data in a database often contains information that should be held in confidence, and that should only be made available to authorized users or programs. For example, the data may be confidential to a certain business organization, or it may contain military secrets. RDBMS access controls basically allow certain privileges, such as the permission to question or to update the data, to selected user IDs or programs, based on the knowledge of a password or passwords. As such, RDBMS access controls provide a first line of defense for confidential information that is held in a database. However, experience shows that, while there are strong reasons for making data from an RDBMS available over networks to authorized users or programs, there is an ongoing cycle of penetration by unauthorized users followed by incremental improvement in access controls. This can be seen by visiting the US government's National Infrastructure Protection Center at www.nipc.gov. For example, NIPC advisory 01-003 lists a security hole that allows unauthorized users to tunnel Structured Query Language (SQL) requests through a public connection to a private back-end network. It is believed that unauthorized users have obtained the details of many credit cards by such methods, see www.sans.org/newlook/alerts/NTE-bank.htm.

A second line of defense is to encrypt some or all of the entries in the tables in a database, using a standard method such as the Data Encryption Standard (DES) or public key cryptography. However this line of defense is also subject to a cycle of penetration followed by improvements. In addition, there is currently active research into advances in mathematics and software that could lead to rapid methods of unauthorized decryption of data that has been encrypted using these standard methods. Moreover, some information, such as the number of rows in a table, remains anyway open to unauthorized users or programs. In addition, the performance of the RDBMS for authorized users is reduced by the need to perform decryption for every query, and encryption for every update.

There is a need for an improved method of hiding data from unauthorized users and programs, while making it efficiently available to those who are authorized.
This paper offers a third line of defense, based on a Semantic Encoding method, that is different from either access control or encryption of database entries. Semantic encoding can be used standalone, or with any combination of prior methods.

**Algorithms for Semantic Encoding and Decoding**

The basic idea in Semantic Encoding is to store some projections of a table, rather than the table itself. As illustrated in Figure 1, there are many ways to assemble a collection of projections to form a possible original table.

![Figure 1](image)

Figure 1  Many Ways of Assembling Projections of a Table

However, only one of the ways of assembling the projections will be correct. An attacker who tries to assemble the projections will not know whether or not his result is correct.

During the Semantic Encoding process, the projections are created, and some additional key information is recorded. An authorized person or program can use the key information to assemble portions of the correct table as needed.
Next we describe the Semantic Encoder and Decoder algorithms. During Encoding, we can add padding to the key information, and we describe a measure designed to tell how good the padding is. We show how Semantic Encoding can be used to support billing for the use of a database. Then we give algorithms for secure **insert** and **delete** operations.

For each algorithm (except for billing) we first give a flowchart, then we give an English text version.

**Semantic Encoder**

\[
\text{Inputs: } \text{table } T \text{ having a set of column names } C.
\]

Two subsets $C_1$ and $C_2$ of $C$ such that $C_1 \cup C_2 = C$. A positive integer $s$.

One or more key numbers that can be used to define a bijective function.

Let $r$ be an integer, $r > |T|$, and let $R = \{0, 1, \ldots, r-1\}$. Choose a bijective function $h: R \mapsto R$, and let $g$ be the inverse function of $h$. For example, choose to define $h$ as follows: $h(j) = (pf(j) + q) \mod r$, where $r$ is a prime number, $p < r$ is a prime number, $q < r$ is a positive integer, and $f$ is a permutation of $R$ defined from the key numbers using the conventional encryption method 3-DES.

Construct $T_1$ by projecting $T$ on the columns $C_1$. Construct $T_2$ by projecting $T$ on the columns $C_2$. Reorder the rows of $T_1$ and $T_2$. Optionally pad $T_1$ and/or $T_2$ with additional rows. Number the rows of $T_1$ and $T_2$. The row numbers can be unevenly spaced over $R$, and can be non-sequential. (See also Note in Figure 2)

Construct an interconnection table defined by $T_{12} = \{ \langle h(i), h(j) \rangle \mid T_1[i, u], T_2[j, v], T[u \| v] \}$, or

$T_{12} = \{ \langle h(i), j \rangle \mid T_1[i, u], T_2[j, v], T[u \| v] \}$, or

use $h$ to permute the row numbers of $T_1$

Figure 2 Semantic Encoder Flowchart Part 1 of 2
\[ Ta = \{ (u || v) | T1(i, u), T12(i, j), T2(j, v) \} /\ T \]

