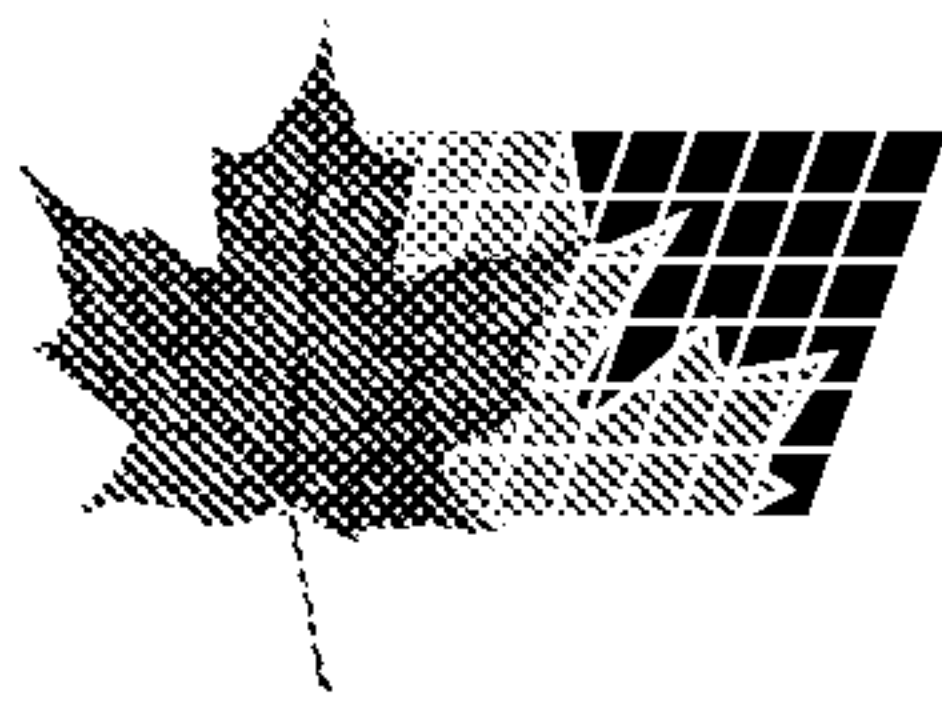




(72) WHITWORTH, Garry David, CA

(72) CHRISTENSEN, Rolf Christopher, CA

(71) 3534421 CANADA CORPORATION, CA

(51) Int.Cl.⁶ G06F 1/14, G06F 9/44

(54) **AUTOMATISATION DE LA RECUPERATION DE DONNEES DE
CONCEPTION ET DE LA MODERNISATION
D'APPLICATIONS PATRIMONIALES**

(54) **AUTOMATION OF THE DESIGN RECOVERY AND FORWARD
ENGINEERING OF LEGACY APPLICATIONS**

(57) A method of recovering design information of a legacy application program comprising translating the legacy application program into tokens representing elements of the program, and automatically rewriting the legacy application program using the tokens to define elements to be included in the rewritten program.

ABSTRACT

A method of recovering design information of a legacy application program comprising translating the legacy application program into tokens representing elements of the program, and automatically rewriting the legacy application program using the tokens to define elements to be included in the rewritten program.

AUTOMATION OF THE DESIGN RECOVERY AND
FORWARD ENGINEERING OF LEGACY APPLICATIONS

FIELD OF THE INVENTION

This invention relates to methods of operating computers, and in particular to a method of recovering design information of a legacy application program.

5 BACKGROUND TO THE INVENTION

As time passes, new computer program languages are created, and old ones fall out of favor. The number of programmers who are sufficiently knowledgeable to maintain an old program gradually decreases. Background
10 documentation concerning the structure of such programs is lost. Indeed, many existing (legacy) programs have been in use for decades, with few, if any programmers left who can modify them, and with little or no documentation which can be used as a base from which modification can be made.

15 The above problems have become very clear with the onset of the "year 2000 problem", by which programs which had been written designating the year using the last two numerical digits instead of four will create errors with the turn of the century. These legacy programs must be modified
20 to correct the year to four digits and to be able to handle four digits, but due to insufficient programmers who know these legacy programs, learning and correction of the programs is generally time consuming and therefore is costly.

25 In order to avoid the above problems, it is sometimes desirable to substitute a new program written in a more modern and widely used language. However, to do this, the business rules processed by the legacy program must be uncovered, and the program completely rewritten in the new
30 language. Both the uncovering and rewriting processes are generally very time consuming and therefore costly.

SUMMARY OF THE INVENTION

The present invention is a method of automating the recovery of design information of a legacy application
35 program, and which can be used to provide a modified representative of the legacy application program in the

previous language or to provide a new application program in a different computer language, or can be used for display of various aspects of the design information of the legacy application. Since the process is automated both in the
5 analysis and program creation, it takes considerably less time and cost than in the past.

Further, documentation of the legacy application, modified legacy application or new application can be automated, thus making analysis of the application easier.

10 In accordance with an embodiment of the present invention, a method of recovering design information of a legacy application program comprises using a common set of tokens to represent corresponding elements of a plurality of different application programs which are in the same or in
15 different languages, and using a set of the tokens relating to a particular legacy application program to display aspects of the legacy application program or to construct another application program in the same or in a different language which is similar to or is modified from the legacy
20 application program.

In accordance with another embodiment, a method of recovering design information of a legacy application program is comprised of translating the legacy application program into tokens representing elements of the program,
25 and automatically rewriting the legacy application program using the tokens to define elements to be included in the rewritten program.

In accordance with another embodiment, a method of recovering design information of a legacy application
30 program comprises:

- (a) parsing an application program to obtain plural parsed program parts,
- (b) storing plural tokens which are common to similar functioning program parts of various computer languages in a
35 database,

(c) mapping the parsed program parts to corresponding ones of the common tokens in the database, and storing the map in the database, and

(d) using the mapped tokens to generate a modified legacy application, a new application in a same language as the legacy application, a new application in a language different from the legacy application, or for display of aspects of the legacy or new application via a user interface.

10 BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the invention will be obtained by a consideration of the detailed description below, in conjunction with the following drawings, in which:

Figure 1 is a block diagram of the preferred embodiment of the invention,

Figure 2 is a block diagram of a preferred embodiment of an engineering interface which is used in the embodiment of Figure 1,

Figure 3 is a representation of a portion of a data model,

Figure 4 is a representation of portions of the data model that represents navigator data structures in more detail, and

Figure 5 is a screenshot of a tree view for the preferred embodiment of the invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

A glossary of terms used in the description below follows:

ANSI	American National Standards Institute
API	Application Programmatic Interface
Application	A series of inter-related programs which, when used together, automate a solution for a business problem

	Application Metadata	Data which describes the objects and methods of an application
5	Borland Interbase	A Commercial Relational Database Management System owned by the company currently known as Borland
	COBOL	Common Business Oriented Language
10	Data Model	A representation of the data structures and relationships between these data structures which describe a business application
15	Data Structure	One or more data elements in a particular data relationship usually used to describe some entity
20	Grammar	A syntactical representation of a programming language
	Legacy Application Program	An older application (usually written in a third generation language which automates a business function or functions
25	LEX	A commercially available software application which is used to tokenize strings in a human readable text file(s) (Lexical Analyzer)
30	Metadata	Data which describes the structures in a database
	ODBC	Open Database Connectivity
	Oracle RDBMS	A Commercial Relational Database Management System owned by the company currently known as Oracle Corporation
35		

	Parser	A computer program which is used to break down the grammar of a particular computer programming language
5	PL/SQL	A computer programming language known as Procedure Language Structures Query Language commercially owned by Oracle Corporation
10	PowerHouse	A fourth generation programming language owned by Cognos Corporation
	RDBMS	Relational Database Management System
15	Relational Database	A data base constructed out of normalized data structures
	Repository	A relational database used to store the metadata and application metadata of a business computer application
20	Reverse Engineering	The process of decomposing the objects and methods which make up a computer program into a format easily understandable by a person
25	SQL	Structures Query Language
	Symbol	A numeric representation of an entity
	Token	An individual object in a repository
30	Transaction	The logical grouping of objections on a database that can be performed or undone against a relational database
35	Tree View	A user interface which operates on the same

		principals as Microsoft Explorer™ interface
5	User Interface	A computer display operated by the user of an application to facilitate the usage of the application
10	Visual Basic	A computer language owned by Microsoft Corporation that is used to build Windows Based Applications
15	YACC	Yet another Compiler Compiler. A program which aides in the development of computer parsers usually coupled with the grammar of a computer language.

Turning to Figure 1, a parser 1A is used to break out the various parts of a legacy application program 3A. Indeed, it is preferred that plural parsers 1A - 1N should be used in the invention, each one corresponding to a different program language, e.g. COBOL, PowerHouse, Interbase, ANSI SQL, etc. Thus programs 3A - 3N written in any source legacy language may be parsed by a corresponding parser.

Each parser scans the legacy application program, and accesses a corresponding knowledge base 5A - 5N specific to its language, which stores representations of every command and option available in the source program language, i.e. its "grammar", to enable it to parse a program having source code written in its language. The legacy code is thus decomposed into individual objects and their associated methods, based on the knowledge base.

The objects, methods and business rules generated by the parser are in the form of standard tokens for all parsers, which can be also considered intermediate files

which are readable by the application interface API, and which has a "front end" 7 which interfaces a preferably relational database 9 (i.e. repository). The repository stores the standard tokens in a record or in a separate
5 database for the particular legacy program which has been parsed. The parsed program parts are thus mapped to corresponding ones of plural common tokens in the repository.

While the above is a preferred way of mapping the
10 program parts to corresponding common tokens, alternatively the parsers can generate files or words which, when applied to the database, are stored as pointers to specific standard tokens which have already been prestored.

It is an important aspect of the invention that
15 common tokens are used for all legacy languages. These common tokens can then be looked up and used for reverse engineering of a legacy program, for documentation, for recreation of the legacy program with modification, and for creation of a new program in the same language as the legacy
20 program or in a different language than the legacy program, to provide the same tasks the legacy application was developed to perform. All of the above programs can be stored in a memory 8 coupled to a processor of a server, for operating the method described herein.

25 An engineering interface 13 stored on another computer (or on the same server) is coupled to the database 9 via a user window 15 provided by the application front end 7. This allows a user to access the records specific to a program, to modify it, to obtain the stored data to provide
30 to another application design program 17 (e.g. Oracle Designer/2000) to create a modified legacy application program or to create a new application program in a different language.

Thus under control of a user, the engineering
35 interface accesses the desired legacy application program records from the database 9 via window 15, sends its tokens

to forward engineering program 17 and commands it to
translate the tokens to a new program in a different
language. It can be instructed by the application designer
to modify all of a particular parameter (such as a two digit
5 date indicator token to four digits), while creating a new
application program using the same design rules indicated by
the common tokens used in the translated legacy application.
The new application program can be in the legacy language,
resulting in the automatic correction of the date in the
10 legacy application program to avoid the "year 2000" problem
described earlier. As the newly reconstituted program is
being created, routines thereof are used by the engineering
interface 13 for checking the recreated tokens against those
designated in the stored database or record relating to the
15 legacy application, so that errors in the recreated program
can be corrected.

Thus the common tokens are used in a relational
database where the tokens represent common elements used in
all or at least a large number of programs. The database 9
20 (repository) memory 8 not only stores the tokens, but also
preferably stores all of the knowledge databases, the
application program interface 5 functions, and the objects
methods and business rules of a legacy or new program.

The application front end can also access the record
25 or database related to a particular legacy program, and
document it, e.g. by providing a display or printout of
menu-related hierarchy of programs, etc.

In respect of parsers relating to particular
languages:

30 Preferably the PowerHouse parser is comprised of
four sub-parsers, including the PowerHouse Dictionary
Language (PDL), the PowerHouse Reporting language (QUIZ),
the PowerHouse batch update language (QTP) and the
PowerHouse Forms Language (QDESIGN). Each of these sub-
35 languages includes a separate knowledge base that is used to
map the PowerHouse language to the API 5.

The COBOL parser should reverse engineer ANSI COBOL code including (but not limited to) the Working Storage, Data Division and Procedure Divisions as well as the application logic coded in the legacy application.

5 For legacy based applications that use Borland Interbase as a database manager, the Borland Interbase GDML Parser is preferred to be used to recover the design from databases using this technology. The parser is capable of parsing all relational objects that can be defined in a
10 Borland Interbase schema including (but not limited to) Relational tables, Relational Fields, Indexes, Index Segments, Triggers, Procedures as well as Database Views.

 For legacy applications that use ANSI SQL data definition language to create their legacy RDBMS, an ANSI
15 SQL Parser should be used to recover the design from databases using this technology. The parser should parse all relational objects that can be defined in a ANSI SQL schema including Relational tables, Relational Fields, Indexes, Index Segments, Triggers, Procedures as well as
20 Database Views.

 The common dialect of the present embodiment is a language referred to herein as the Chameleon API. The primary purpose of each language parser is to read the legacy source application file and to re-write the legacy
25 code into the Chameleon API language. Once this has been completed, the re-written module is then loaded into the repository using the front end (loader) 7.

 The heart of this embodiment is the database 9 repository. This is the area in which all the application
30 and database objects, methods and business rules generated is physically housed.

 The database 9 repository stores information and business rules for business and data services, as well as user interface and appearance control for user services.
35 This environment independent information is preferably stored in an ODBC compliant design database and is intended

to be used when building an application. The use of an open repository enables iterative design and reuse and meets multi-developer requirements. Code is generated from design information in the database 9 repository.

5 The results of the design activities include a data model, business rules and design information and application preferences which are stored in the database 9 repository.

10 A database metadata 19 section of the database 9 repository is used to store the data definition attributes of the legacy application. This area of the database 9 repository preferably stores information about the following objects:

15 Nodes
 Database Roles
 Databases
 Domain Drop Downs
 Domains
 Element Drop Downs
 Element Permissions
 20 Elements
 Index Segments
 Indexes
 Lookup Tables
 Relation Elements
 25 Relation Permissions
 Relations
 Sequences
 Stores Procedures
 Structures
 30 Template Drop Downs
 Templates
 Trigger Elements
 Triggers
 User Roles
 35 Users
 View Sources

An application Metadata 21 section of the repository is used to store the objects, methods and business rules associated with the legacy application. This area of the database 9 repository stores information about the following

5 objects:

	Assignment
	Blob
	Block of Code
	Called Module
10	Node
	Node Element
	Node List
	Node Select
	Node Sort
15	Comment
	Condition
	Condition Detail
	Coordinate
	Core Function
20	Core Function Xref
	Core Message
	Cross Reference
	Cursor
	Cursor Key
25	Database Update
	Event
	Event Detail
	External
	Field Property
30	Field Value
	Function
	Highlight
	Highlight Detail
	Index Segment
35	Legacy Source
	Literal

	Log File
	Lookup
	Message
	Method
5	Module
	Object
	Panel
	Panel Graph
	Passing List
10	Procedure
	Procedure Detail
	Project
	Project Directory
	Receiving List
15	Relation
	Relation Access Field
	Relation Accesses
	Relation Field
	Relation Index
20	Report Section Table
	Report Tables
	Screen Property
	Sort Option
	Stack
25	Stack Detail
	Stored Procedure
	Suminto Operation
	Transaction
	Transaction Relation
30	User
	Variable

35 A knowledge base 23 is a collection of tables that
 are used to document the grammar of the legacy-based
 application. This includes documenting the "command" keyword

and all its options and the mapping of these to the API 5.
This area of the repository should store information about
the following objects:

	API Functions
5	API Parameters
	Element Usages
	Functions
	Function Inputs
	Function Options
10	Function Results
	General Terms
	General Term Keywords
	General Term Keyword Options
	Keywords
15	Keyword Options
	Keyword Option Defaults
	Procedures
	Procedure Options
	Verbs
20	Verb Options
	Table Types
	Table Values

The primary responsibility of the API loader 7 is to
25 load the intermediary files created by the parsers into the
database 9 repository. Each API loader script that is read
and parsed should be checked for correctness and validated
against the data already loaded. If any conflicts arise,
the entire transaction should be rolled back and an error
30 written to a log file. Otherwise, the API loader creates
the appropriate objects for the database 9 repository.

The engineering interface 13 is for use by
application developers to navigate around the various
objects and methods of the application; to report on,
35 analyze as well as to refine the extracted business

information housed by the database 9 repository. A more detailed block diagram of the preferred form of engineering interface is shown in Figure 2.

5 A database navigator 25 is preferably a Microsoft Explorer™ style interface to the objects that make up the data dictionary and physical database of the legacy application. This interface allows the developer to drill to detail on objects such as relational tables, database views, stored procedures as well as triggers. In addition to the
10 navigational capabilities of this interface, it allows the developer to enhance and modify the underlying physical structure of the database by adding, deleting and modifying the attributes that make up the database component of the legacy application.

15 Similar to the database navigator, an application navigator 27 allows the developer to drill down on the functional modules that make up the application including Screens, Reports and Batch Processes. Using this interface the developer, analysts and end users can quickly pinpoint
20 objects located within the module and make any adjustments necessary. This component also preferably includes an editor, which allows you to view the business rules expressed in your application in, for example, a Microsoft style notepad editor.

25 If the legacy application contains a menu style interface, the engineering interface preferably automatically recreates this system to display or print the hierarchy of the application. This allows IT developers and business users to quickly navigate to the module they need
30 to see at the click of a button.

An impact analysis module 29 should provide features that allow assessment of the impacts of making changes to the existing and new application. This modules should include an impact analysis navigator as well as provide a
35 number of standard reports that allow review of the impacts based on any object in the database repository. This should

include for example the ability to assess the impact of removing database fields, the ability to determine any bottlenecks that may be present in the current application, etc.

5 Because the design recovery process has been automated, the speed at which the information is loaded into the repository is only limited by the speed of the hardware on which this invention operates. In addition, because language specific parsers are used, they must understand the
10 complete grammar of the legacy source language in order for the process to be successful. Because of this, all the important objects, methods and business rules coded in the legacy application should be captured.

 A forward engineering application 31 is preferably
15 used to rebuild the legacy application, and forward engineer reports, screens and batch processes to a number of computing platforms including PL/SQL, Visual Basic and Dynamic C modules for batch processes.

 An application designer 17 bridge is preferably also
20 used to load the application directly into Oracle's Designer 2000, once the legacy application has been loaded into the database repository 9.