- \( Ta \) has at least one row?  
  - yes  
    - \( u1 = R - (i | T1(i, u) \)  
      \( u2 = R - (j | T2(j, v) \)  
      \( P1 = \{ <i', j'> | i' \text{ in } u1 \text{ or } j' \text{ in } u2, \)  
      \( g(i') \text{ in } u1 \text{ or } g(j') \text{ in } u2 \} - T12 \)
  - no

- \( |P1| < s \)
  - yes
    - \( P2 = P1 \)
  - no
    - \( T12pad = T12 \cup P2 \)

- \( k = |P1|/s \), rounded down.
- \( P2 = \{ <i', j'> | \text{ for } t=1,\ldots,s, \text{ is the element in the position } (t \times k - k/2 + 1) \text{ of } P1 \} \)

Output: \( h(\text{or } p, q, r, s), T1, T2, T12pad \)

Figure 3  Semantic Encoder Flowchart Part 2 of 2
The Semantic Encoder in text form:

1.1 INPUTS

A table \( T \) having a set of column names \( C \). Two subsets \( C_1 \) and \( C_2 \) of \( C \) such that \( C_1 \cup C_2 = C \). A positive integer \( s \) indicating the amount of padding required. One or more keys or key numbers that can be used to define a bijective Function.

1.2 CHOOSE a function \( h \)

Let \( r \) be an integer, \( r > |T| \), and let \( R = \{0,1,...,r-1\} \). Choose a bijective function \( h: R \rightarrow R \), and let \( g \) be the inverse function of \( h \). For example, choose to define \( h \) as follows. \( h(j) = (pf(j) + q) \mod r \), where \( r \) is a prime number, \( p < r \) is a prime number, \( q < r \) is a positive integer, and \( f \) is a permutation of \( R \), defined from the keys or key numbers using the conventional encryption method 3-DES.

1.3  PROJECTION and ROW NUMBERING

Construct a table \( T_1 \) by taking the relational algebra projection of \( T \) onto the column names \( C_1 \). Reorder the rows of \( T_1 \). Optionally add more rows to \( T_1 \) containing plausible information (semantic padding). Then assign a unique row number from \( R \) to each of the rows of \( T_1 \). The row numbers can be unevenly spaced over \( R \), and can be non-sequential. (See Note 1, below.)

Construct a table \( T_2 \), using the column names \( C_2 \), in the same manner as the construction of \( T_1 \).

1.4 INTERCONNECTION AND PERMUTATION

Construct an interconnection table defined by

\[
T_{12} = \{ <h(i),h(j)> \mid T_1(i,u), T_2(j,v), T(u || v) \}
\]

Where \( T(u || v) \) denotes a row of \( T \) with entries \( u \) under the column names \( C_1 \) and entries \( v \) under the column names \( C_2 \). Note that \( u || v \) is not necessarily a simple concatenation of a row \( u \) with a row \( v \).

Note 1: with certain patterns of data in \( T \), e.g. if \( C_1 \) contains a key of \( T \), we can simplify this step by omitting the interconnection array, reordering the rows of \( T_1 \), and simply permuting the row numbers of \( T_1 \) according to the function \( h \). We can also optionally leave duplicate rows in \( T_1 \) and/or \( T_2 \).

Note 2: We can alternatively define

\[
T_{12} = \{ <h(i),j> \mid T_1(i,u), T_2(j,v), T(u || v) \}
\]
1.5 CHECK THE CHOICES OF ROW NUMBERS AND OF THE FUNCTION $h$

Construct

$$Ta = \{ (u \parallel v) \mid T1(i,u), T12(i,j), T2(j,v) \} \cap T$$

If $Ta$ has at least one row, optionally (see Note 3) go back to step 1.3 and assign different row numbers to $T1$ and/or $T2$, such that $Ta$ has no rows.

If no such row numbers for $T1$ and $T2$ can be found optionally (see Note 3 below) go back to step 1.2.

1.6 PADDING

The input positive integer $s$ is the desired amount of padding.