 All of the elements 25, 27, 29, 31 and 13 interface the application front end 7 and its window 17, to access the
25 database 9 depository.

 The invention can be operated on a server preferably comprised of an IBM compatible computer which uses a Pentium ITM or Pentium IITM processor, to which 128 MB of random access memory is coupled, and to which a hard disk
30 drive (or equivalent) is also coupled comprising at least 1 GB of memory space, and on which the programs Windows 95TM or Windows NTTM and OracleTM 7 or 8 are resident. The repository containing the parsers, knowledge base, API application front end are stored on the hard disk
35 drive. The engineering interface can be operated by a similar computer which is in communication with the server,

the engineering interface being comprised preferably of at least a Pentium I™ or Pentium II™ processor to which 64 MB of random access memory is coupled, and a hard disk drive (or equivalent) with at least 1 GB of memory space. The latter computer preferably also should have Windows 95 or Windows NT 4.0 resident on its hard disk drive. However, other computers and programs can be used, as long as they are capable of carrying out the methods described herein.

EXAMPLE

The following is a detailed example of processing of a typical PowerHouse source language program file. The example provided represents a typical, but simplified version of a PowerHouse QUIZ program.

This example provides four QUIZ commands. The first statement indicated that a table named ORDER in the SALES database is to be read. The second statement is the definition of a variable called D_DISCOUNT that is used to calculate a 7% discount on all orders over \$10,000. The third statement is used to identify the information to be printed on the report while the last statement is used to identify the compiled name of the program.

A. *SAMPLE QUIZ SOURCE CODE*

```

25  ACCESS ORDER IN SALES ALIAS SALES_98
    DEFINE D_DISCOUNT NUMERIC * 7 = &
    (ORDER_AMT * (7/100)) &
    IF ORDER_AMT > 10000 ELSE 0
    REPORT ORDER_ID ORDER_AMT D_DISCOUNT
30  BUILD ORDERREP

```

The following follows the above example through the various processes that have been described above.

B. *KNOWLEDGE BASE DATA/STRUCTURES*

35

Each parser is created preferably using Yet Another Compiler Compiler (YACC) and a Lexical Analyser (Lex). In addition to these two components, the parsers should incorporate a third component, a knowledge base (KB) to store the mapping of the
 5 keywords that make up the various commands in the source language as well as to map to the application programmatic interface (API) functions and parameters that are to be invoked once a grammar rule in YACC has been processed.

10 The following section describes four of the main tables that make up the KB.

B.1 KBKEYWORD

15 The KBKEYWORD table is used to store all the command keywords in the PowerHouse source language. In this example, there are four command keywords present; **ACCESS, DEFINE, REPORT, BUILD.**

20 The KBKEYWORD table is used to store all the keywords available in Powerhouse and is made up of three fields; Yacc_Keyword_Symbol, Keyword_Name and Lookahead_Token. Each of these fields is described as follows:

25 Yacc_Keyword_Symbol: This field stores the actual command symbol used in the YACC grammar specification. These correspond to the %token commands specified in the grammar file.

30 Keyword_Name: This field stores contains the actual string found in the legacy source code of the keyword that will be mapped by the Lexical Analyser (Lex). This field is actually used to lookup the token returned by Lex against
 35 the KBKeyword table. Once a valid match is

encountered, the Yacc_Keyword_Symbol is returned to Yacc by Lex.

- 5 Lookahead_Token: This field stores a symbol which represents whether the next token after the keyword is read is an identifier or not.

Yacc_Keyword_Symbol	Keyword_Name	Lookahead_Token
K_ACCESS	Access	IDENTIFIER

10

B.2 KBKEYWORD OPTION

15 The KBKEYWORD_OPTION table is used to store all the keyword options available in the PowerHouse source language for a particular command keyword. In this example, the **ACCESS** statement contains a keyword option called **ALIAS**.

20 Yacc_Keyword_Symbol: This field stores the actual keyword symbol used in the YACC grammar specification.

25 Yacc_Option_Symbol: This field stores the actual keyword option symbol used in the YACC grammar specifications that represents an option on a command keyword.

30 Keyword_Name: This field contains a string representing the Powerhouse Keyword option that is read by Lex. This field is actually used to lookup the token returned by Lex against the KBKeyword_Option table. Once a valid match is

encountered, the Yacc_Option_Symbol is returned to Yacc by Lex.

Lookahead_Token: This field stores a symbol which represents whether the next token after the keyword option is read is an identifier or not.

Yacc Keyword Symbol	Yacc Option Symbol	Keyword Option	Lookahead Token
K ACCESS	ALIAS	Alias	IDENTIFIER
K ACCESS	ANDX	And	_NOTIDENT
K ACCESS	IN	In	_NOTIDENT
K ACCESS	LINK	Link	_NOTIDENT
K ACCESS	OF	Of	IDENTIFIER
K ACCESS	OPTIONAL	Optional	_NOTIDENT
K ACCESS	RECORD	Record	_NOTIDENT
K ACCESS	TO	To	_NOTIDENT
K ACCESS	VIAINDEX	Viaindex	_NOTIDENT

10

B.3 KBAPI_FUNCTION

The KBAPI_FUNCTION table is used to store a mapping of all the API functions that can be called by the Parser. This table is made up of two fields; Api_Function_Symbol and API_Function_Name. Each of these fields are described as follows:

20 Api_Function_Symbol: This field stores the actual function symbol used in the API specification. This is the actual field that is used to lookup against this table.

Api_Function_Name: This field stores the name of the API function that is to be invoked by the parser once a grammar rule has been processed.

5

Api_Function_Symbol	Api_Function_Name
RPACCESS	iu_rpaccess

10 B.4 KBAPI_PARAMETER

The KBAPI_PARAMETER table is used to store a mapping of all the API parameters that are available for a particular API function. This table is made up of four fields;

15 Api_Function_Symbol, API_Parameter_Symbol,
Api_Parameter_Name1 and Api_Parameter_Name2,

Api_Function_Symbol: This field stores the actual function symbol used in the YACC grammar specification that represents the function that is being specified.

20

Api_Parameter_Symbol: This field stores the actual parameter symbol used in the YACC grammar specification that represents the parameter that is being specified.

25

Api_Parameter_Name1: This field stores the 1st actual parameter name that is to be returned to the YACC grammar.

30

Api_Parameter_Name1: This field stores the 2nd actual parameter name that is to be returned to the YACC grammar.

5

Api Function Symbol	Api Parameter Symbol	Api Parameter Name1	Api Parameter Name2
_RPACCES S	_ACCESS_NAME	p_access_name	p_access_name
_RPACCES S	_ALIAS_NAME	p_alias_name	p_alias_name
_RPACCES S	_AUTOCOMMIT_FLAG	p_autocommit_flag	p_autocommit_flag
_RPACCES S	_AUTOLINK_FLAG	p_autolink_flag	p_autolink_flag
_RPACCES S	_BACKWARDS_FLAG	p_backwards_flag	p_backwards_flag
_RPACCES S	_DATABASE	p_database_name	p_database_name
_RPACCES S	_GENERIC_FLAG	p_generic_flag	p_generic_flag
_RPACCES S	_INDEX_NAME	p_index_name	p_index_name
_RPACCES S	_LINKAGE_TYPE	p_linkage_type_code	p_linkage_type_code
_RPACCES S	_NAME	p_relation_name	p_relation_name
_RPACCES S	_OPTIONAL_FLAG	p_optional_flag	p_optional_flag
_RPACCES S	_ORDER_TYPE_CODE	p_order_type_code	p_order_type_code
_RPACCES S	_ORDERED_FLAG	p_ordered_flag	p_ordered_flag

_RPACCES S	_PHYSICAL_NAME	p_physical_n ame	p_physical_n ame
_RPACCES S	_SEQUENTIAL_FL AG	p_sequential _flag	p_sequential _flag
_RPACCES S	_UNIQUE_FLAG	p_unique_fla g	p_unique_fla g
_RPACCES S	_WARNING_FLAG	p_nowarn_fla g	p_nowarn_fla g

C. LANGUAGE GRAMMAR FOR THE QUIZ ACCESS STATEMENT

The following is a representation of a portion of the
5 grammar that is used to parse and process the ACCESS
statement in the source file of the example given above.

```

K_ACCESS data_structure_ref opt_alias
{
10     keyword = _RPACCESS ;

    /* Don't bother writing out the a iu_rpmodule for the
first */
    /* access statement as this was beformed by the extended
15 */
    /* BEGIN MODULE rule.
*/

    ++ctr.access_id;

20     ctr.block_seq_no = 0 ;
    load_block_id(&object_hdr_item, ++ctr.block_id,
++ctr.block_seq_no) ;

25     iu_rpassess(&object_hdr_item, ctr.access_id,
data_structure_str, $3, NULL, NULL, FALSE,
physical_file_name, access_subfile_flag) ;

```

```

primary_relation = current_ohdr;
current_relation = current_ohdr;

```

```

5   }

```

```

opt_alias:

```

```

        /* empty */          { $$ = NULL ; }
        | ALIAS id            { $$ = $2 ; }
10      ;

```

```

data_structure_ref:

```

```

        table_ref
        {
15          data_structure_str = encode_data_structure($1,
NULL) ;

          access_subfile_flag = FALSE ;
          physical_file_name = NULL ;
          $$ = data_structure_str ;
20        }

        | table_ref IN database_ref
        {
          data_structure_str = encode_data_structure($1,
25      $3) ;

          access_subfile_flag = FALSE ;
          physical_file_name = NULL ;
          $$ = data_structure_str ;

        }
30

        | '*' filespec_ref
        {
          /* Note: physical_file_name set in filespec_ref
rule */
35          data_structure_str = encode_data_structure($2,
NULL) ;

```

```

        access_subfile_flag = TRUE ;
        $$ = data_structure_str ;
    }

5      | owner_ref '.' table_ref
      {
        data_structure_str = encode_data_structure($3,
NULL) ;

        access_subfile_flag = FALSE ;
10     physical_file_name = NULL ;
        $$ = data_structure_str ;
      }

      | owner_ref '.' table_ref IN database_ref
15     {
        data_structure_str = encode_data_structure($3,
$5) ;

        access_subfile_flag = FALSE ;
        physical_file_name = NULL ;
20     $$ = data_structure_str ;
      }
    ;

table_ref:
25     id                                { $$ = $1; }
    ;

database_ref:
    id
30     ;

owner_ref:
    id
    ;

35 id:

```

```

IDENTIFIER      { $$ = $1; }
;

```

D. *'C' API LIBRARY FOR THE IU_RPACCESS FUNCTION*

5 The following represents examples of the the C API library that is called by the YACC grammar specification.

```

/*-----
-----*/
10 /*----- iu_rpassess() -----
-----*/
/*-----
-----*/

15 int iu_rpassess( struct object_hdr *p_object_hdr_item, int
p_access_id, char *p_relation, char *p_alias, char
*p_linkage_type, char *p_optional, int p_autolink, char
*p_physical_name, int p_subfile_flag )
{
20
char access_name[MAXFIELD_SIZE] = "" ;
char database_name[MAXFIELD_SIZE] = "" ;
char relation_name[MAXFIELD_SIZE] = "" ;
char object_name[MAXFIELD_SIZE] = "" ;

25
if (g_debug_switch == 1)
printf("iu_rpassess(p_access_id=%d, p_relation=%s,
p_alias=%s, p_linkage_type=%s, p_optional=%s,
p_autolink=%d, p_physical_name=%s,
30 p_subfile_flag=%d)\n",
p_access_id, p_relation, p_alias, p_linkage_type,
p_optional, p_autolink, p_physical_name,
p_subfile_flag);

35 sprintf(access_name, "ACCESS%d", p_access_id) ;

```

```

        strcpy(relation_name, gettoken(p_relation, '~', _LEFT)) ;

        if ( p_subfile_flag == FALSE )
5      {
            strcpy(database_name, gettoken(gettoken(p_relation, '
', _RIGHT), '~', _LEFT)) ;

            if (strlen(database_name) == 0)
10      {
                /* Check to see if it has already been declared by
looking      */
                /* up on object_hdr
*/
15      strcpy(database_name, lookup_object(relation_name,
_RPRELATION, _DATABASE));

                if (strlen(database_name) == 0)
20      strcpy(database_name, "Non Relational") ;
            }
        }
        else
            strcpy(database_name, "Temporary Data") ;
25

        if (p_alias == NULL)
            strcpy(object_name, gettoken(p_relation, '~', _LEFT))
;
        else
30      strcpy(object_name, gettoken(p_alias, '~', _LEFT)) ;

        load_object_hdr(p_object_hdr_item, _QUIZ, _RPACCESS,
g_project_name, g_module_name,
            database_name, gettoken(p_relation, '~', _LEFT),
35 object_name,
            decode(_OBJECT_TYPE, "rprelrelation"), _RPRELATION) ;

```

```

    push_object_hdr(p_object_hdr_item ) ;

    write_obj_detail(_RACCESS, _ACCESS_NAME, access_name,
5  _QUOTE) ;

    if (p_alias != NULL)
        write_obj_detail(_RACCESS, _ALIAS_NAME, p_alias,
    _QUOTE) ;
10

    if ( p_subfile_flag == TRUE )
    {
        write_obj_detail(_RACCESS, _PHYSICAL_NAME,
p_physical_name , _QUOTE);
15        write_obj_detail(_RPRELATION, _RELATION_TYPE,
            decode(_RELATION_TYPE, "subfile") , _QUOTE);
    }

    if (p_linkage_type != NULL)
20        write_obj_detail(_RACCESS, _LINKAGE_TYPE,
            decode(_LINKAGE_TYPE, p_linkage_type) , _QUOTE);

    if (p_optional != NULL)
        write_obj_detail(_RACCESS, _OPTIONAL_FLAG,
25 decode(_FLAG, p_optional), _QUOTE);

    if (p_autolink == TRUE)
        write_obj_detail(_RACCESS, _AUTOLINK_FLAG,
decode(_FLAG, "TRUE"), _QUOTE);
30

    write_obj_detail(_RPMETHOD, _METHOD_TYPE,
decode(_METHOD_TYPE, "retrieve"), _QUOTE);

    return 1 ;
35
}

```

```

/* -----
-----*/
/* ----- load_object_hdr() -----
5  -----*/
/* -----
-----*/

load_object_hdr(struct object_hdr *p_object_hdr_item,
10      int p_module_type,
      int p_keyword_symbol,
      char *p_project_name,
      char *p_module_name,
      char *p_database_name,
15      char *p_relation_name,
      char *p_object_name,
      char *p_object_type_code,
      int p_object_type)
{
20      register int v_mid;

      if (g_debug_switch == 1)
          printf("load_object_hdr(p_module_type=%d,
p_keyword_symbol=%d, p_project_name=%s,
25      p_module_name=%s, p_database_name=%s,
p_relation_name=%s, p_object_name=%s,
      p_object_type_code=%s, p_object_type=%d)\n",
      p_module_type, p_keyword_symbol, p_project_name,
      p_module_name, p_database_name, p_relation_name,
30      p_object_name,
      p_object_type_code, p_object_type);

      v_mid = get_api_function(p_keyword_symbol);

35      if (g_first_ohdr_created == FALSE)
      {

```

```

    /* create the first object header record. */

    g_object_id = 1;
    g_object_seq_no = 0;
5    ++g_object_hdr_count;
    g_first_ohdr_created = TRUE;
}
else
{
10    g_object_id++;
    g_object_seq_no = 0;
    g_object_hdr_count++;
}

15    store_object_hdr(p_object_hdr_item, _OBJECT_ID, NULL,
g_object_id) ;
    store_object_hdr(p_object_hdr_item, _API_FUNCTION,
api_function[v_mid].api_function_name, -1) ;

20    if (p_module_type == _QUIZ)
    {

        store_object_hdr(p_object_hdr_item, _PROJECT_NAME,
p_project_name, -1) ;
25        store_object_hdr(p_object_hdr_item, _MODULE_NAME,
p_module_name, -1) ;

        if ( gl_file_stack_pointer > 0 )
            store_object_hdr(p_object_hdr_item,
30 _ORIGINATING_MODULE, originating_module, -1) ;

        store_object_hdr(p_object_hdr_item, _DATABASE_NAME,
p_database_name, -1) ;
        store_object_hdr(p_object_hdr_item, _RELATION_NAME,
35 p_relation_name, -1) ;

```

```

        store_object_hdr(p_object_hdr_item, _OBJECT_NAME,
p_object_name, -1) ;
        store_object_hdr(p_object_hdr_item, _OBJECT_TYPE_CODE,
p_object_type_code, -1) ;
5      store_object_hdr(p_object_hdr_item, _OBJECT_TYPE,
NULL, p_object_type) ;
    }

    return 1 ;
10 }

/*-----
-----*/
/*----- store_object_hdr() -----
15 -----*/
/*-----
-----*/

int store_object_hdr(struct object_hdr *p_object_hdr_item,
20 int p_object_hdr_type, char *p_char_value, int p_int_value)

{

    char *object_hdr_value ;
25

    if ( g_debug_switch == 1 )
        printf("store_object_hdr_item(p_object_hdr_type=%d,
p_char_value=%s, p_int_value=%d)\n",
            p_object_hdr_type, p_char_value, p_int_value) ;
30

    if ( p_char_value != NULL )
    {
        object_hdr_value =
malloc(strlen(gettoken(p_char_value, '~', _LEFT)) + 1) ;
35      strcpy(object_hdr_value, gettoken(p_char_value, '~',
_LEFT)) ;

```

```
    }

    switch ( p_object_hdr_type )
    {
5      case _OBJECT_ID:
          p_object_hdr_item->object_id = p_int_value ;
          break ;

          case _API_FUNCTION:
10
              p_object_hdr_item->api_function =
                malloc(strlen(object_hdr_value) + 1) ;
              strcpy(p_object_hdr_item->api_function,
                object_hdr_value) ;
15
              break ;

          case _PROJECT_NAME:
20
              p_object_hdr_item->project_name =
                malloc(strlen(object_hdr_value) + 1) ;
              strcpy(p_object_hdr_item->project_name,
                object_hdr_value) ;
25
              break ;

          case _MODULE_NAME:

              p_object_hdr_item->module_name =
30      malloc(strlen(object_hdr_value) + 1) ;
              strcpy(p_object_hdr_item->module_name,
                object_hdr_value) ;

              break ;
35

          case _ORIGINATING_MODULE:
```

```
        p_object_hdr_item->originating_module =
        malloc(strlen(object_hdr_value) + 1) ;
        strcpy(p_object_hdr_item->originating_module,
5   object_hdr_value) ;

        break ;

        case _OBJECT_NAME:
10
            p_object_hdr_item->object_name =
            malloc(strlen(object_hdr_value) + 1) ;
            strcpy(p_object_hdr_item->object_name,
            object_hdr_value) ;
15
            break ;

            case _OBJECT_TYPE_CODE:

20
                p_object_hdr_item->object_type_code =
                malloc(strlen(object_hdr_value) + 1) ;
                strcpy(p_object_hdr_item->object_type_code,
                object_hdr_value) ;

25
                break ;

                case _DATABASE_NAME:

                    p_object_hdr_item->database_name =
30   malloc(strlen(object_hdr_value) + 1) ;
                    strcpy(p_object_hdr_item->database_name,
                    object_hdr_value) ;

                    break ;

35
                    case _RELATION_NAME:
```

```
        p_object_hdr_item->relation_name =  
        malloc(strlen(object_hdr_value) + 1) ;  
        strcpy(p_object_hdr_item->relation_name,  
5    object_hdr_value) ;  
  
        break ;  
  
    case _OCCURS_VALUE:  
10  
        p_object_hdr_item->occurs =  
        malloc(strlen(object_hdr_value) + 1) ;  
        strcpy(p_object_hdr_item->occurs, object_hdr_value)  
        ;  
15  
        break ;  
  
    case _BLOCK_ID:  
        p_object_hdr_item->block_id = p_int_value ;  
20    break ;  
  
    case _BLOCK_SEQ_NO:  
        p_object_hdr_item->block_seq_no = p_int_value ;  
        break ;  
25  
    case _PANEL_ID:  
        p_object_hdr_item->panel_id = p_int_value ;  
        break ;  
  
30    case _OBJECT_TYPE:  
        p_object_hdr_item->object_type = p_int_value ;  
        break ;  
  
    case _OBJECT_DTL_COUNT:  
35    p_object_hdr_item->object_dtl_count = p_int_value ;  
        break ;
```

```

        case _INCLUDE_MODULE_FLAG:
            p_object_hdr_item->include_module_flag =
p_int_value ;
5            break ;

        default:
            break ;

10    }

    return 1 ;
}

15  /* -----
    -----*/
    /* ----- print_obj_header() -----
    -----*/
    /* -----
20  -----*/

void print_obj_header(char *p_filespec_str)
{
    struct object_hdr *ohdr = first_ohdr;
25    int object_count = 0;
    int obj_type = 0 ;

    char *p_compiled_name ;

30    long time(), tm = time((long *)0);
    char *ctime();

    char block_name[MAXFIELD_SIZE] = "" ;
    char block_seq_no[MAXFIELD_SIZE] = "" ;

35    if (g_debug_switch == 1)

```

```

    printf("print_obj_header()\n");

    if ( p_filespec_str != NULL )
    {
5        if (strlen(gettoken(p_filespec_str, ':', _RIGHT)) ==
0)
        p_compiled_name = p_filespec_str ;
        else
        p_compiled_name = gettoken(p_filespec_str, ':',
10 _RIGHT) ;
    }

    fprintf (coreout, "--\n");
    fprintf (coreout, "-- CORE SOFTWARE CORP. (R) 32-bit
15 PowerHouse %s Parser %s for 80x86 \n",
        core_parser, core_version);
    fprintf (coreout, "-- Copyright (C) CORE SOFTWARE CORP.
1997-1998. Patent Pending.\n");
    fprintf (coreout, "-- All rights reserved.\n");
20    fprintf (coreout, "--\n");
    fprintf (coreout, "-- Automatically generated on: %s",
ctime(&tm));
    fprintf (coreout, "--          ...was generated by:
garrywh\n");
25    fprintf (coreout, "--          ...with version: %s
(%s)\n", core_parser, core_version);
    fprintf (coreout, "--\n\n");

    do
30    {

        ++g_commit_count ;
        ++object_count ;

35    sprintf(block_name, "BLOCK%d", ohdr->block_id) ;
    sprintf(block_seq_no, "%d", ohdr->block_seq_no) ;

```

```

obj_type = ohdr->object_type ;

if (g_commit_count == 1)
5   {
    fprintf(coreout, "Begin\n");
  }

  fprintf(coreout, "  %s\n", ohdr->api_function);
10  fprintf(coreout, "  (\n");

    /* Print the column name and value. Logic added to
control when the */
    /* comma is printed at the end of the line. If at
15  least one detail */
    /* record exists, then print the line, otherwise check
to see if */
    /* the next line is to be printed.
*/
20

    /* always (mandatory) need at least a project name */

    if (ohdr->object_type == _RPPROJECT)
    {
25      /* when printing the project, no comma is required
*/
        if ( ohdr->project_name != NULL )
            fprintf(coreout, "      p_project_name=>'%s'\n",
ohdr->project_name);
30    }

    else
    {
        if ( ohdr->project_name != NULL )
35      fprintf(coreout, "      p_project_name=>'%s',\n",
ohdr->project_name);

```

```

    }

    /* if not the _RPPROJECT record, then module name is
(mandatory) */
5    /* (but not for PDL */

    if (ohdr->object_type != _RPPROJECT)
    {
        if ( ohdr->object_type == _RPMODULE )
10    {
            /* As the API is creating ohdr records, it
stamps the module_name */
            /* with the source file name (i.e., BF013.QZS).
When is comes time */
15    /* to generate the API (after parsing),
determine if a BUILD statement */
            /* was encountered by the parser. If so, the
module_name becomes the */
            /* name (compiled) supplied on the BUILD
20 statement. If the module */
            /* record was created from a BEGIN INCLUDE
statement, then the */
            /* module name is the name of the INCLUDE file
*/
25
            if ( ohdr->include_module_flag == TRUE )
                fprintf(coreout, "
p_module_name=>'%s',\n", ohdr->module_name);
            else
30    fprintf(coreout, "
p_module_name=>'%s',\n", p_compiled_name);
        }
        else
        {
35    if ( p_filespec_str != NULL )

```

```

        fprintf(coreout, "
p_module_name=>'%s',\n", p_compiled_name);
    }
}

5
    if ( ohdr->originating_module != NULL )
    {
        /* print the originating module */
        if (strlen(ohdr->originating_module) > 0)
10        {
            if ((ohdr->object_dtl_count > 0) ||
                (strlen(ohdr->relation_name) > 0))
                fprintf(coreout, "
p_originating_module_name=>'%s',\n", ohdr-
15 >originating_module);
            else
                fprintf(coreout, "
p_originating_module_name=>'%s'\n", ohdr-
>originating_module);
20        }
    }

    /* _RPRPROJECT and _RPMODULE object records cannot
    have any additional properties */
25
    if (ohdr->object_type != _RPPROJECT && ohdr-
>object_type != _RPMODULE)
    {

30        if (strlen(ohdr->object_name) > 0)
        {
            if ((ohdr->object_dtl_count > 0) ||
                (strlen(ohdr->object_type_code) > 0))
                fprintf(coreout, "
35 p_object_name=>'%s',\n", ohdr->object_name);
            else

```

```

        fprintf(coreout, "
p_object_name=>'%s'\n", ohdr->object_name);
    }

5      if (strlen(ohdr->object_type_code) > 0)
    {
        if (ohdr->object_dtl_count > 0)
            fprintf(coreout, "
p_object_type_code=>'%s',\n", ohdr->object_type_code);
10     else
            fprintf(coreout, "
p_object_name=>'%s'\n", ohdr->object_type_code);
    }

15     /* print the database_name property */
    if (strlen(ohdr->database_name) > 0)
    {
        if ((ohdr->object_dtl_count > 0) ||
            (strlen(ohdr->relation_name) > 0))
20         fprintf(coreout, "
p_database_name=>'%s',\n", ohdr->database_name);
        else
            fprintf(coreout, "
p_database_name=>'%s'\n", ohdr->database_name);
25     }

    /* print the relation_name property */
    if (strlen(ohdr->relation_name) > 0)
    {
30         if ((ohdr->object_dtl_count > 0) ||
            (strlen(ohdr->object_name) > 0))
            fprintf(coreout, "
p_relation_name=>'%s',\n", ohdr->relation_name);
        else
35         fprintf(coreout, "
p_relation_name=>'%s'\n", ohdr->relation_name);

```

```

    }

    if ( ohdr->block_id > 0)
    {
5         if ((ohdr->object_dtl_count > 0) ||
(strlen(ohdr->object_type_code) > 0))
            fprintf(coreout, "
p_block_name=>'%s',\n", block_name);
        else
10         fprintf(coreout, "      p_block_name=>'%s'\n",
block_name);
    }

    if ( ohdr->block_seq_no > 0)
15    {
        if ((ohdr->object_dtl_count > 0) ||
(strlen(ohdr->object_type_code) > 0))
            fprintf(coreout, "
p_block_sequence_number=>%s,\n", block_seq_no);
20        else
            fprintf(coreout, "
p_block_sequence_number=>%s\n", block_seq_no);
    }

25    }

    /* now, print the details */
    print_obj_detail(ohdr->firstdtl, ohdr->object_id,
ohdr->object_dtl_count);
30

    fprintf(coreout, "  );\n");
    if (g_commit_count == g_commit_point || ohdr->next ==
NULL)
    {
35        fprintf(coreout, "End;\n");
        fprintf(coreout, " /\n\n");
    }

```

```

        g_commit_count = 0;
    }
    } while ((ohdr = ohdr->next) != NULL);
}

```

5

E. INTERMEDIARY API FILE

Once a parser has successfully processed the source file, the following Intermediary API file is generated for the above example by the parser. This file will later be used by the API Loader described later.

```

Begin
    iu_rpproject
15    (
        p_project_name=>'Sales History'
    );
End;
/
20
Begin
    iu_rpmodule
    (
        p_project_name=>'Sales History',
25    p_module_name=>'ORDERREP',
        p_module_type_code=>'0002',
        p_module_source=>'orderrep.qzs',
        p_module_sequence_number=>1
    );
30 End;
/

Begin
    iu_rpblob
35    (

```

```

    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',
    p_filedir=>'C:\Clients\CSC\RPData',
    p_filename=>'orderrep.qzs'
5      );
End;
/

Begin
10    iu_rpaccess
    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'SALES_98',
15    p_object_type_code=>'0039',
        p_block_name=>'BLOCK1',
        p_block_sequence_number=>1,
        p_database_name=>'SALES',
        p_relation_name=>'ORDER',
20    p_access_name=>'ACCESS1',
        p_alias_name=>'SALES_98',
        p_method_type_code=>'0012'
    );
End;
25 /

Begin
    iu_rpvariable
    (
30    p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'D_DISCOUNT',
        p_object_type_code=>'0054',
        p_block_name=>'BLOCK2',
35    p_block_sequence_number=>1,
        p_variable_name=>'D_DISCOUNT',

```

```

    p_variable_type=>'0013',
    p_item_datatype_code=>'0013',
    p_variable_size=>'7',
    p_signed_flag=>'0002',
5    p_method_type_code=>'0005'
    );
End;
/

10 Begin
    iu_rpprocedure
    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
15    p_object_name=>'PROCEDURE1',
        p_object_type_code=>'0034',
        p_block_name=>'BLOCK2',
        p_block_sequence_number=>2,
        p_variable_name=>'D_DISCOUNT',
20    p_procedure_name=>'PROCEDURE1',
        p_event_flag=>'0002',
        p_source_procedure_type_code=>'0007',
        p_method_type_code=>'0001'
    );
25 End;
/

Begin
    iu_rpevent_detail
30    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'PROCEDURE1',
        p_object_type_code=>'0034',
35    p_block_name=>'BLOCK2',
        p_block_sequence_number=>2,

```

```

    p_procedure_name=>'PROCEDURE1',
    p_event_name=>'EVENT1',
    p_event_sequence_number=>1,
    p_condition_name=>'CONDITION1',
5    p_event_type_code=>'0001',
    p_nesting_level=>1,
    p_token=>'if',
    p_token_type_code=>'0030',
    p_method_type_code=>'0001'
10  );
    End;
    /

```

```

Begin
15    iu_rpcondition_detail
    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'PROCEDURE1',
20    p_object_type_code=>'0034',
        p_block_name=>'BLOCK2',
        p_block_sequence_number=>2,
        p_procedure_name=>'PROCEDURE1',
        p_condition_name=>'CONDITION1',
25    p_condition_sequence_number=>1,
        p_stack_name=>'STACK1',
        p_method_type_code=>'0004'
    );
    End;
30  /

```

```

Begin
    iu_rpstack
    (
35    p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',

```

```

    p_object_name=>'PROCEDURE1',
    p_object_type_code=>'0034',
    p_block_name=>'BLOCK2',
    p_block_sequence_number=>2,
5    p_stack_name=>'STACK1',
    p_method_type_code=>'0001'
    );
End;
/
10
Begin
    iu_rpstack_detail
    (
        p_project_name=>'Sales History',
15    p_module_name=>'ORDERREP',
        p_object_name=>'ORDER_AMT',
        p_object_type_code=>'0040',
        p_block_name=>'BLOCK2',
        p_block_sequence_number=>2,
20    p_element_name=>'ORDER_AMT',
        p_get_relation_flag=>'0001',
        p_stack_name=>'STACK1',
        p_stack_sequence_number=>1,
        p_token=>'ORDER_AMT',
25    p_token_type_code=>'0020',
        p_method_type_code=>'0001'
    );
End;
/
30
Begin
    iu_rpstack_detail
    (
        p_project_name=>'Sales History',
35    p_module_name=>'ORDERREP',
        p_block_name=>'BLOCK2',

```

```

        p_block_sequence_number=>2,
        p_stack_name=>'STACK1',
        p_stack_sequence_number=>2,
        p_token=>'>',
5         p_token_type_code=>'0016',
        p_method_type_code=>'0001'
    );
End;
/
10
Begin
    iu_rpstack_detail
    (
        p_project_name=>'Sales History',
15        p_module_name=>'ORDERREP',
        p_object_name=>'LITERAL1',
        p_object_type_code=>'0022',
        p_block_name=>'BLOCK2',
        p_block_sequence_number=>2,
20        p_literal_name=>'LITERAL1',
        p_text_value=>'10000',
        p_stack_name=>'STACK1',
        p_stack_sequence_number=>3,
        p_token=>'10000',
25        p_token_type_code=>'0019',
        p_method_type_code=>'0001'
    );
End;
/
30
Begin
    iu_rpevent_detail
    (
        p_project_name=>'Sales History',
35        p_module_name=>'ORDERREP',
        p_object_name=>'PROCEDURE1',

```

```

    p_object_type_code=>'0034',
    p_block_name=>'BLOCK3',
    p_block_sequence_number=>2,
    p_procedure_name=>'PROCEDURE1',
5    p_event_name=>'EVENT1',
    p_event_sequence_number=>2,
    p_event_type_code=>'0002',
    p_nesting_level=>1,
    p_related_block_name=>'BLOCK2',
10    p_related_block_sequence_number=>'3',
    p_token=>'then',
    p_token_type_code=>'0031',
    p_method_type_code=>'0001'
    );
15 End;
/

Begin
    iu_rpassignment
20    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'D_DISCOUNT',
        p_object_type_code=>'0054',
25    p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,
        p_variable_name=>'D_DISCOUNT',
        p_assignment_type_code=>'0003',
        p_assignment_name=>'ASSIGN1',
30    p_stack_name=>'STACK2',
        p_method_type_code=>'0002'
    );
End;
/
35
Begin

```

```

iu_rpstack
(
    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',
5    p_object_name=>'D_DISCOUNT',
    p_object_type_code=>'0054',
    p_block_name=>'BLOCK3',
    p_block_sequence_number=>3,
    p_variable_name=>'D_DISCOUNT',
10   p_stack_name=>'STACK2',
    p_method_type_code=>'0001'
);
End;
/
15
Begin
    iu_rpstack_detail
    (
        p_project_name=>'Sales History',
20        p_module_name=>'ORDERREP',
        p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,
        p_stack_name=>'STACK2',
        p_stack_sequence_number=>1,
25        p_token=>'(',
        p_token_type_code=>'0006',
        p_method_type_code=>'0001'
    );
End;
30 /

Begin
    iu_rpstack_detail
    (
35        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',

```

```

        p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,
        p_stack_name=>'STACK2',
        p_stack_sequence_number=>2,
5      p_token=>'(',
        p_token_type_code=>'0006',
        p_method_type_code=>'0001'
    );
End;
10 /

Begin
    iu_rpstack_detail
    (
15      p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'ORDER_AMT',
        p_object_type_code=>'0040',
        p_block_name=>'BLOCK3',
20      p_block_sequence_number=>3,
        p_element_name=>'ORDER_AMT',
        p_get_relation_flag=>'0001',
        p_stack_name=>'STACK2',
        p_stack_sequence_number=>3,
25      p_token=>'ORDER_AMT',
        p_token_type_code=>'0020',
        p_method_type_code=>'0001'
    );
End;
30 /

Begin
    iu_rpstack_detail
    (
35      p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',

```

```

    p_block_name=>'BLOCK3',
    p_block_sequence_number=>3,
    p_stack_name=>'STACK2',
    p_stack_sequence_number=>4,
5    p_token=> '*',
    p_token_type_code=>'0010',
    p_method_type_code=>'0001'
    );
End;
10 /

Begin
    iu_rpstack_detail
    (
15    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',
    p_object_name=>'LITERAL2',
    p_object_type_code=>'0022',
    p_block_name=>'BLOCK3',
20    p_block_sequence_number=>3,
    p_literal_name=>'LITERAL2',
    p_text_value=>'100',
    p_stack_name=>'STACK2',
    p_stack_sequence_number=>5,
25    p_token=>'100',
    p_token_type_code=>'0019',
    p_method_type_code=>'0001'
    );
End;
30 /

Begin
    iu_rpstack_detail
    (
35    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',

```

```

        p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,
        p_stack_name=>'STACK2',
        p_stack_sequence_number=>6,
5      p_token=>')',
        p_token_type_code=>'0007',
        p_method_type_code=>'0001'
      );
End;
10 /

Begin
  iu_rpstack_detail
  (
15    p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,
        p_stack_name=>'STACK2',
20    p_stack_sequence_number=>7,
        p_token=>'/ ',
        p_token_type_code=>'0011',
        p_method_type_code=>'0001'
      );
25 End;
/

Begin
  iu_rpstack_detail
30  (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'LITERAL3',
        p_object_type_code=>'0022',
35    p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,