Let $U1 = R - \{ i \mid T1(i,u) \}$ and $U2 = R - \{ j \mid T2(j,v) \}$,

And define an ordered set

$P1 = \{ <i',j'> \mid i' \text{ in } U1 \text{ or } j' \text{ in } U2, \ g(i') \text{ in } U1 \text{ or } g(j') \text{ in } U2 \} - T12$

$P2 = P1$, if $|P1| \leq s$

Otherwise, let $k = |P1|/s$, where '/' denotes division and rounding down, and let

$P2 = \{ <i',j'> \mid \text{for } t=1,\ldots,s, <i',j'> \text{ is the element in }$

The position $(t^k - k/2 + 1)$ of $P1$)

Finally, let $T12pad = T12 \cup P2$

Note 3: If it is desired to prevent an unauthorized user or program from correctly concluding that a certain row is not in $T$, one can readily see how to modify the above steps to allow a controlled subset of $T$ to be present in $\{ (u \parallel v) \mid T1(i,u), T12pad(i,j), T2(j,v) \}$. If this is done, an unauthorized user or program still cannot know which rows, amongst others, were actually in $T$. One can also readily see how to distribute the rows in $P2$ in other useful ways.

1.7 CHECK THE PADDING

Find the padding security of $T1$, $T2$, $T12pad$ using a measure such as that in MEASURE OF PADDING SECURITY, in Figure 5. If the padding security needs to be increased, go back to step 1.3 or to step 1.2.

1.8 OUTPUT

The function $h$ (e.g. defined by $p$, $q$, $r$, and $f$), $T1$, $T2$, $T12pad$
The Semantic Decoder

2.1 INPUTS

Functions h and g, (for example specified using p, q, r, and f), C1, C2, T1, T2, and T12pad, as described in 1. SEMANTIC ENCODER

2.2 DECODE

We describe the decode procedure for the main encoder described in steps 1.1 to 1.8 of the SEMANTIC ENCODER. One can easily see how to adapt the decode procedure described here to the cases described in Note 1 and Note 2 in step 1.4 of the SEMANTIC ENCODER, and how to adapt the procedure described here to other ways of defining the functions h and g.

Let h be the function defined by $h(j) = (pf(j) + q) \mod r$, and let g be the inverse function of h.

Note: This is a decode procedure for the main encoder described in step 1.4 of the SEMANTIC ENCODER in Figure 2. One skilled in the art will easily see how to adapt the decode procedure described here to the cases described in Note 1 and Note 2 in step 1.4 of the SEMANTIC ENCODER.
Construct the table

\[ T_d = \{ (u \| v) \mid T_1(i, u), \; i' = h(i), \; T_12\text{pad}(i', j'), \; j = g(j'), \; T_2(j, v) \} \]

Where \((u \| v)\) denotes a row with entries \(u\) under the column names \(C_1\) and entries \(v\) under the column names \(C_2\). Note that \(u \| v\) is not necessarily a simple concatenation of a row \(u\) with a row \(v\).

2.3 OUTPUT

Output the table \(T_d\).

Padding security

\[
\begin{align*}
\text{Input: } & T, \; T_1, \; T_2, \; T_12 \text{ and } T_12\text{pad}, \\
& \text{as described in Figures 2 and 3, SEMANTIC ENCODER}
\end{align*}
\]

\[
T_a = \{ (u \| v) \mid T_1(i, u), \; T_12(i, j), \; T_2(j, v) \}
\]

\[
\text{PadEffect} = (|T_12|/|T_12\text{pad}|)^* E \quad \text{(See Figure 5 part 3.2)}
\]

\[
\begin{align*}
\text{yes} & \quad |T_a \setminus T| \text{ is 0 ?} \\
\text{no} & \quad S = 1 - \text{PadEffect} \\
C & = \min \{|(u + v') \text{ in } T_a \cup \{(u' + v) \text{ in } T_a\}| \\
& \quad (u + v) \text{ in } (T_a \setminus T)\}
\end{align*}
\]

\[
S = \max(1 - \text{PadEffect} - 1/C, 0)
\]

Output: \(S\)

Figure 5  Measure of Padding security Flowchart
Padding Security in text form:

3.1 INPUT

T, T1, T2, T12, T12pad as described in SEMANTIC ENCODER

3.2 PADDING EFFECT

Ta = \{ (u || v) | T1(i,u), T12(i,j), T2(j,v) \}

PadEffect = ( |T12|/|T12pad| ) ** E ,

where E is a positive integer, e.g. 10.