```

```

        p_literal_name=>'LITERAL3',
        p_text_value=>'100',
        p_stack_name=>'STACK2',
        p_stack_sequence_number=>8,
5      p_token=>'100',
        p_token_type_code=>'0019',
        p_method_type_code=>'0001'
    );
End;
10 /

Begin
    iu_rpstack_detail
    (
15      p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_block_name=>'BLOCK3',
        p_block_sequence_number=>3,
        p_stack_name=>'STACK2',
20      p_stack_sequence_number=>9,
        p_token=>')',
        p_token_type_code=>'0007',
        p_method_type_code=>'0001'
    );
25 End;
/

Begin
    iu_rpevent_detail
30  (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'PROCEDURE1',
        p_object_type_code=>'0034',
35      p_block_name=>'BLOCK4',
        p_block_sequence_number=>4,

```

```

    p_procedure_name=>'PROCEDURE1',
    p_event_name=>'EVENT1',
    p_event_sequence_number=>3,
    p_event_type_code=>'0003',
5    p_nesting_level=>1,
    p_related_block_name=>'BLOCK2',
    p_related_block_sequence_number=>'5',
    p_token=>'else',
    p_token_type_code=>'0032',
10    p_method_type_code=>'0001'
    );
End;
/

15  Begin
    iu_rpassignment
    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'D_DISCOUNT',
20    p_object_type_code=>'0054',
        p_block_name=>'BLOCK4',
        p_block_sequence_number=>5,
        p_variable_name=>'D_DISCOUNT',
        p_assignment_type_code=>'0003',
25    p_assignment_name=>'ASSIGN2',
        p_stack_name=>'STACK3',
        p_method_type_code=>'0002'
    );
30  End;
/

    Begin
        iu_rpstack
35    (
        p_project_name=>'Sales History',

```

```

        p_module_name=>'ORDERREP',
        p_object_name=>'D_DISCOUNT',
        p_object_type_code=>'0054',
        p_block_name=>'BLOCK4',
5      p_block_sequence_number=>5,
        p_variable_name=>'D_DISCOUNT',
        p_stack_name=>'STACK3',
        p_method_type_code=>'0001'
      );
10  End;
    /

    Begin
      iu_rpstack_detail
15    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'LITERAL4',
        p_object_type_code=>'0022',
20      p_block_name=>'BLOCK4',
        p_block_sequence_number=>5,
        p_literal_name=>'LITERAL4',
        p_text_value=>'0',
        p_stack_name=>'STACK3',
25      p_stack_sequence_number=>1,
        p_token=>'0',
        p_token_type_code=>'0019',
        p_method_type_code=>'0001'
      );
30  End;
    /

    Begin
      iu_rppanel
35    (
        p_project_name=>'Sales History',

```

```

    p_module_name=>'ORDERREP',
    p_object_name=>'PANEL1',
    p_object_type_code=>'0030',
    p_block_name=>'BLOCK5',
5    p_block_sequence_number=>1,
    p_panel_name=>'PANEL1',
    p_panel_type_code=>'0011',
    p_method_type_code=>'0001'
);
10 End;
/

Begin
    iu_rpfield_property
15    (
        p_project_name=>'Sales History',
        p_module_name=>'ORDERREP',
        p_object_name=>'ORDER_ID',
        p_object_type_code=>'0040',
20    p_block_name=>'BLOCK5',
        p_block_sequence_number=>2,
        p_element_name=>'ORDER_ID',
        p_get_relation_flag=>'0001',
        p_panel_name=>'PANEL1',
25    p_field_property_name=>'PROPERTY1',
        p_method_type_code=>'0006'
    );
End;
/
30

Begin
    iu_rpcoordinate
    (
        p_project_name=>'Sales History',
35    p_module_name=>'ORDERREP',
        p_object_name=>'PROPERTY1',

```

```

    p_object_type_code=>'0015',
    p_block_name=>'BLOCK5',
    p_block_sequence_number=>2,
    p_coordinate_name=>'COORDINATE1',
5    p_property_name=>'PROPERTY1',
    p_coordinate_type_code=>'0004',
    p_start_row=>1
  );
End;
10 /

Begin
  iu_rpfield_property
  (
15    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',
    p_object_name=>'ORDER_AMT',
    p_object_type_code=>'0040',
    p_block_name=>'BLOCK5',
20    p_block_sequence_number=>3,
    p_element_name=>'ORDER_AMT',
    p_get_relation_flag=>'0001',
    p_panel_name=>'PANEL1',
    p_field_property_name=>'PROPERTY2',
25    p_method_type_code=>'0006'
  );
End;
/

30 Begin
  iu_rpcoordinate
  (
    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',
35    p_object_name=>'PROPERTY2',
    p_object_type_code=>'0015',

```

```

    p_block_name=>'BLOCK5',
    p_block_sequence_number=>3,
    p_coordinate_name=>'COORDINATE2',
    p_property_name=>'PROPERTY2',
5    p_coordinate_type_code=>'0004',
    p_start_row=>1
  );
End;
/
10
Begin
  iu_rpfield_property
  (
    p_project_name=>'Sales History',
15    p_module_name=>'ORDERREP',
    p_object_name=>'D_DISCOUNT',
    p_object_type_code=>'0054',
    p_block_name=>'BLOCK5',
    p_block_sequence_number=>4,
20    p_variable_name=>'D_DISCOUNT',
    p_panel_name=>'PANEL1',
    p_field_property_name=>'PROPERTY3',
    p_method_type_code=>'0006'
  );
25 End;
/

Begin
  iu_rpcoordinate
30  (
    p_project_name=>'Sales History',
    p_module_name=>'ORDERREP',
    p_object_name=>'PROPERTY3',
    p_object_type_code=>'0015',
35    p_block_name=>'BLOCK5',
    p_block_sequence_number=>4,

```

```

        p_coordinate_name=>'COORDINATE3',
        p_property_name=>'PROPERTY3',
        p_coordinate_type_code=>'0004',
        p_start_row=>1
5      );
      End;
      /

```

F. *Data Model for Access & Access Field*

10

Figure 3 shows a representation of the portion of the data model that encompasses the ACCESS and ACCESS_FIELDS relations. These entities in the CORE Repository are used to store the information that will be used to regenerate the

15 data selection criteria for the target environment.

G. *Oracle API Source Code*

The following sample API includes the structure and functionality inherent in the remainder of the APIs. Each

20 API is programmed with the capability to determine if the action to be executed is an Insert action or an Update action. Prior to actually performing the required action, the API's will perform a number of validation steps that establish the relationship between the current Object and

25 the remainder of the objects already stored in the CORE repository.

```

CREATE OR REPLACE PROCEDURE IU_RPAccess (
30      p_Project_Name
          RPPProject.Project_Name%type default null,
      p_Module_Name
          RPModule.Module_Name%type default null,

```

```

        p_Originating_Module_Name      RModule.Module_Name%type
default null,
        p_Database_Name
        DBDatabase.Database_Name%type default null,
5      p_Relation_Name
        DBRelation.Relation_Name%type default null,
        p_Object_Name
        RPObject.Object_Name%type default null,
        p_Object_Type_Code
10      RPObject.Object_Type_Code%type default null,
        p_method_type_code
        RPMethod.Method_Type_code%type default null,
        p_Debug_switch                  Varchar2 default
null,
15      p_Alias_Name
        RPAccess.Alias_Name%type default ' ',
        p_Physical_Name
        DBRelation.Physical_Name%type default null,
        p_Relation_type_code
20      DBRelation.Relation_Type_Code%type default Null ,
        p_Access_Name
        RPAccess.Access_Name%Type default Null,
        p_Stack_name
        RPStack.Stack_Name%Type Default null,
25      p_Index_Name
        RPStack.Stack_name%Type Default null,
        p_Project_ID
        RPAccess.Project_ID%type default null,
        p_Module_ID
30      RPAccess.Module_ID%type default null,
        p_Originating_Module_ID
        RPAccess.Originating_Module_ID%type default null,
        p_Object_ID
        RPAccess.Object_ID%type default null,
35      p_Method_ID
        RPAccess.Method_ID%type default null,

```

```

p_Access_ID
    RPAccess.Access_ID%type default null,
p_Database_ID
    RPAccess.Database_ID%type default Null ,
5  p_Relation_ID
    RPAccess.Relation_ID%type default Null ,
p_Block_ID
    RPAccess.Block_ID%type default Null ,
p_Block_Name
10    RPCODE_Block.Block_Name%type default null,
    p_Block_sequence_Number
    RPMethod.Block_sequence_Number%type default 0,
    p_Linkage_type_code
    RPAccess.Linkage_type_code%type default Null ,
15  p_Backward_Flag
    RPAccess.Backward_Flag%type default Null ,
    p_Generic_Flag
    RPAccess.Generic_Flag%type default Null ,
    p_Optional_Flag
20    RPAccess.Optional_Flag%type default Null ,
    p_Order_Type_Code
    RPAccess.Order_Type_Code%type default Null ,
    p_Sequential_Flag
    RPAccess.Sequential_Flag%type default Null ,
25  p_Unique_Flag
    RPAccess.Unique_Flag%type default Null ,
    p_Autocommit_Flag
    RPAccess.Autocommit_Flag%type default Null ,
    p_Autolink_Flag
30    RPAccess.Autolink_Flag%type default Null ,
    p_Nowarn_Flag
    RPAccess.Nowarn_Flag%type default Null ,
    p_Lock_Type_Code
    RPAccess.Lock_Type_Code%type default Null ,
35  p_Index_ID
    RPAccess.Index_ID%type default Null ,

```

```

p_Stack_ID
    RPAccess.Stack_ID%type default null,
p_Audit_Creation_Date
    RPAccess.Audit_Creation_Date%type default Null ,
5  p_Audit_Modified_Date
    RPAccess.Audit_Modified_Date%type default Null ,
    p_Audit_Update_Type_Code
RPAccess.Audit_Update_Type_Code%type default Null ,
    p_Audit_Who
10    RPAccess.Audit_Who%type default Null ,
    p_Audit_Where
    RPAccess.Audit_Where%type default Null ) IS

t_Project_ID
15    RPAccess.Project_ID%type;
t_Module_ID
    RPAccess.Module_ID%type;
t_Originating_Module_ID
    RPAccess.Originating_Module_ID%type;
20 t_Object_ID
    RPAccess.Object_ID%type;
t_Method_ID
    RPAccess.Method_ID%type;
t_Database_ID
25    RPAccess.Database_ID%type;
t_Relation_ID
    RPAccess.Relation_ID%type;
t_Block_ID
    RPAccess.Block_ID%type default 0 ;
30 t_Procedure_ID
    RPAccess.Block_ID%type default 0 ;
t_Index_ID
    RPAccess.Index_ID%type default 0;
t_Stack_ID
35    RPAccess.Stack_ID%type default 0;

```

```

V_Project_ID
    RPAccess.Project_ID%type;
V_Module_ID
5    RPAccess.Module_ID%type;
    V_Originating_Module_ID
RPAccess.Originating_Module_ID%type;
    V_Object_ID
    RPAccess.Object_ID%type;
10   V_Method_ID
    RPAccess.Method_ID%type;
    V_Block_ID
    RPAccess.Block_ID%type default 0;
    V_Block_sequence_Number
15   RPAccess.Block_Sequence_number%type default 0;
    v_Access_Name
    RPAccess.Access_Name%Type;
    V_Database_ID
    RPAccess.Database_ID%type;
20   V_Relation_ID
    RPAccess.Relation_ID%type;
    V_Access_ID
    RPAccess.Access_ID%type;
    V_Method_Type_Code
25   RPMethod.Method_Type_Code%type;
    V_Alias_Name
    RPAccess.Alias_Name%type;
    v_Linkage_type_code
    RPAccess.Linkage_type_code%type;
30   v_Physical_Name
    DBRelation.Physical_Name%type;
    v_Relation_type_code
    DBRelation.Relation_Type_Code%type;
    V_Backward_Flag
35   RPAccess.Backward_Flag%type;

```

```

V_Generic_Flag
    RPAccess.Generic_Flag%type;
V_Optional_Flag
    RPAccess.Optional_Flag%type;
5  V_Order_Type_Code
    RPAccess.Order_Type_Code%type;
V_Sequential_Flag
    RPAccess.Sequential_Flag%type;
V_Unique_Flag
10    RPAccess.Unique_Flag%type;
V_Autocommit_Flag
    RPAccess.Autocommit_Flag%type;
v_Autolink_Flag
    RPAccess.Autolink_Flag%type default Null;
15 V_Nowarn_Flag
    RPAccess.Nowarn_Flag%type;
V_Lock_Type_Code
    RPAccess.Lock_Type_Code%type;
V_Index_ID
20    RPAccess.Index_ID%type;
v_Stack_ID
    RPAccess.Stack_ID%type default 0;
V_Audit_Creation_Date
    RPAccess.Audit_Creation_Date%type;
25 V_Audit_Modified_Date
    RPAccess.Audit_Modified_Date%type;
V_Audit_Update_Type_Code
    RPAccess.Audit_Update_Type_Code%type;
V_Audit_Who
30    RPAccess.Audit_Who%type;
V_Audit_Where
    RPAccess.Audit_Where%type;

Zero      Number      := 0;
35 Spaces  Varchar(04) := ' ';

BEGIN

```

```

If p_Project_ID is Null or p_project_id = 0
Then
    Null;
else
5     v_Project_id := p_project_id;
End If;

if p_Module_ID is null or p_Module_ID = 0
Then
10    Null;
else
    v_Module_ID := p_Module_ID;
End If;

15    if p_Originating_Module_ID is null or
p_Originating_Module_ID = 0
    Then
        Null;
    else
20        v_Originating_Module_ID :=
p_Originating_Module_ID;
    End If;

if p_Object_ID is null or p_Object_ID = 0
25 Then
    Null;
else
    v_Object_ID := p_Object_ID;
End If;

30    if p_method_ID is null or p_method_ID = 0
    Then
        Null;
    else
35        v_method_ID := p_method_ID;
    End If;
```

```

    if p_database_ID is null or p_database_ID = 0
    Then
        Null;
5      else
        v_database_ID := p_database_ID;
    End If;

    if p_relation_ID is null or p_relation_ID = 0
10   Then
        Null;
    else
        v_relation_ID := p_relation_ID;
    End If;

15   if p_access_ID is null or p_access_ID = 0
    Then
        Null;
    else
20     v_access_ID := p_access_ID;
    End If;

    if p_Block_ID is null or p_Block_ID = 0
    Then
25     Null;
    else
        v_block_ID := p_Block_ID;
    End If;

30   if p_Block_sequence_Number is null or
    p_Block_sequence_Number = 0
    Then
        Null;
    else
35     v_Block_sequence_number :=
    p_Block_sequence_number;
```

```

End If;

if p_Project_name is Null or p_Project_name = spaces
Then
5      Null;
Else
      rp_validate_Project(p_Project_name,
p_Debug_Switch,t_Project_id);
      V_Project_ID := t_Project_ID ;
10     If P_Debug_Switch = 'Y'
      Then
          DBMS_OUTPUT.PUT_LINE('RPACCESS - Project
Validation = ' || t_Project_id);
          End If;
15     End If;

IF p_Module_Name is Null or p_module_name = spaces
Then
      Null;
20     else
          rp_validate_Module(t_project_id, p_Module_name,
p_Debug_Switch, t_Module_id);
          V_Module_ID := t_Module_ID ;
          If P_Debug_Switch = 'Y'
25         Then
              DBMS_OUTPUT.PUT_LINE('RPACCESS - Module
Validation = ' || t_Module_id);
              End if;
          End IF;
30

IF p_Originating_Module_Name is Null and p_module_name
<> spaces
Then
      rp_validate_Module(t_project_id, p_Module_name,
35 p_Debug_Switch, t_Originating_Module_id);

```

```

        V_Originating_Module_ID      := t_Originating_Module_ID
;
        If P_Debug_Switch = 'Y'
        Then
5           DBMS_OUTPUT.PUT_LINE('RPACCESS- Originating
Module Validation = ' || t_Originating_module_id);
        End If;
        else
            rp_validate_Module(t_project_id,
10 p_Originating_Module_name, p_Debug_Switch,
t_Originating_Module_id);
            V_Originating_Module_ID      := t_Originating_Module_ID
;
            If P_Debug_Switch = 'Y'
15            Then
                DBMS_OUTPUT.PUT_LINE('RPACCESS- Originating
Module Validation = ' || t_Originating_module_id);
                End If;
            END If;
20
            If p_object_name is null or p_object_name = spaces
            Then
                null;
            else
25            rp_validate_Object(t_project_id, t_Module_id,
t_Originating_Module_id, p_object_name, P_Object_type_code,
p_Debug_Switch,t_Object_id);
                V_Object_ID              := t_Object_ID ;
                If P_Debug_Switch = 'Y'
30            Then
                    DBMS_OUTPUT.PUT_LINE('RPAccess - Object
Validation = ' || t_object_id);
                    End if;
                End If;
35

```

```

        If p_Method_Type_code is null or p_method_type_code =
spaces
        Then
            null;
5         else
            rp_validate_Method(t_project_id, t_Module_id,
t_Originating_Module_id, t_object_Id, P_Method_type_code,
p_Debug_Switch , p_Block_Name,
p_Block_sequence_number,t_Method_id);
10         V_Method_ID                := t_Method_ID ;
            If P_Debug_Switch = 'Y'
            Then
                DBMS_OUTPUT.PUT_LINE('RPACCESS - Method
Validation = ' || t_method_id);
15         End if;
            End If;

        If p_Block_name is null
20         Then
            v_Block_id := 0;
        Else
            rp_validate_Code_Block(t_project_id, t_Module_id,
t_Originating_Module_id, p_Block_name, p_Debug_Switch,
25 t_Block_id, t_Procedure_id);
            V_Block_ID                := t_Block_ID ;
            If P_Debug_Switch = 'Y'
            Then
                DBMS_OUTPUT.PUT_LINE('RPAccess - Validate Code
30 Block = ' || t_Block_id);
            End If;
        End If;

        If p_Database_name is null or p_database_name = spaces
35         Then
            null;

```

```

        else
            DB_validate_Database(p_database_name,
p_Debug_Switch,t_database_id);
            V_Database_ID                := t_database_ID ;
5          If P_Debug_Switch = 'Y'
            Then
                DBMS_OUTPUT.PUT_LINE('RPACCESS - Database = ' ||
t_database_id);
                End If;
10         End IF;

        If p_Relation_name is null or p_Relation_name = spaces
        Then
            null;
15        else
            DB_validate_Relation(t_Database_id,p_Relation_name,
p_Debug_Switch,t_Module_id,t_Relation_id);
            v_relation_id                := t_relation_id;
            If P_Debug_Switch = 'Y'
20          Then
                DBMS_OUTPUT.PUT_LINE('RPAccess - DBRelation
Validation = ' || t_relation_id);
                End If;
                If V_Relation_id = 0
25          then
                    RP_validate_Relation(t_Project_id, t_Module_id,
t_originating_module_id, p_relation_name,
p_Debug_Switch,t_Relation_id);
                    v_relation_id                := t_relation_id;
30          If P_Debug_Switch = 'Y'
                Then
                    DBMS_OUTPUT.PUT_LINE('RPAccess - RPRelation
Validation = ' || t_relation_id);
                    End If;
35          End If;
        End If;

```

```

    If P_Stack_name is Null or p_Stack_Name = ' '
    Then
        Null;
5      else
        rp_validate_Stack(t_project_id, t_Module_id,
t_Originating_Module_id, P_Stack_Name, p_Debug_Switch,
t_Stack_id);
        v_Stack_id      := t_Stack_id;
10     If P_Debug_Switch = 'Y'
        Then
            DBMS_OUTPUT.PUT_LINE('RPAccess - Get Stack = '
|| t_Stack_id);
            End If;
15     End If;

    If P_Index_name is Null or p_Index_Name = ' '
    Then
        Null;
20     else
        rp_validate_Relation_Index(t_project_id,
t_Module_id, t_Originating_Module_id, P_Index_Name,
p_Debug_Switch, t_Index_id);
        v_Index_id      := t_Index_id;
25     If P_Debug_Switch = 'Y'
        Then
            DBMS_OUTPUT.PUT_LINE('RPAccess - Get Index = '
|| t_Index_id);
            End If;
30     End If;

    V_access_Name := p_access_Name;

    SELECT      Project_ID,
35             Module_ID,
              Originating_Module_ID,

```

	Object_ID,
	Method_ID,
	Access_ID,
	Block_ID,
5	Block_Sequence_Number,
	Access_Name,
	Database_ID,
	Relation_ID,
	Alias_Name,
10	Index_ID,
	Stack_ID,
	linkage_Type_Code,
	Backward_Flag,
	Generic_Flag,
15	Optional_Flag,
	Order_Type_Code,
	Sequential_Flag,
	Unique_Flag,
	Autocommit_Flag,
20	Autolink_Flag,
	Nowarn_Flag,
	Lock_Type_Code,
	Audit_Creation_Date,
	Audit_Modified_Date,
25	Audit_Update_Type_Code,
	Audit_Who,
	Audit_Where
	INTO v_Project_ID,
	v_Module_ID,
30	v_Originating_Module_ID,
	v_Object_ID,
	v_Method_ID,
	v_Access_ID,
	v_Block_ID,
35	v_Block_Sequence_Number,
	v_Access_Name,

```

        v_Database_ID,
        v_Relation_ID,
        v_Alias_Name,
        v_Index_ID,
5      v_Stack_ID,
        v_linkage_Type_Code,
        v_Backward_Flag,
        v_Generic_Flag,
        v_Optional_Flag,
10     v_Order_Type_Code,
        v_Sequential_Flag,
        v_Unique_Flag,
        v_Autocommit_Flag,
        v_Autolink_Flag,
15     v_Nowarn_Flag,
        v_Lock_Type_Code,
        v_Audit_Creation_Date,
        v_Audit_Modified_Date,
        v_Audit_Update_Type_Code,
20     v_Audit_Who,
        v_Audit_Where

FROM    RPAccess

WHERE    Project_ID           = t_Project_ID and
        Module_ID            = t_Module_ID and
25     Originating_Module_ID =
t_Originating_Module_ID and
        Object_ID             = t_Object_ID and
        Method_ID             = t_Method_ID and
        Block_id              = v_Block_id and
30     Block_Sequence_number =
p_Block_sequence_Number and
        access_name           = p_Access_Name;

35     IF P_Alias_Name is null
        Then

```

```
        null;
    Else
        v_Alias_Name := p_Alias_Name;
    End If;

5
    IF P_Linkage_Type_code is null
    Then
        null;
    Else
10        v_Linkage_Type_Code := p_Linkage_Type_Code;
    End If;

    IF P_Backward_Flag is null
    Then
15        null;
    Else
        v_Backward_Flag := p_Backward_Flag;
    End If;
    IF P_Generic_Flag is null
20    Then
        null;
    Else
        v_Generic_Flag := p_Generic_Flag;
    End If;
25    IF P_Optional_Flag is null
    Then
        null;
    Else
        v_Optional_Flag := p_Optional_Flag;
30    End If;
    IF P_Order_Type_Code is null
    Then
        null;
    Else
35        v_Order_Type_Code := p_Order_Type_Code;
    End If;
```

```
IF P_Sequential_Flag is null
Then
    null;
Else
5    v_Sequential_Flag := p_Sequential_Flag;
End If;
IF P_Unique_Flag is null
Then
    null;
10 Else
    v_Unique_Flag := p_Unique_Flag;
End If;
IF P_Autocommit_Flag is null
Then
15    null;
Else
    v_Autocommit_Flag := p_Autocommit_Flag;
End If;
IF P_Autolink_Flag is null
20 Then
    null;
Else
    v_Autolink_Flag := p_Autolink_Flag;
End If;
25 IF P_Nowarn_Flag is null
Then
    null;
Else
    v_Nowarn_Flag := p_Nowarn_Flag;
30 End If;
IF P_Lock_Type_Code is null
Then
    null;
Else
35    v_Lock_Type_Code := p_Lock_Type_Code;
End If;
```

```
IF P_Index_ID is null
Then
    null;
5 Else
    v_Index_ID := p_Index_ID;
End If;

IF P_Stack_ID is null
10 Then
    null;
Else
    v_stack_ID := p_stack_ID;
End If;

15 IF P_Audit_Creation_Date is null
Then
    null;
Else
20 v_Audit_Creation_Date := p_Audit_Creation_Date;
End If;
IF P_Audit_Modified_Date is null
Then
    null;
25 Else
    v_Audit_Modified_Date := p_Audit_Modified_Date;
End If;
IF P_Audit_Update_Type_Code is null
Then
30 null;
Else
    v_Audit_Update_Type_Code :=
p_Audit_Update_Type_Code;
End If;
35 IF P_Audit_Who is null
Then
```

```

        null;
    Else
        v_Audit_Who := p_Audit_Who;
    End If;
5   IF P_Audit_Where is null
    Then
        null;
    Else
        v_Audit_Where := p_Audit_Where;
10  End If;

```

```

UPDATE          RPACCESS

```

```

SET

```

```

15      Access_Name = v_Access_Name,
        Database_ID = v_Database_ID ,
        Relation_ID = v_Relation_ID ,
        Alias_Name  = v_Alias_Name ,
        Index_ID    = v_Index_ID,
20      Stack_ID    = v_Stack_ID,
        Linkage_type_code = v_Linkage_Type_Code,
        Backward_Flag = v_Backward_Flag ,
        Generic_Flag = v_Generic_Flag ,
        Optional_Flag = v_Optional_Flag ,
25      Order_Type_Code = v_Order_Type_Code ,
        Sequential_Flag = v_Sequential_Flag ,
        Unique_Flag = v_Unique_Flag ,
        Autocommit_Flag = v_Autocommit_Flag ,
        Autolink_Flag = v_Autolink_Flag ,
30      Nowarn_Flag = v_Nowarn_Flag ,
        Lock_Type_Code = v_Lock_Type_Code ,
        Audit_Creation_Date = v_Audit_Creation_Date ,
        Audit_Modified_Date = v_Audit_Modified_Date ,
        Audit_Update_Type_Code = v_Audit_Update_Type_Code
35      ,

        Audit_Who = v_Audit_Who ,

```

```

        Audit_Where = v_Audit_Where
WHERE
        Access_ID           = v_Access_ID and
        Object_ID           = v_Object_ID and
5       Method_ID          = v_Method_ID and
        Project_ID          = v_Project_ID and
        Originating_Module_ID = v-Originating_Module_ID and
        Module_ID           = v_Module_ID and
        Block_id            = v_Block_id and
10      Block_Sequence_number = p_Block_sequence_Number ;

EXCEPTION
WHEN NO_DATA_FOUND THEN
    If v_Object_id = 0 or
15     v_Object_id is null
    Then
        IU_RPObject
        (
            p_Project_name,
            p_Module_Name,
20     p_originating_Module_name,
            p_Object_Name,
            p_object_Type_code,
            p_debug_switch
25     );
        rp_validate_Object(t_project_id, t_Module_id,
t_Originating_Module_id, p_object_name, P_Object_type_code,
p_Debug_Switch,t_Object_id);
        V_Object_ID           := t_Object_ID ;
30     If P_Debug_Switch = 'Y'
        Then
            DBMS_OUTPUT.PUT_LINE('RPAccess - Insert Phase Object
Validation = ' || v_object_id);
            End If;
35     End If;

```

```

If v_Method_id = 0 or
  v_Method_id is null
Then
  IU_RPMethod
5      (
      p_Project_name,
      p_Module_Name,
      p_originating_Module_name,
      p_Object_Name,
10     p_object_Type_code,
      p_Method_Type_code,
      p_debug_switch,
      p_Block_Name,
      p_Block_sequence_number
15     );
      rp_validate_Method(t_project_id, t_Module_id,
t_Originating_Module_id, t_object_Id, P_Method_type_code,
p_Debug_Switch, p_Block_Name,
p_Block_sequence_Number,t_Method_id);
20     V_Method_ID := t_Method_ID ;
      If P_Debug_Switch = 'Y'
      Then
          DBMS_OUTPUT.PUT_LINE('RPACCESS - Insert Phase Method
Validation = ' || v_method_id);
25     End If;
      End If;

      If v_Database_id = 0 or v_Database_id is null
      Then
30     IU_DBDatabase
      (
          p_Database_name,
          p_debug_switch
      );
35     DB_validate_Database(p_database_name,
p_Debug_Switch,t_database_id);

```

```

V_Database_ID                := t_database_ID ;
If P_Debug_Switch = 'Y'
Then
    DBMS_OUTPUT.PUT_LINE('RPACCESS - Database = ' ||
5  t_database_id);
    End IF;
End If;

If P_Relation_name <> ' '
10  then
    IU_RPRelation
    (
        p_Project_name,
        p_Module_Name,
15  p_originating_Module_name,
        p_Object_Name,
        p_object_type_code,
        p_method_type_code,
        p_Database_Name,
20  p_Relation_Name,
        p_Alias_name,
        p_debug_switch,
        P_Physical_name,
        p_Relation_Type_Code
25  );
    DB_validate_Relation(t_Database_id, p_relation_name,
p_Debug_Switch,t_Module_id,t_Relation_id);
    v_relation_id                := t_relation_id;
    If P_Debug_Switch = 'Y'
30  Then
        DBMS_OUTPUT.PUT_LINE('RPAccess - RPRelation
Validation = ' || t_relation_id);
        End If;
    End If;
35
    IF P_Alias_name is null

```

```
Then
    v_Alias_Name := Spaces;
Else
    v_Alias_Name := p_Alias_name;
5 End If;

IF P_Linkage_Type_Code is null
Then
    v_Linkage_Type_Code := Spaces;
10 Else
    v_Linkage_Type_Code := p_Linkage_Type_Code;
End If;

IF P_Backward_Flag is null
15 Then
    v_Backward_Flag := Spaces;
Else
    v_Backward_Flag := p_Backward_Flag;
End If;

20 IF P_Generic_Flag is null
Then
    v_Generic_Flag := Spaces;
Else
25 v_Generic_Flag := p_Generic_Flag;
End If;

IF P_Optional_Flag is null
Then
30 v_Optional_Flag := Spaces;
Else
    v_Optional_Flag := p_Optional_Flag;
End If;

35 IF P_Order_Type_Code is null
Then
```

```
        v_Order_Type_Code := Spaces;
Else
        v_Order_Type_Code := p_Order_Type_Code;
End If;

5
IF P_Sequential_Flag is null
Then
        v_Sequential_Flag := Spaces;
Else
10        v_Sequential_Flag := p_Sequential_Flag;
End If;

IF P_Unique_Flag is null
Then
15        v_Unique_Flag := Spaces;
Else
        v_Unique_Flag := p_Unique_Flag;
End If;

20
IF P_Autocommit_Flag is null
Then
        v_Autocommit_Flag := Spaces;
Else
        v_Autocommit_Flag := p_Autocommit_Flag;
25
End If;

IF P_Autolink_Flag is null
Then
        v_Autolink_Flag := Spaces;
30
Else
        v_Autolink_Flag := p_Autolink_Flag;
End If;

IF P_Nowarn_Flag is null
35
Then
        v_Nowarn_Flag := Spaces;
```

```
Else
    v_Nowarn_Flag := p_Nowarn_Flag;
End If;

5  IF P_Lock_Type_Code is null
    Then
        v_Lock_Type_Code := Spaces;
    Else
        v_Lock_Type_Code := p_Lock_Type_Code;
10 End If;

    IF P_Index_ID is null
    Then
        v_Index_ID := Zero;
15 Else
        v_Index_ID := p_Index_ID;
    End If;

    IF P_Stack_ID is null
20 Then
        v_Stack_ID := Zero;
    Else
        v_Stack_ID := p_Stack_ID;
    End If;

25 IF P_Audit_Creation_Date is null
    Then
        v_Audit_Creation_Date := Sysdate;
    Else
30 v_Audit_Creation_Date := p_Audit_Creation_Date;
    End If;

    IF P_Audit_Modified_Date is null
    Then
35 v_Audit_Modified_Date := Sysdate;
    Else
```

```

        v_Audit_Modified_Date := p_Audit_Modified_Date;
End If;

```

```

        IF P_Audit_Update_Type_Code is null
5      Then
            v_Audit_Update_Type_Code := Spaces;
        Else
            v_Audit_Update_Type_Code :=
10      p_Audit_Update_Type_Code;
        End If;

```

```

        IF P_Audit_Who is null
        Then
            v_Audit_Who := Spaces;
15      Else
            v_Audit_Who := p_Audit_Who;
        End If;

```

```

        IF P_Audit_Where is null
20      Then
            v_Audit_Where := Spaces;
        Else
            v_Audit_Where := p_Audit_Where;
        End If;
25

```

```

INSERT INTO RPAccess
        VALUES (v_Project_ID,
30      v_Module_ID,
            v_Originating_Module_ID,
            v_Object_ID,
            v_Method_ID,
            RPAccess_IDS.NEXTVAL,
            v_block_id,
35      v_block_sequence_number,
            v_Access_Name,

```

```

        v_Database_ID,
        v_Relation_ID,
        v_Alias_Name,
        v_Index_ID,
5      v_stack_ID,
        v_Linkage_Type_Code,
        v_Backward_Flag,
        v_Generic_Flag,
        v_Optional_Flag,
10     v_Order_Type_Code,
        v_Sequential_Flag,
        v_Unique_Flag,
        v_Autocommit_Flag,
        v_Autolink_Flag,
15     v_Nowarn_Flag,
        v_Lock_Type_Code,
        v_Audit_Creation_Date,
        v_Audit_Modified_Date,
        v_Audit_Update_Type_Code,
20     v_Audit_Who,
        v_Audit_Where);

WHEN OTHERS THEN
    /* Handler which executes for all other errors. */
    v_ErrorCode := SQLCODE;
25     v_ErrorText := SUBSTR(SQLERRM, 1, 200);
    INSERT INTO log_table (code, message, api, info) VALUES
        (v_ErrorCode, v_ErrorText, 'RPAccess', `Oracle error
occurred`);
END IU_RPAccess;
30 /

```

H. Data Model structures for the Dynamic Tree view Navigators

Figure 4 shows a representation of structures represented in the Data Model which form the cornerstone on which the user interface behavior is controlled and managed. For each required node on the tree view, a node and the associated information required to determine the data to be displayed and defined and stored in the structures represented in this figure.

I. Visual Basic Dynamic Tree View Navigator

Figure 5 is a screenshot of a tree view. The code required to create this tree view dynamically (data driven) is illustrated in the figure.

The following code dynamically creates the tree view for the preferred embodiment. This code determines what information is to be displayed in the tree view. It references the "**CreateNode**" procedure (in order to create each node in the tree view) which is demonstrated in the description is following this procedure.

20

```
Public Sub LoadNode(ByVal strKey As String, ByVal
strParentRelation As String)
```

25

```
'-----
'
' This procedure loads the node and adds 2 images into the
' imagelist to be displayed
' in the treeview.
'
' This procedure accepts the following parameters:
'
'   strKey - The key of the parent node. (Ie.
'>NODE>3>Project_Id>1")
```

```

'   strParentRelation - The relation from which the parent
information was retrieved.
'
'   (Ie. The parent relation is
RPPProject when clicking a Module)
5  '-----
-----

    Dim strSQL As String
    Dim intCol As Integer
10  Dim lngId As Long
    Dim imgImage As ListImage
    Dim strRefElement As String
    Dim strRefElements() As typElementInfo
    Dim intElementCount As Integer
15  Dim strSortElement As String
    Dim strSortElements As String
    Dim rsResults As rdoResultset '
Resultset for the reference table info.
    Dim rsNodes As rdoResultset '
20 Resultset for the node table info.
    Dim strRPNodeKey As String ' Stores
the primary key info.
    Dim strOpenPicture As String ' Stores
the location of the open icon.
25  Dim strClosedPicture As String ' Stores
the location of the closed icon.
    Dim strReferenceRelation As String ' Stores
the reference relation.
    Dim lngParentID As Long ' Stores
30 the parent node id.
    Dim strName As String ' Stores
the name to display for the node.
    Dim lngParentNodeID As Long ' Stores
the node's parent id.
35  Dim lngNodeId As Long ' Stores
the node's id.