Note: A smaller value of E will make the padding security measure call for more pad rows in the interconnection table

3.3 CONFUSION

If |Ta \ T| is not 0 then

\[ C = \min \{(u + v') \in Ta \cup (u' + v) \in Ta | (u + v) \in (Ta \setminus T)\} \]

3.4 PADDING SECURITY

\[ S = \begin{cases} 1 - \text{PadEffect}, & \text{if } |Ta \setminus T| \text{ is 0} \\ \max(1 - \text{PadEffect} - 1/C, 0), & \text{otherwise} \end{cases} \]

3.5 OUTPUT

The number S, between 0 and 1, that measures padding security. A value of S close to 1 corresponds to high padding security.
Billing of Client Questioning of Servers or Peers via a Network

This section uses the notations set out in Semantic Encoder and Semantic Decoder

4.1 SET UP ONE OR MORE CLIENTS

Let the function h (or p, q, r, f), C1, and C2 be as described in SEMANTIC ENCODER.

Transmit h, or p, q, r, and f or the index number of f, C1, and C2, encrypted by conventional means, to the client(s) on a network.

4.2 SET UP ONE OR MORE SERVERS

Let T1, T2, and T12pad, as described in SEMANTIC ENCODER, be set up, optionally in an RDBMS, on one or more servers connected to the network.

4.3 CLIENT ASKS QUESTION

Suppose a client wishes to ask for the rows (u || v) of T such that u has certain properties P.

Client transmits C1 and P to server(s).

4.4 SERVER(S) SEND SEMANTICALLY ENCODED RESPONSE TO QUESTION

A server receiving the question consisting of C1 and P responds by sending

\[ T1P = \{ <i,u> | T1(i,u), u \text{ has properties } P \} \]
\[ T2P = \{ <j,v> | T1P(i,u), i'=h(i), T12pad(i',j'), j=g(j'), T2(j,v) \} \]
\[ T12padP = \{ <i',j'> | T1P(i,u), i'=h(i), T12pad(i',j') \} \]

to the client.

4.5 CLIENT SEMANTICALLY DECODES RESPONSE TO QUESTION

A client receiving the response T1P, T2P, and T12padP to its question C1 and P, uses its knowledge of the functions h and g, to semantically decode the response as

\[ TP = \{ (u || v) | T1P(i,u), i'=h(i), T12padP(i',j'), j=g(j'), T2P(j,v) \} \]

Note 4: It will be clear how to modify steps 4.4 and 4.5 to take account of the alternatives described in Note 1 and Note 2 in the Semantic Encoder. It
will also be clear how to extend steps 4.3 to 4.5 to the case that the property P applies to both u and v.

4.6 SERVER ORGANIZATION BILLS CLIENT ORGANIZATION FOR USE OF DATA

Server organization can at any time change the values of h or of p,q,r, and f and reissue the new values to just those client organizations that are in good standing from the point of view of payments.

Server organization can use different functions h or p,q,r,f values for different groups of clients or for different client organizations.

---

**Insert a Row into a Table**

**Input**: A row \((u || v), T1, T2, T12pah h and g,\)

as described in **SEMANTIC ENCODER**.

---

![Flowchart for Inserting a Row into a Table](image)

---

Figure 6  Flowchart for Inserting a Row into a Table
Insert a row into a table, text form:

5.1 INPUT

A row \((u \| v)\), \(T_1\), \(T_2\), \(T_{12pad}\) \(h\) and \(g\), as described in SEMANTIC ENCODER.

5.2 CHECK THAT THE ROW TO BE INSERTED IS NOT ALREADY PRESENT

If \(T_1(i,u), T_2(j,v), i'=h(i), T_{12pad}(i',j'), \) and \(j=g(j')\),
then output 'the row is already present', otherwise go to 5.3.

5.3 INSERT INTO \(T_2\) and \(T_{12pad}\)

If \(T_1(i,u), i'=h(i), \) and there is no \(j'\) such that
\(T_{12pad}(i',j'), j=g(j'), \) and \(T_2(j,v)\)
then
find a \(j\) in \(R\) that is not a row number of \(T_2\),
insert \(T_2(j,v)\), and insert \(T_{12pad}(i',h(j))\)
else go to 5.4