```

```

    Dim strSourceRelation As String                                ' Stores
the relation to fetch the source info.

    Dim strPrimaryKey As String                                    ' Stores
the elements that make up the PK.
5    Dim strRelations As String                                    ' Stores
the concatenated source and reference relations.
    Dim lngLastNodeId As Long                                      ' Stores
the previous node viewed.
    Dim blnNodeIdChanged As Boolean                                ' Stores
10  if the node has changed.
    Dim strWhereClause As String                                  ' Stores
the WHERE clause.
    Dim strWhereClauseFields As String                            ' Fields
that make up the where clause.
15  Dim intPos As Integer
    Dim intArrayCount As Integer
    Dim blnAddToArray As Integer
    Dim strPK As String
    Dim strSeperator As String
20  Dim blnDistinct As Boolean
    Dim strNodeName As String
    Dim strProcDir As String    ' Delete
    Dim strProcFile As String
    Dim strProcedure As String
25  Dim strSortType As String
    Dim strOperators As String

    On Error GoTo ADErr

30  ' If this is a root node, clear the treeview.
    If Len(strKey) = 0 Then
        tvwTables.Nodes.Clear
        lngParentNodeID = ROOT_NODE
    Else
35  lngParentNodeID = GetValue(">NODE>", strKey)
    End If

```

```

' Check if a connection to the database exists.
If gobjDBConnect.rdoDB Is Nothing Then Exit Sub

5   Screen.MousePointer = vbHourglass

' Reset this value to false.
pblnCheckedNodeCount = False

10  ' Select the Node information from the RPNodes table.
    strSQL = "SELECT Reference_Relation, Reference_Element,"
    strSQL = strSQL & " Node_Id, Open_Picture, Closed_Picture,
Sort_Element,"
    strSQL = strSQL & " Node_Relation, Node_Element,
15  Node_Display_Description,"
    strSQL = strSQL & " Parent_Relation, Parent_Element,
Value, Seperator, Static_Flag,"
    strSQL = strSQL & " Element_Name, Source_Relation,
Documentation_Flag,"
20  strSQL = strSQL & " Display_Flag, Distinct_Flag,
Relation_Seperator, Operator,"
    strSQL = strSQL & " Node_List_Relation, Node_List_XRef,
Node_Name,"
    strSQL = strSQL & " External_Directory, External_File,
25  Procedure_To_Call"
    strSQL = strSQL & " FROM RPNode_Select_Reference_View"
    strSQL = strSQL & " WHERE Parent_Node_Id = " &
lngParentNodeID
    strSQL = strSQL & " AND Tree_Group_Code = " &
30  StrToFld(pstrTreeType)
    strSQL = strSQL & " ORDER BY Sequence_Number,
Distinct_Flag, Sort_Sequence, Node_Sequence"

    Set rsNodes = gobjDBConnect.rdoDB.OpenResultset(strSQL,
35  rdOpenForwardOnly, rdConcurReadOnly, rdExecDirect)
    strRPNodeKey = ""

```

```

    Do While Not rsNodes.EOF
Continuation_Point:      ' If the node changed (meaning
display static data), return here.

5      ' Set this value to an empty string so that we don't
pass the incorrect
      ' primary key. Reset the NodeId to FALSE.
      strPrimaryKey = ""
10     strRPNodeKey = ""
      strWhereClauseFields = ""
      strSortElements = ""
      Erase strRefElements()
      blnNodeIdChanged = False

15     ' Store the values.
      lngNodeId = rsNodes("Node_Id").Value
      strOpenPicture = rsNodes("Open_Picture").Value & ""
      strClosedPicture = rsNodes("Closed_Picture").Value & ""
20     strNodeName = rsNodes("Node_Name").Value & ""

      ' If the Static_Flag = '0001', then we are displaying
SELECT, VARIABLES, LITERALS, etc.
      If rsNodes("Static_Flag").Value & "" = YES_STR Then

25         If lngLastNodeId <> rsNodes("Node_Id") Then
            lngLastNodeId = rsNodes("Node_Id") & ""

            strReferenceRelation =
30     rsNodes("Node_Display_Description").Value & ""
            ' If we haven't checked the node count, do it now.
            If pblnCheckedNodeCount = False Then
                If Len(Trim$(rsNodes("Node_List_Relation") & ""))

Then

```

```

        Call CheckNodeCount(strKey,
rsNodes("Node_List_Relation") & "",
rsNodes("Node_List_XRef") & "")
        End If
5      End If

        ' Used for editable static nodes, such as Original
Sources.
        ' This allows us to edit the source table which
10    contains the BLOB.
        strSourceRelation = rsNodes("Source_Relation").Value
& ""

        ' Check if this node is to be displayed.
15    If DisplayNode(rsNodes("Node_List_XRef").Value & "")
Then
        On Error Resume Next
        ' Load the imagelist with the new items.
        Set imgImage = imlTreePics.ListImages.Add(,
20    strNodeName & "_Closed", LoadPicture(strClosedPicture))
        Set imgImage = imlTreePics.ListImages.Add(,
strNodeName & "_Open", LoadPicture(strOpenPicture))
        strRelations = strSourceRelation & ";" &
strReferenceRelation
25    ' Create the node.
        strWhereClauseFields = strWhereClauseFields & "<"
& strOperators
        Call CreateNode(strKey, lngNodeId, strRelations,
rsNodes("Node_Display_Description").Value, strPrimaryKey,
30    strWhereClauseFields, strNodeName)
        End If
    End If

Else
35    ' Store the relevant information for the nodes.

```

```

        strReferenceRelation =
rsNodes("Reference_Relation").Value & ""
        strSourceRelation = rsNodes("Source_Relation").Value &
""
5
        lngLastNodeId = lngNodeId
        strWhereClauseFields = ""
        ' Fetch all of the element_names that make up the
primary key.
10        ' Exit the loop if a new Node_Id is found.
        Do While Not rsNodes.EOF
            If lngLastNodeId <> rsNodes("Node_Id").Value Then
                ' Node Id has changed.
                blnNodeIdChanged = True
15                Exit Do
            Else
                ' Keep looping to fetch the necessary values.
                lngLastNodeId = rsNodes("Node_Id").Value
                strRefElement = rsNodes("Reference_Element").Value
20                & ""
                strSortElement = rsNodes("Sort_Element").Value &
                ""

                ' Check if the array or the string is empty.
25                On Error Resume Next
                If UBound(strRefElements) < 0 And
Len(Trim$(strRefElement)) > 0 Then
                    If Err.Number = 9 Then                ' Subscript out of
range.
30                    Err.Clear
                    End If
                    ReDim Preserve strRefElements(0) As
typElementInfo
                    strRefElements(0).strElement = strRefElement
35                    strRefElements(0).strSeperator =
rsNodes("Seperator").Value & ""

```

```

        strRefElements(0).strDistinctFlag =
rsNodes("Distinct_Flag").Value & ""
        strRefElements(0).strDisplayFlag =
rsNodes("Display_Flag").Value & ""
5         strRefElements(0).strExternalDir =
Trim$(rsNodes("External_Directory") & "")
        strRefElements(0).strExternalFile =
Trim$(rsNodes("External_File") & "")
        strRefElements(0).strProcedure =
10 Trim$(rsNodes("Procedure_To_Call") & "")
        intElementCount = intElementCount + 1
        End If
        If Len(strSortType) = 0 Then strSortType = "ASC"
        If Len(strSortElements) = 0 And
15 Len(Trim$(strSortElement)) > 0 Then strSortElements =
strSortElement & "-" & strSortType & ";"

        ' Check if we want to add the element to the
array.
20         blnAddToArray = True
        For intArrayCount = 0 To UBound(strRefElements)
            If strRefElements(intArrayCount).strElement =
strRefElement Then
                blnAddToArray = False
25             Exit For
            End If
        Next
        ' Add the element to the array.
        If blnAddToArray Then
30             ReDim Preserve strRefElements(intElementCount)
            As typElementInfo
                strRefElements(intElementCount).strElement =
strRefElement
                strRefElements(intElementCount).strSeperator =
35 rsNodes("Seperator").Value & ""

```

```

        strRefElements(intElementCount).strDistinctFlag
= rsNodes("Distinct_Flag").Value & ""
        strRefElements(intElementCount).strDisplayFlag =
rsNodes("Display_Flag").Value & ""
5         strRefElements(intElementCount).strExternalDir =
Trim$(rsNodes("External_Directory") & "")
        strRefElements(intElementCount).strExternalFile
= Trim$(rsNodes("External_File") & "")
        strRefElements(intElementCount).strProcedure =
10 Trim$(rsNodes("Procedure_To_Call") & "")
        intElementCount = intElementCount + 1
    End If

    If AddItemToString(strSortElements,
15 strSortElement, ";") Then
        strSortType = rsNodes("Sort_Type_Code")
        If Len(strSortType) = 0 Then strSortType = "ASC"
        strSortElements = strSortElements &
strSortElement & "-" & strSortType & ";"
20     End If

    ' Element_Name returns the elements that are part
of the primary key.
    If Trim$(rsNodes("Element_Name") & "") <> "" Then
25         ' Check whether to add the item to the string.
        If AddItemToString(strRPNodeKey,
rsNodes("Element_Name").Value, ";") Then
            strRPNodeKey = rsNodes("Element_Name").Value &
";" & strRPNodeKey
30         End If

        ' Check if fields have been added to the
WhereClauseFields variable.
        If Len(strWhereClauseFields) Then
35         ' If there are fields, this ensures that no
duplicates are added.

```

```

        If Instr(1, strWhereClauseFields,
rsNodes("Node_Relation") & rsNodes("Relation_Separator") &
rsNodes("Node_Element") & rsNodes("Operator")) = 0 Then
            strWhereClauseFields = strWhereClauseFields
5  & (rsNodes("Node_Relation") & "") &
(rsNodes("Relation_Separator").Value & "") &
(rsNodes("Node_Element") & "") & (rsNodes("Operator").Value
& "")

            strOperators = strOperators &
10 rsNodes("Operator") & ";"

            If (rsNodes("Parent_Relation") & "") = " "
Then
                ' Check if the type is a string value.
                'If rsNodes("Value").Type > 7 Or
15 rsNodes("Value").Type < 2 Then
                    ' strWhereClauseFields =
strWhereClauseFields & StrToFld(rsNodes("Value") & "") &
";"

                    'Else
20 strWhereClauseFields =
strWhereClauseFields & (rsNodes("Value") & "") & ";"
                    'End If
                Else
                    strWhereClauseFields =
25 strWhereClauseFields & (rsNodes("Parent_Relation") & "") &
(rsNodes("Relation_Separator").Value & "") &
(rsNodes("Parent_Element") & "") & ";"
                End If
            End If

30 Else
        ' Add the first field to the WhereClauseFields
that is part of the where clause.
        If Len(rsNodes("Node_Relation") & "") Then
            strWhereClauseFields = strWhereClauseFields
35 & (rsNodes("Node_Relation") & "") &
(rsNodes("Relation_Separator").Value & "") &

```

```

(rsNodes("Node_Element") & "") & (rsNodes("Operator").Value
& "")
        strOperators = strOperators &
rsNodes("Operator") & ";"
5         If (rsNodes("Parent_Relation") & "") = " "
Then
        'If rsNodes("Value").Type > 7 Or
rsNodes("Value").Type < 2 Then
        ' strWhereClauseFields =
10 strWhereClauseFields & StrToFld((rsNodes("Value") & "")) &
        ";"
        'Else
        strWhereClauseFields =
strWhereClauseFields & (rsNodes("Value") & "") & ";"
15         'End If
        Else
        strWhereClauseFields =
strWhereClauseFields & (rsNodes("Parent_Relation") & "") &
(rsNodes("Relation_Separator").Value & "") &
20 (rsNodes("Parent_Element") & "") & ";"
        End If
        End If
        End If
        End If
25         rsNodes.MoveNext
        End If
        Loop

        On Error Resume Next
30         ' Load the imagelist with the new items.
        Set imgImage = imlTreePics.ListImages.Add(,
strNodeName & "_Closed", LoadPicture(strClosedPicture))
        Set imgImage = imlTreePics.ListImages.Add(,
strNodeName & "_Open", LoadPicture(strOpenPicture))
35

```

```

        ' Return the primary key values into this string.  We
will remove any duplicate fields.
        strPK = ReturnPrimaryKey(strRPNodeKey)

5          ' Select the detailed information from the Source
Relation.
        strSQL = "SELECT "
        For intArrayCount = 0 To UBound(strRefElements)
            If UBound(strRefElements) > 0 And strSQL <> "SELECT
10      " Then strSQL = strSQL & ", "
            If strRefElements(intArrayCount).strDistinctFlag =
YES_STR Then
                blnDistinct = True
                strSQL = strSQL & "DISTINCT(" &
15      strRefElements(intArrayCount).strElement & ")"
            Else
                strSQL = strSQL &
Trim$(strRefElements(intArrayCount).strElement)
            End If
20          ' Remove duplicates from the primary key.
            strPK = RemoveDuplicates(strPK,
strRefElements(intArrayCount).strElement)
        Next
        If Not blnDistinct Then
25          If Len(strPK) Then
                If strSQL = "SELECT " Then
                    strSQL = strSQL & strPK
                Else
                    If Right$(strSQL, 2) = ", " Then strSQL =
30      Left$(strSQL, Len(strSQL) - 2)
                    strSQL = strSQL & ", " & strPK
                End If
            End If
        End If
        End If
35      strSQL = strSQL & " FROM " & strReferenceRelation

```

```

        ' Add the WHERE clause of the SQL statement.
        ' If no node was clicked (form loaded), then do the
first part of the IF statement.
        If Len(strKey) = 0 Then
5           If pstrTreeType = PROJECT_TREE Then
                strSQL = strSQL & " WHERE " & Left$(strRPNodeKey,
Len(strRPNodeKey) - 1) & " = " & plngProjectID
                End If
            Else
10             ' Ensure that there is a where clause.
                strWhereClause = ReturnWhereClause(strKey,
strWhereClauseFields, strParentRelation, strOperators)
                If Trim$(strWhereClause) <> "" Then
                    strSQL = strSQL & " WHERE " & strWhereClause
15                 End If
                End If

                ' Add the sort criteria.
                strSQL = strSQL & " ORDER BY " &
20 ReturnSortElements(strSortElements)

                Set rsResults =
gobjDBConnect.rdoDB.OpenResultset(strSQL, rdOpenForwardOnly,
rdConcurReadOnly, rdExecDirect)
25
                If rsResults.EOF Then
                    If strReferenceRelation = "RPPProject" Then
                        MsgBox "There are no Projects in the system."
                        Screen.MousePointer = vbArrow
30                     Exit Sub
                    End If
                End If

                strRelations = strSourceRelation & ";" &
35 strReferenceRelation

```

```

' Reset the array
On Error GoTo ADErr
Do While Not rsResults.EOF

5       strPrimaryKey = ""
       ' Create the primary key.
       For intCol = 0 To rsResults.rdoColumns.Count - 1
           ' Build the primary key from the resultset
           ensuring all applicable fields are used.
10          intPos = InStr(1, strRPNodeKey,
rsResults(intCol).Name, vbTextCompare)
           If intPos > 0 Then
               If intPos = 1 Then
                   strPrimaryKey = strPrimaryKey & ">" &
15 rsResults(intCol).Name & ">" & rsResults(intCol).Value
               Else
                   If Mid$(strRPNodeKey, intPos - 1, 1) = ";"
Then
                       strPrimaryKey = strPrimaryKey & ">" &
20 rsResults(intCol).Name & ">" & rsResults(intCol).Value
                   Else
                       intPos = InStr(intPos + 1, strRPNodeKey,
rsResults(intCol).Name, vbTextCompare)
                       If intPos > 0 Then
25 strPrimaryKey = strPrimaryKey & ">" &
rsResults(intCol).Name & ">" & rsResults(intCol).Value
                       End If
                   End If
               End If
           End If
30       End If
       Next

       strName = ""
       For intArrayCount = 0 To UBound(strRefElements)
35       If strRefElements(intArrayCount).strDisplayFlag <>
NO_STR Then

```

```

        If
Trim$(strRefElements(intArrayCount).strElement) <> "" Then
        If Len(strName) And
Len(Trim$(rsResults(strRefElements(intArrayCount).strElement
5  ).Value & "")) > 0 Then strName = strName & strSeperator
        strName = strName &
Trim$(rsResults(strRefElements(intArrayCount).strElement).Va
lue & "")
        ' If the seperator is a period with no leading
10 or trailing spaces, then
        ' don't add the trailing and leading space
that we do with the other seperators.
        If strRefElements(intArrayCount).strSeperator
= "." Then
15         strSeperator = "."
        Else
        strSeperator = " " &
Trim$(strRefElements(intArrayCount).strSeperator) & " "
        End If
20     Else
        ' Check if we call the expression editor.
        If strRefElements(intArrayCount).strProcedure
<> "" And strName = "" Then
        strName =
25 GetExpression(strRefElements(intArrayCount).strProcedure,
strRefElements(intArrayCount).strExternalDir,
strRefElements(intArrayCount).strExternalFile,
strPrimaryKey, strReferenceRelation)
        End If
30     End If
    End If
Next

    ' Create the node.
35     strWhereClauseFields = strWhereClauseFields & "<" &
strOperators

```

```

        Call CreateNode(strKey, lngNodeId, strRelations,
strName, strPrimaryKey, strWhereClauseFields, strNodeName)
        rsResults.MoveNext
    Loop
5
    End If

    If blnNodeIdChanged Then GoTo Continuation_Point

10    On Error Resume Next
        rsNodes.MoveNext

    Loop

15    rsNodes.Close
    On Error Resume Next
    rsResults.Close

    Screen.MousePointer = vbDefault

20    Exit Sub

ADErr:
    Dim errError As rdoError, strMsg As String
25    For Each errError In rdoEngine.rdoErrors
        strMsg = strMsg & errError.Number & " - " &
errError.Description & vbCr
    Next
    Screen.MousePointer = vbArrow
30    MsgBox strMsg, vbOKOnly + vbExclamation, App.ProductName

End Sub

35 The following code is preferably used to create the node
    (Physically) in the treeview.

```

```

Private Sub CreateNode(ByVal strKey As String, ByVal
lngNodeId As Long, ByVal strRelations As String, ByVal
strName As String, ByVal strPrimaryKey As String,
5 strWhereClauseFields As String, strNodeName As String)

' This procedure creates the physical node on the treeview
form.

10 Dim nodX As Node      ' Create variable.
Dim strCurNodeKey As String
Dim intPos As Integer
Dim strReferenceRelation As String

15 strCurNodeKey = ">NODE>" & lngNodeId

intPos = InStr(1, strRelations, ";")
strReferenceRelation = Mid$(strRelations, intPos + 1,
Len(strRelations) - intPos)

20 ' If strRPNodeKey is blank, then don't include the next
statement.
If Trim$(strPrimaryKey) <> "" Then
    If Right$(strCurNodeKey, 1) = ">" Then strCurNodeKey =
25 Left$(strCurNodeKey, Len(strCurNodeKey) - 1)
    strCurNodeKey = strCurNodeKey & strPrimaryKey
Else
    intPos = InStr(1, strKey, ">NODE>", vbTextCompare)
    intPos = InStr(intPos + Len(">NODE>"), strKey, ">")
30 On Error Resume Next
    strCurNodeKey = strCurNodeKey & Mid$(strKey, intPos,
(Len(strKey) - intPos) + 1)
End If
strCurNodeKey = tvwTables.Nodes.Count & strCurNodeKey
35 ' If strKey is blank, then the node is a root node.

```

```

On Error Resume Next
If Len(strKey) <> 0 Then
    Set nodX = tvwTables.Nodes.Add(strKey, tvwChild,
    strCurNodeKey, strName, strNodeName & "_Closed")
5 Else
    Set nodX = tvwTables.Nodes.Add(, , strCurNodeKey, strName,
    strNodeName & "_Closed")
End If

10 nodX.Tag = strRelations & ">" & strWhereClauseFields

End Sub

```

J. Visual Basic Dynamic Property Sheet

15

The following code dynamically creates a Property Sheet for the preferred embodiment of the invention.

Private Sub LoadData(ByVal strWhereClause As String)

```

20 ' -----
' -----
'
' This procedure loads the data in the grid. If lngValue is
passed,
25 ' then the data is loaded because the user selected an item
from the
' cboObjects combo box.
'
' The grid contains the following information:
30 ' Field 0 - Field Name (ie. Project_ID)
' Field 1 - Value (ie. oms for Project_ID = 1 if that is the
associated Project_Name)
' Field 2 - Control type to be used. (ie. Display value in
a combo box, text box, elipsis)
35 ' Field 3 - Value (same as Field 1, but used to compare if
the fields were changed.)

```

```

' Field 4 - Size of field (ie. 10 for Varchar2(10))
' Field 5 - Data type (ie. Varchar2, Numeric, Timestamp,
etc.)
' Field 6 - Modifiable (ie. Allow modifications to this
5 field?)
' Field 7 - Primary Key? (ie. Is this field part of the
primary key?)
' Field 8 - Actual value from main table.
' Field 9 - Source view or table (ie.
10 RPCross_Reference_Decode_View)
'
' -----
-----

15 Dim cl As rdoColumns
rdoColumns collection
Dim intCurrent As Integer
indexes
Dim strSQL As String
' SQL
20 string.
Dim rsCrossReference As rdoResultset
Resultset for RPCross_Reference_View
Dim strFieldName As String
Dim typArray() As ptypCrossReference
25 Dim intCount As Integer
Dim intCurPosition As Integer
Dim strValue As String
Dim blnArrayEmpty As Boolean
Dim blnAlternateValueFound As Boolean
30 Dim intAlternatePosition As Integer
Dim blnDisplayValue As Boolean
Dim strWhere As String
Dim strSource As String
Dim strElementName As String
35 Dim intLessNumRows As Integer
' If the
column is not to be displayed,

```

```

subtract this number from the current row

On Error GoTo LoadData_ER
5  pblnDisplayControl = False
  Call ClearGrid
  grdData.ColAlignment(1) = 0    ' Left align.
  If Len(strWhereClause) Then
    strSQL = Me.SQLCriteria(strWhereClause)
10 Else
    strSQL = Me.SQLCriteria("")
  End If

  ' Create the resultset for the table.
15 Set rsResult = gobjDBConnect.rdoDB.OpenResultset(strSQL,
  rdOpenForwardOnly, rdConcurReadOnly, rdExecDirect)
  Set cl = rsResult.rdoColumns

  ' Fetch the values from the lookup table.
20 strSQL = "SELECT * FROM RPCross_Reference_View"
  strSQL = strSQL & " WHERE UPPER(Relation_Name) = " &
  UCase(StrToFld(Left$(Me.SourceTable, InStr(1,
  Me.SourceTable, ";") - 1)))

25 Set rsCrossReference =
  gobjDBConnect.rdoDB.OpenResultset(strSQL, rdOpenForwardOnly,
  rdConcurReadOnly, rdExecDirect)

  blnArrayEmpty = True
30 Do While Not rsCrossReference.EOF
    blnArrayEmpty = False
    intCount = intCount + 1
    ReDim Preserve typArray(intCount)
    typArray(intCount).Control_Type_Code =
35 rsCrossReference("Control_Type_Code").Value & ""

```

```

        typArray(intCount).Display_Flag =
rsCrossReference("Display_Flag").Value & ""
        typArray(intCount).Element_Name =
rsCrossReference("Element_Name").Value & ""
5    typArray(intCount).Full_English_Desc =
rsCrossReference("Full_English_Desc").Value & ""
        typArray(intCount).Reference_Group =
rsCrossReference("Reference_Group").Value & ""
        typArray(intCount).Modifiable_Flag =
10 rsCrossReference("Modifiable_Flag").Value & ""
        typArray(intCount).Reference_Code =
rsCrossReference("Reference_Code").Value & ""
        typArray(intCount).Primary_Key_Flag =
rsCrossReference("Primary_Key_Flag").Value & ""
15    typArray(intCount).Reference_Database =
rsCrossReference("Reference_Database").Value & ""
        typArray(intCount).Reference_Relation =
rsCrossReference("Reference_Relation").Value & ""
        typArray(intCount).Reference_Element =
20 rsCrossReference("Reference_Element").Value & ""
        typArray(intCount).Reference_Display_Element =
rsCrossReference("Reference_Display_Element").Value & ""
        typArray(intCount).NodeID =
rsCrossReference("Node_Id").Value & ""
25    typArray(intCount).ParentNodeID =
rsCrossReference("Parent_Node_Id").Value & ""
        typArray(intCount).TreeGroupCode =
rsCrossReference("Tree_Group_Code").Value & ""
        rsCrossReference.MoveNext
30 Loop
    rsCrossReference.Close
    Set rsCrossReference = Nothing

' Insert the data into the grid
35 For intCurrent = 0 To rsResult.rdoColumns.Count - 1

```

```

    If intCurrent > pintRowsVisible - 1 Then          ' Add a
new row for each column in the
    grdData.Rows = intCurrent + 1 - intLessNumRows
    ' rdoColumns collection.
5    grdData.RowHeight(intCurrent - intLessNumRows) = 280
    End If

    On Error Resume Next
    grdData.Row = intCurrent - intLessNumRows
10    If Err.Number = 30009 Then      ' Invalid row value.
        grdData.Rows = intCurrent + 1 - intLessNumRows
    ' rdoColumns collection.
        grdData.RowHeight(intCurrent - intLessNumRows) = 280
        grdData.Row = intCurrent - intLessNumRows
15    Err.Clear
        On Error GoTo 0
    End If

    ' Store the field name and value to be used later.
20    strFieldName = cl(intCurrent).Name
    On Error Resume Next
    strValue = cl(intCurrent).Value
    On Error GoTo 0

25    grdData.Col = 2

    ' For the current field, loop through the array and see if
the field has any related
    ' values in the cross reference table. This allows us to
30    later display the value
    ' rather than the code.
    strSource = 0
    intCurPosition = 0
    intAlternatePosition = 0
35    If Not blnArrayEmpty Then
        For intCount = 1 To UBound(typArray)

```

```

        If Trim$(UCase(typArray(intCount).Element_Name)) =
    UCase(strFieldName) Then
        If typArray(intCount).Reference_Code = strValue Then
            strSource = RPCROSS_REFERENCE_VIEW
5            blnDisplayValue = True
            intCurPosition = intCount
            Exit For
        Else
            strSource = RPCROSS_REFERENCE_VIEW ' CV
10        End If
        ' If there is a value for the Reference Relation and
        Element, then retrieve this value
        ' (description) from the database. This is used when
        displaying ID's from the database.
15        If Len(Trim$(typArray(intCount).Reference_Relation))
        > 0 And Len(Trim$(typArray(intCount).Reference_Element)) > 0
        Then
            ' If we are getting a blob, then do the following.
            If typArray(intCount).Control_Type_Code =
20 CONTROL_BLOB Then
                strSQL = "{CALL RPBlob_Extract(P_Blob_Id=>" &
                strValue & ", P_FileName=>" & strValue & ".txt')}"
                Dim intVal As Integer
                intVal = gobjDBConnect.COREExecute(strSQL)
25                If intVal <> False Then
                    ' Read the textfile.
                    Open GetSetting("Chameleon", "Options",
                    "Temporary Directory", RetrieveTempDir) & "\" & strValue &
                    ".txt" For Binary As #1
30                    strValue = Input(LOF(1), #1)
                    Close #1
                    End If
                Else
                    ' Retrieve the PK values for the table from
35 which we are fetching the description.

```

```

        strSQL = "SELECT * FROM
RPCross_Reference_Decode_View WHERE"
        strSQL = strSQL & " Node_Id = " &
typArray(intCount).NodeID
5         strSQL = strSQL & " AND Tree_Group_Code =
'0099'"

        Set rsCrossReference =
gobjDBConnect.rdoDB.OpenResultset(strSQL, rdOpenForwardOnly,
rdConcurReadOnly, rdExecDirect)
10
        ' Select the value from the source table.
        strSQL = "SELECT " &
typArray(intCount).Reference_Display_Element
        strSQL = strSQL & " FROM " &
15 typArray(intCount).Reference_Relation
        strSQL = strSQL & " WHERE "
        strWhere = ""
        Do While Not rsCrossReference.EOF
        ' Create the WHERE clause.
20         strElementName = ""
        strElementName =
rsCrossReference("Node_Element")
        If InStr(1, strWhere, " " &
(rsCrossReference("Parent_Element")) & " = " &
25 rsResult(strElementName)) = 0 Then
            If Len(strWhere) Then strWhere = strWhere &
" AND "
            strWhere = strWhere &
(rsCrossReference("Parent_Element")) & " = " &
30 rsResult(strElementName)
            End If
            rsCrossReference.MoveNext
        Loop
        If Len(strWhere) Then
35         strSQL = strSQL & strWhere
        Else

```

```

        strSQL = Left$(strSQL, Len(strSQL) - 7)
    End If

    Set rsCrossReference =
5   gobjDBConnect.rdoDB.OpenResultset(strSQL, rdOpenForwardOnly,
    rdConcurReadOnly, rdExecDirect)
        ' Check if a resultset was populated.
        If Not rsCrossReference.EOF Then
            strValue = rsCrossReference(0).Value & ""
10        End If

        ' Concatenate the NODE_ID to strSource for later
        use.
            strSource = RPCROSS_REFERENCE_DECODE_VIEW & "-"
15    & typArray(intCount).NodeID

            intAlternatePosition = intCount
            blnAlternateValueFound = True
        End If
20    End If
        If typArray(intCount).Display_Flag = NO_STR Then
            intAlternatePosition = intCount
            blnAlternateValueFound = True
        End If
25    If typArray(intCount).Modifiable_Flag = NO_STR Then
            intAlternatePosition = intCount
            blnAlternateValueFound = True
        End If
        If typArray(intCount).Primary_Key_Flag = YES_STR
30    Then
            intAlternatePosition = intCount
            blnAlternateValueFound = True
        End If
        If typArray(intCount).Control_Type_Code <>
35    CONTROL_TEXT Then
            intAlternatePosition = intCount

```

```

        blnAlternateValueFound = True
    End If
End If
Next
5   End If

    If intCurPosition = 0 And blnAlternateValueFound = True
Then
        intCurPosition = intAlternatePosition
10  End If

    ' If no match was found, enter the value from the
SourceTable, otherwise, use the value
    ' from the cross reference table.
15  If intCurPosition = 0 Then
        grdData.Text = CONTROL_TEXT
        grdData.Col = 0
        grdData.Text = strFieldName
        grdData.Col = 1
20  If cl(intCurrent).Type = rdTypeTIMESTAMP Then
        grdData.Text = Format(cl(intCurrent).Value & "",
GetSetting("Chameleon", "Options", "DateFormat",
"DD/MM/YYYY"))
        grdData.Col = 3
25  grdData.Text = Format(cl(intCurrent).Value & "",
GetSetting("Chameleon", "Options", "DateFormat",
"DD/MM/YYYY"))
        Else
            grdData.Text = cl(intCurrent).Value & ""
30  grdData.Col = 3
            '
Enter the value in the fourth column as well so that
            grdData.Text = cl(intCurrent).Value & ""
            ' we
can verify if the value has changed.
            End If
35  grdData.Col = 4
            '
Store the size of the data in this column, so as to

```

```

    If cl(intCurrent).Type = rdTypeTIMESTAMP Then
        grdData.Text = 19
    Else
        grdData.Text = cl(intCurrent).Size & ""
5  limit what can be entered into the textbox.
    End If
    grdData.Col = 5
    grdData.Text = cl(intCurrent).Type & ""
    grdData.Col = 6
10  grdData.Text = YES_STR
    grdData.Col = 8
    grdData.Text = cl(intCurrent).Value & ""
    grdData.Col = 9
    grdData.Text = strSource
15  Else
    If typArray(intCurPosition).Display_Flag = YES_STR Then
        grdData.Text =
typArray(intCurPosition).Control_Type_Code
        grdData.Col = 0
20  grdData.Text = strFieldName
        grdData.Col = 1
        If blnAlternateValueFound = False Or blnDisplayValue
Then
            ' If we are to display a description from the
25  full_english_desc field.
            blnDisplayValue = False
            grdData.Text =
typArray(intCurPosition).Full_English_Desc
            grdData.Col = 3
30  grdData.Text =
typArray(intCurPosition).Full_English_Desc
        Else
            blnAlternateValueFound = False
            If cl(intCurrent).Type = rdTypeTIMESTAMP Then

```

```

        grdData.Text = Format(strValue,
GetSetting("Chameleon", "Options", "DateFormat",
"DD/MM/YYYY"))
        grdData.Col = 3
5         grdData.Text = Format(strValue,
GetSetting("PROGNAME", "Options", "DateFormat",
"DD/MM/YYYY"))
        Else
            grdData.Text = strValue
10         grdData.Col = 3
            grdData.Text = strValue
        End If
    End If
    grdData.Col = 4
15 Store the size of the data in this column, so as to
    If cl(intCurrent).Type = rdTypeTIMESTAMP Then
        limit what can be entered into the textbox.
        grdData.Text = 19
    Else
20         If typArray(intCurPosition).Control_Type_Code =
CONTROL_BLOB Then
            grdData.Text = 0
        Else
            grdData.Text = cl(intCurrent).Size & ""
25         End If
    End If
    grdData.Col = 5
    grdData.Text = cl(intCurrent).Type & ""
    grdData.Col = 6
30    grdData.Text =
typArray(intCurPosition).Modifiable_Flag
    grdData.Col = 7
    grdData.Text =
typArray(intCurPosition).Primary_Key_Flag
35    grdData.Col = 8
    On Error Resume Next

```

```

        grdData.Text = cl(intCurrent).Value & ""
        On Error GoTo 0
        grdData.Col = 9
        grdData.Text = strSource
5      Else
        intLessNumRows = intLessNumRows + 1
        End If
      End If
    Next
10  grdData.Col = 0
    If grdData.Text = "" Then
        grdData.Rows = grdData.Rows - 1
    End If

15  ' Reset the column
    grdData.Col = 0
    grdData.Row = 0

    ' Set the column width.
20  grdData.ColWidth(1) = (grdData.Width * 0.6) - 80

    cboCombo.Visible = False
    txtText.Visible = False
    cmdElipsis.Visible = False
25
    Exit Sub

LoadData_ER:
    MsgBox Err.Number & " - " & Err.Description, vbOKOnly +
30  vbExclamation, App.ProductName

End Sub

```

K. Code Segment for Forward Engineering Component

The following code manages dynamically generates PL/SQL source code based on the objects that have been
 5 design recovered and stored in the CORE Chameleon repository. This specific excerpt is designed to create the equivalent SQL "SELECT" statement component if the newly generated Oracle PL/SQL report.

```

10  CREATE OR REPLACE PROCEDURE RPPGEN_GET_SELECT (
      p_project_id
      RPEvent_Detail.Project_ID%Type ,
      p_module_id
      RPEvent_Detail.Module_ID%Type,
15      p_Debug_Switch                      Varchar2
      Default null ,
      p_FileDir                          VARCHAR2 Default
      'c:\core\Temp',
      p_FileName                          VARCHAR2 default
20      'Report.sql') AS

      v_Database_Name                    DBDatabase.Database_Name%Type;
      v_Relation_Name                    DBRelation.Relation_Name%Type;
      v_Element_Name                     DBElement.Element_Name%Type;
25      v_Alias_Name                      RPPRelation.Alias_Name%Type;
      v_Related_Database_Name
      DBDatabase.Database_Name%Type;
      v_Related_Relation_Name            DBRelation.Relation_Name%Type;
      v_Related_Element_Name             DBElement.Element_Name%Type;
30      v_Variable_Name
      RPVariable.Variable_Name%Type;
      v_stack_id                         RPStack.Stack_ID%Type;
      v_FileHandle                       UTL_FILE.FILE_TYPE;
      v_Called_Module                    Varchar2(1) default 'Y';
35      t_Counter                         Integer default 0;

```



```

-----
-----
-- Get The elements That are to be selected

5  v_FileHandle := UTL_FILE.FOPEN(p_FileDir, p_FileName, 'a');
   Open C_RPRELATION_FIELD;
   t_Expression := 'SELECT ';
   UTL_FILE.PUT_line(v_FileHandle, T_expression );
   t_expression := ' ';
10
   <<RPRELATION_FIELD_LOOP>>
   Loop
   Fetch C_RPRELATION_FIELD Into v_Relation_Name,
   v_Alias_Name, v_Element_Name;
15
   Exit RPRELATION_FIELD_LOOP When C_RPRELATION_FIELD%NOTFOUND;

   If V_Alias_Name != ' '
   Then
20     T_expression := T_expression || '          ' ||
V_Alias_Name || '.' || v_Element_Name;
   Else
     T_expression := T_expression || '          ' ||
V_Relation_Name || '.' || v_Element_Name;
25   End If;

   UTL_FILE.PUT_line(v_FileHandle, t_expression );
   t_expression := ' ';

30   End Loop RPRELATION_FIELD_LOOP;
   Close C_RPRELATION_FIELD;

-----
-----
35  -- Get The Relation That are to be selected from