5.4 INSERT INTO \(T_1\) and \(T_{12pad}\)

If \(T_2(j,v), j'=h(j), \) and there is no \(i'\) such that
\(T_{12pad}(i',j'), i=g(i'), \) and \(T_1(i,u)\)
then
find an \(i\) in \(R\) that is not a row number of \(T_1\),
insert \(T_1(i,u)\), and insert \(T_{12pad}(h(i),j')\)
else go to 5.5

5.5 INSERT INTO \(T_1\), \(T_2\), and \(T_{12pad}\)

If there is no \(i'\) such that, for some \(j_1'\),
\(T_{12pad}(i',j_1'), i'=h(i), \) and \(T_1(i,u)\)
and there is no \(j'\) such that, for some \(i_1'\)
\(T_{12pad}(i_1',j'), j'=h(j), \) and \(T_2(j,v)\)
Then
find an \(i\) in \(R\) that is not a row number of \(T_1\),
find a \(j\) in \(R\) that is not a row number of \(T_2\),
and insert \(T_1(i,u), T_{12pad}(h(i),h(j)), \) and \(T_2(j,v)\).
Delete a Row from a Table

**Figure 7** Flowchart for Deleting a Row from a Table
Delete a row from a table, text form:

6.1 INPUT

A row \((u \mid v)\), \(T_1\), \(T_2\), \(T_{12}\) pad \(h\) and \(g\), as described in SEMANTIC ENCODER.

6.2 CHECK THAT THE ROW TO BE DELETED IS ACTUALLY PRESENT

If
\[ T_1(i,u), i'=h(i), T_{12}(i',j'), j=g(j'), \text{ and } T_2(j,v), \]
then go to 6.3, otherwise output 'the row to be deleted is absent'

6.3 DELETE THE ROW

Delete \(T_{12}(i',j')\)

6.4 OPTIONALLY REMOVE UNUSED ENTRIES IN \(T_1\) and \(T_2\)

If there is no \(j_1, j_1', v_1\) such that
\[ i'=h(i), T_{12}(i',j_1') \text{ and } T_2(j_1,v_1) \]
then delete \(T_1(i,u)\)

If there is no \(i_1, i_1', u_1\) such that
\[ j'=h(j), T_{12}(i_1',j') \text{ and } T_1(i_1,u_1) \]
then delete \(T_2(j,v)\)
Conclusion

The permutations used in Semantic Encoding can be generated by many different methods, including software and/or hardware based pseudo-random number generators, software and/or hardware based encryption methods, or natural sources of truly random numbers.

Semantic Encoding and Decoding of tables of information can optionally be used for billing for use of the information. SE provides an improved system for encoding information, in that the information cannot be decoded without knowledge of the permutations, the keys or key numbers, and the equation used to combine the permutations and keys or key numbers.

Semantic Encoding offers several features that current database security technologies do not:

* While current cryptographic methods are based on mathematical problems that are not yet known to have a solution, Semantic Encoding is based on a mathematical problem that is known to have no solution.

* Semantic Encoding is impervious to cryptographic attack;

* If a non cryptographic method is used in an unauthorized to attempt to decode a semantically encoded database, there is no way for an attacker to tell, by just looking at the result, whether or not the method has succeeded;

* A Semantic Encoding key can be very quickly changed, helping to make key theft useless;

* Separate keys can be assigned to different authorized users;

* The database is split up into components that can easily be distributed on a network;

* The database can be compressed when it is split up, reducing storage and transmission requirements;

These capabilities augment the security of encryption-based methods, and provide the database administrator with new tools to combat key theft and network security problems.

Semantic Encoding can be used with any database application. It can be implemented very easily using SQL, and is very efficient. Large organizations can gain the most from such a general and powerful technology, and we are interested in discussing the joint exploitation of our patents with you.
About the Authors

The inventors of Semantic Encoding are Dr. D. Paul Benjamin, Associate Professor, Pace University, and Dr. Adrian Walker, Reengineering LLC.

Professor Benjamin has held research grants totaling over $1,000,000 from AFOSR and NSF, and is currently the head of the Computer Science Department at Pace University in New York City. Dr. Walker's experience includes research and publication on application enabling software for relational databases at IBM Yorktown, and heading a group of twenty programmers working on B2B supply chain database applications.

Please contact Dr. Adrian Walker to discuss this opportunity.

Adrian Walker
www.reengineeringllc.com
PO Box 1412
Bristol
CT 06011-1412 USA

Phone: USA 860 583 9677
Cell: USA 860 830 2085
Fax: USA 860 314 1029