```

```

Open C_RPACCESS;
t_Expression := 'FROM ';
UTL_FILE.PUT_line(v_FileHandle, T_expression );
t_expression := ' ';
5
<<RPACCESS_LOOP>>
Loop
Fetch C_RPACCESS  Into v_Relation_Name, v_Alias_Name;

10 Exit RPACCESS_LOOP When C_RPACCESS%NOTFOUND;

T_expression := T_expression || '          ' ||
V_Relation_Name || '      ' || v_Alias_name;

15 UTL_FILE.PUT_line(v_FileHandle, t_expression );
t_expression := ' ';

End Loop RPACCESS_LOOP;
Close C_RPACCESS;
20

-----
-----

-- Get The Items That Make up the Where Clause of the select
25 statement for the Program

Open C_RPACCESS_FIELD;
t_Counter := 0;
<<RPACCESS_FIELD_LOOP>>
30 Loop
Fetch C_RPACCESS_FIELD  Into v_Database_name,
V_Relation_name, v_Element_Name,
                                v_Related_Database_name,
V_Related_Relation_name, v_Related_Element_Name,
35                                v_Stack_id;
Exit RPACCESS_FIELD_LOOP When C_RPACCESS_FIELD%NOTFOUND;

```

```

t_Counter := t_Counter + 1;

If t_Counter = 1
5  Then
    t_Expression := 'WHERE ';
    UTL_FILE.PUT_line(v_FileHandle, T_expression );
    t_expression := ' ';
End If;

10
If V_STACK_ID != 0
Then
    T_expression := T_expression || '          ' ||
V_Relation_Name || '.' || v_Element_name || ' = ';
15  RP_Get_Stack_Detail( p_project_id, p_module_id,
p_Module_id, v_stack_id, 0, p_Debug_Switch, T_expression,
v_called_module );
Else
    T_expression := T_expression || '          ' ||
20  V_Relation_Name || '.' || v_Element_name || ' = ' ||
V_Related_Relation_Name || '.' || v_Related_Element_name ;
End If;

UTL_FILE.PUT_line(v_FileHandle, t_expression );
25  t_expression := ' ';

End Loop RPACCESS_FIELD_LOOP;
Close C_RPACCESS_FIELD;
-----
30  -----
-- Get The Sort Options That are to be used In the Program

Open C_RPSORT_OPTION;
t_Counter := 0;
35  <<RPSORT_OPTION_LOOP>>
Loop

```

```

Fetch C_RPSORT_OPTION Into V_Relation_name, v_Element_Name,
v_variable_Name;

Exit RPSORT_OPTION_LOOP When C_RPSORT_OPTION%NOTFOUND;
5
t_Counter := t_Counter + 1;

If t_Counter = 1
Then
10   t_Expression := 'ORDER BY ';
      UTL_FILE.PUT_line(v_FileHandle, T_expression );
      t_expression := ' ';
End If;

15  If V_Variable_Name != ' '
      Then
          T_expression := T_expression || ' ' ||
V_Variable_Name;
      Else
20   T_expression := T_expression || ' ' ||
V_Relation_Name || '.' || v_Element_name;
      End If;

      UTL_FILE.PUT_line(v_FileHandle, t_expression );
25  t_expression := ' ';

      End Loop RPSORT_OPTION_LOOP;
      Close C_RPSORT_OPTION;
      -----
30  -----
      UTL_FILE.FCLOSE(v_FileHandle);

      Return;

35  EXCEPTION

```

```

-- Handle the UTL_FILE exceptions meaningfully, and make
sure
-- that the file is properly closed.
WHEN UTL_FILE.INVALID_PATH THEN
5   UTL_FILE.FCLOSE(v_FileHandle);
   RAISE_APPLICATION_ERROR(-20060,
                           'RPGen_Get_Select: Invalid
Path');
WHEN UTL_FILE.INVALID_MODE THEN
10  UTL_FILE.FCLOSE(v_FileHandle);
   RAISE_APPLICATION_ERROR(-20061,
                           'RPGen_Get_Select: Invalid
Mode');
WHEN UTL_FILE.INVALID_FILEHANDLE THEN
15  UTL_FILE.FCLOSE(v_FileHandle);
   RAISE_APPLICATION_ERROR(-20062,
                           'RPGen_Get_Select: Invalid File
Handle');
WHEN UTL_FILE.INVALID_OPERATION THEN
20  UTL_FILE.FCLOSE(v_FileHandle);
   RAISE_APPLICATION_ERROR(-20063,
                           'RPGen_Get_Select: Invalid
Operation');
WHEN UTL_FILE.READ_ERROR THEN
25  UTL_FILE.FCLOSE(v_FileHandle);
   RAISE_APPLICATION_ERROR(-20064,
                           'RPGen_Get_Select: Read Error');
WHEN UTL_FILE.WRITE_ERROR THEN
   UTL_FILE.FCLOSE(v_FileHandle);
30  RAISE_APPLICATION_ERROR(-20065,
                           'RPGen_Get_Select: Write
Error');
WHEN UTL_FILE.INTERNAL_ERROR THEN
   UTL_FILE.FCLOSE(v_FileHandle);
35  RAISE_APPLICATION_ERROR(-20066,

```

```

                                'RPGen_Get_Select: Internal
Error');
WHEN OTHERS THEN
    UTL_FILE.FCLOSE(v_FileHandle);
5    RAISE_APPLICATION_ERROR(-20069,
                                'RPGen_Get_Select: Unknown
Error');

END RPGEN_GET_SELECT;
10 /

```

L. Generated PL/SQL Source Code

15 The following sample PL/SQL program represents the converted report, from its original QUIZ format. All of the program code is automatically generated, without any manual intervention.

```

--
----- PROJECT NAME           :    Sales History
20  ----- PROJECT MANAGER      :
----- PROJECT LEADER         :
----- PROJECT START DATE     :    25-AUG-98
----- PROJECT DESCRIPTION    :    0
-----
25  -----
-----
-----
----- MODULE NAME             :    ORDERREP
----- MODULE SOURCE           :    sales.qzs
30  ----- MODULE TYPE         :    quiz
----- MODULE DESCRIPTION    :
-----
-----
35  -----

```

----- Prompts -----

DECLARE

5

```
----- CORE SOFTWARE Variables -----
```

```
v_FileHandle          UTL_FILE.FILE_TYPE;
```

----- Variables that hold information defined in the

10 original report -----

```
D_DISCOUNT_PERCENT      numeric(2);
```

```
D_DISCOUNT      numeric(7);
```

```
----- Variables that hold information fetched in the cursor
```

15 -----

T_ORDER ORDER_AMT

ORDER.ORDER_AMT%Type;

T ORDER ORDER_ID

```
ORDER.ORDER_ID%Type;
```

20

CURSOR C Report Cursor Is

Select

ORDER.ORDER AMT,

ORDER.ORDER ID

25 From

ORDER ;

30

BEGIN

```
----- Open a file to store output -----
```

```
V FileHandle := UTL_FILE.FOPEN("C:\TEMP", "ORDERREP.TXT");
```

35

```

----- Open Cursor and Fetch, Exit when no more data -----
OPEN C_Report_Cursor;
LOOP
Fetch C_Report_Cursor Into
5      T_ORDER_ORDER_AMT,
      T_ORDER_ORDER_ID;

EXIT WHEN C_Report_Cursor%NOTFOUND;

10

----- Definitions of Variables -----
      D_DISCOUNT_PERCENT :=      7 /      100;

15  If      T_ORDER_ORDER_AMT >      10000
      then
          D_DISCOUNT :=      D_DISCOUNT_PERCENT *
T_ORDER_ORDER_AMT;
      else
20      D_DISCOUNT :=      0;
      End If;

----- Conditions to satisfy output criteria -----

25

----- Write Record to file -----
      UTL_FILE.PUT_Line( v_FileHandle, T_ORDER_ORDER_ID || ',' ||
T_ORDER_ORDER_AMT || ',' || D_DISCOUNT);

30  END LOOP;

      UTL_FILE.FCLOSE(V_FileHandle);

35  END;
/

```

-- Code Generation Completed

5 The above has thus described a method of recovering
design information of a legacy application program using a
common set of tokens to represent corresponding elements of
a plurality of different application programs which are in
the same or in different languages, and using a set of the
tokens relating to a particular legacy application program
10 to display aspects of the legacy application program or to
construct another application program in the same or in a
different language which is similar to or is modified from
the legacy application program.

15 A person understanding this invention may now
conceive of alternate embodiments and enhancements using the
principles described herein. All such embodiments and
enhancements are considered to be within the spirit and
scope of this invention as defined in the claims appended
hereto.

We Claim:

1. A method of recovering design information of a legacy application program comprising:

(a) parsing an application program to obtain plural parsed program parts,

5 (b) storing plural tokens which are common to similarly functioning program parts of various computer languages in a database,

(c) mapping the parsed program parts to corresponding ones of the common tokens in the database, and
10 storing the map in the database, and

(d) using the mapped tokens to generate a modified legacy application, a new application in a same language as the legacy application, a new application in a language different from the legacy application, or for display of
15 aspects of the legacy or new application via a user interface.

2. A method as defined in claim 1, in which the modified legacy application or either new application contains substantially the same objects, methods and business rules as contained in the legacy application
5 program, devoid of syntax specific aspects thereof.

3. A method as defined in claim 1 including storing a distinct knowledge database for each of plural program languages, and using one of the knowledge databases and a corresponding one of plural parsers to carry out the parsing
5 step in respect of a source code language used in the legacy application.

4. A method as defined in claim 3, in which the step of storing the map includes selecting ones of the common tokens which correspond to parsed program parts and

storing copies thereof in a separate database or record
5 related to the legacy application program.

5. A method as defined in claim 3, in which the
step of storing the map includes selecting ones of the
common tokens which correspond to parsed program parts and
storing pointers to the selected ones of the common tokens
5 in a separate database or record related to the legacy
application program.

6. A method as defined in claim 1 including
searching the mapped tokens for a particular string, and
generating a modified legacy application using a modified
string in place of the particular string.
5

7. A method as defined in claim 6 in which the
particular string is comprised of a two character year
designation, and substituting a four character year
designation in place of the two character year designation
5 as the modified string.

8. A method as defined in claim 1 in which the
database is a relational database.

9. A method of recovering design information of a
legacy application program comprising using a common set of
tokens to represent corresponding elements of a plurality of
different application programs which are in the same or in
5 different languages, and using a set of the tokens relating
to a particular legacy application program to display
aspects of the legacy application program or to construct
another application program in the same or in a different
language which is similar to or is modified from the legacy
10 application program.

10. A method of recovering design information of a legacy application program comprising translating the legacy application program into tokens representing elements of the program, and automatically rewriting the legacy application
5 program using the tokens to define elements to be included in the rewritten program.

11. A method as defined in claim 10 including rewriting the legacy application program in a computer language which is different from that of the legacy application program.
5

12. A method as defined in claim 10 including modifying at least one parameter of the legacy application program prior to or as it is being rewritten.

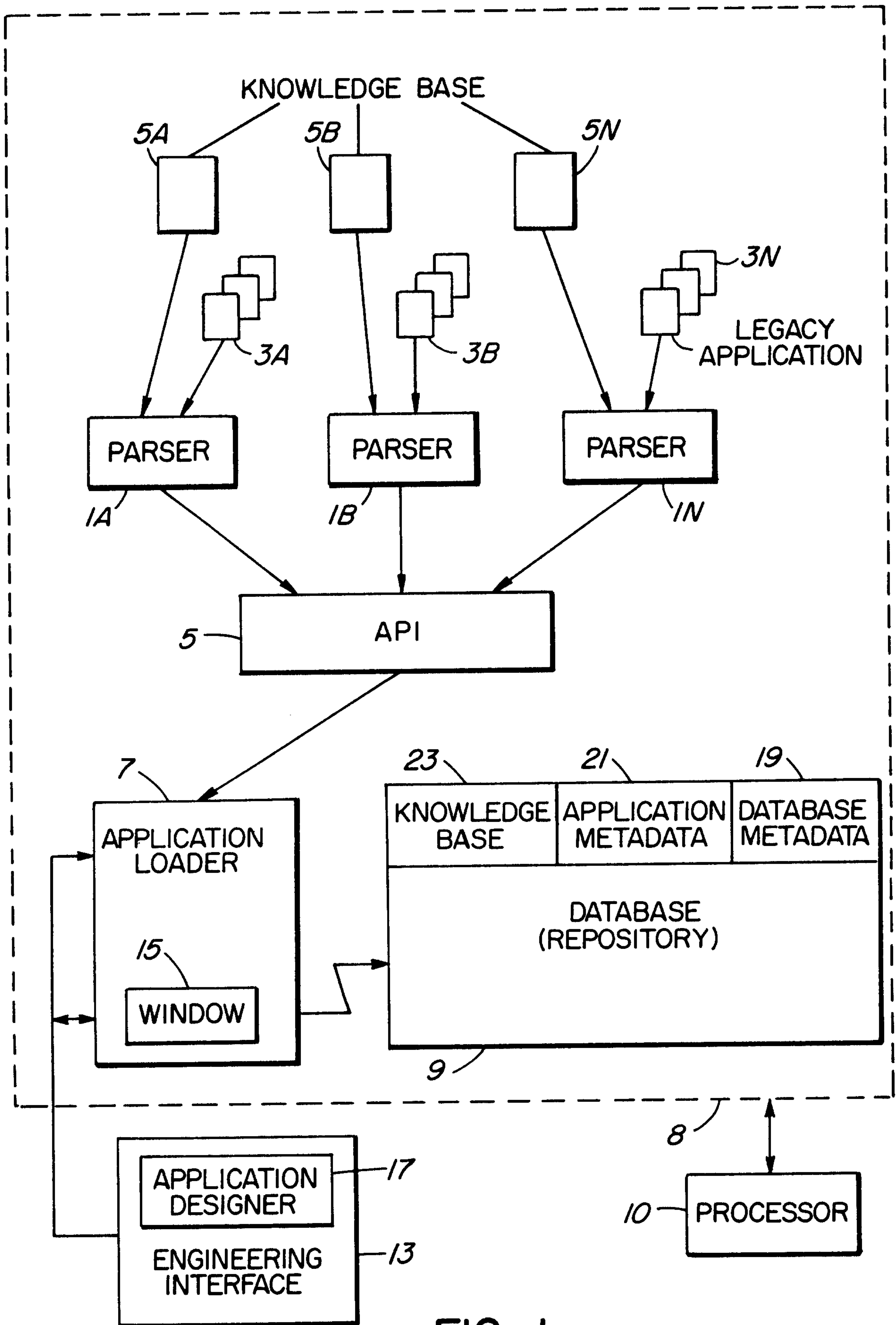


FIG. 1

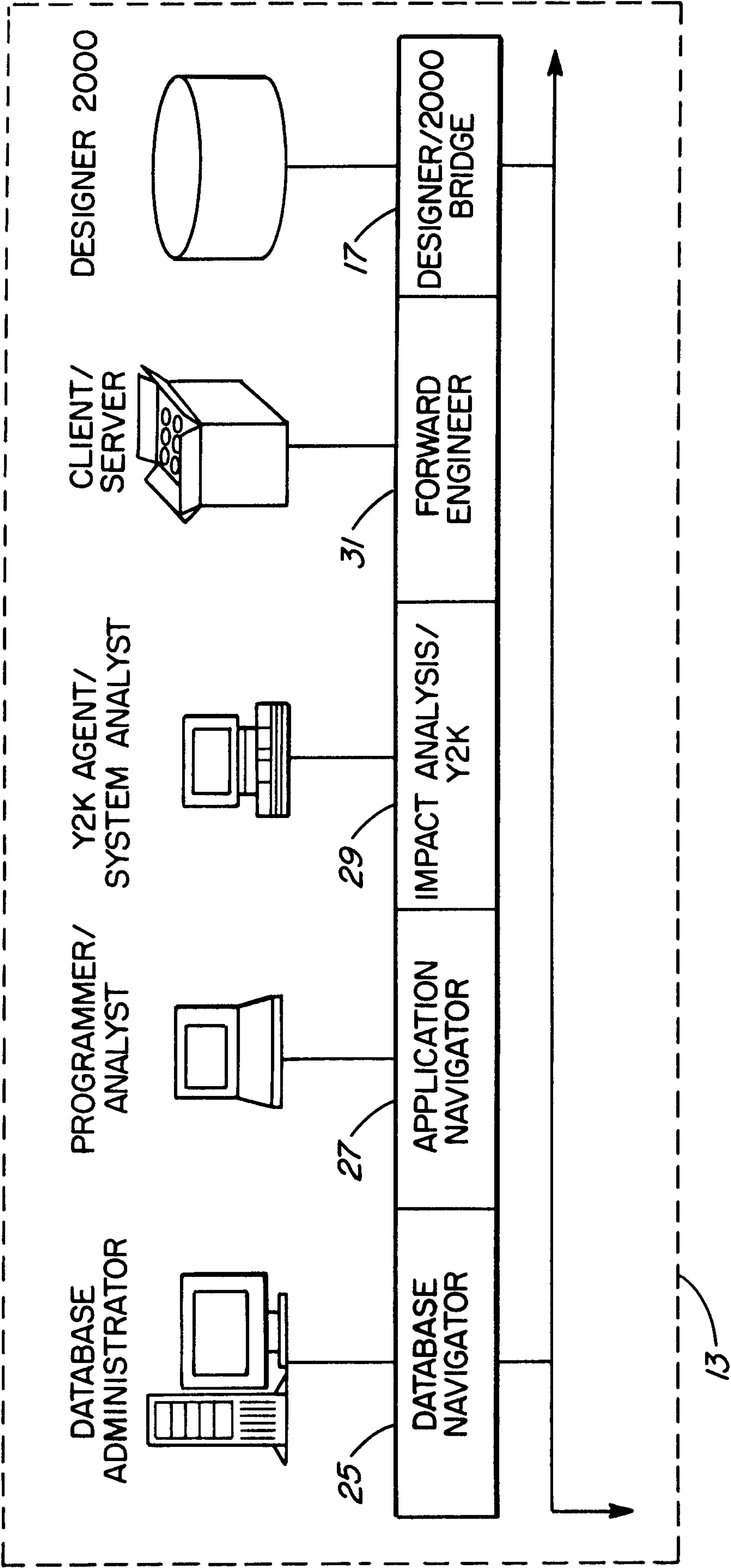


FIG. 2

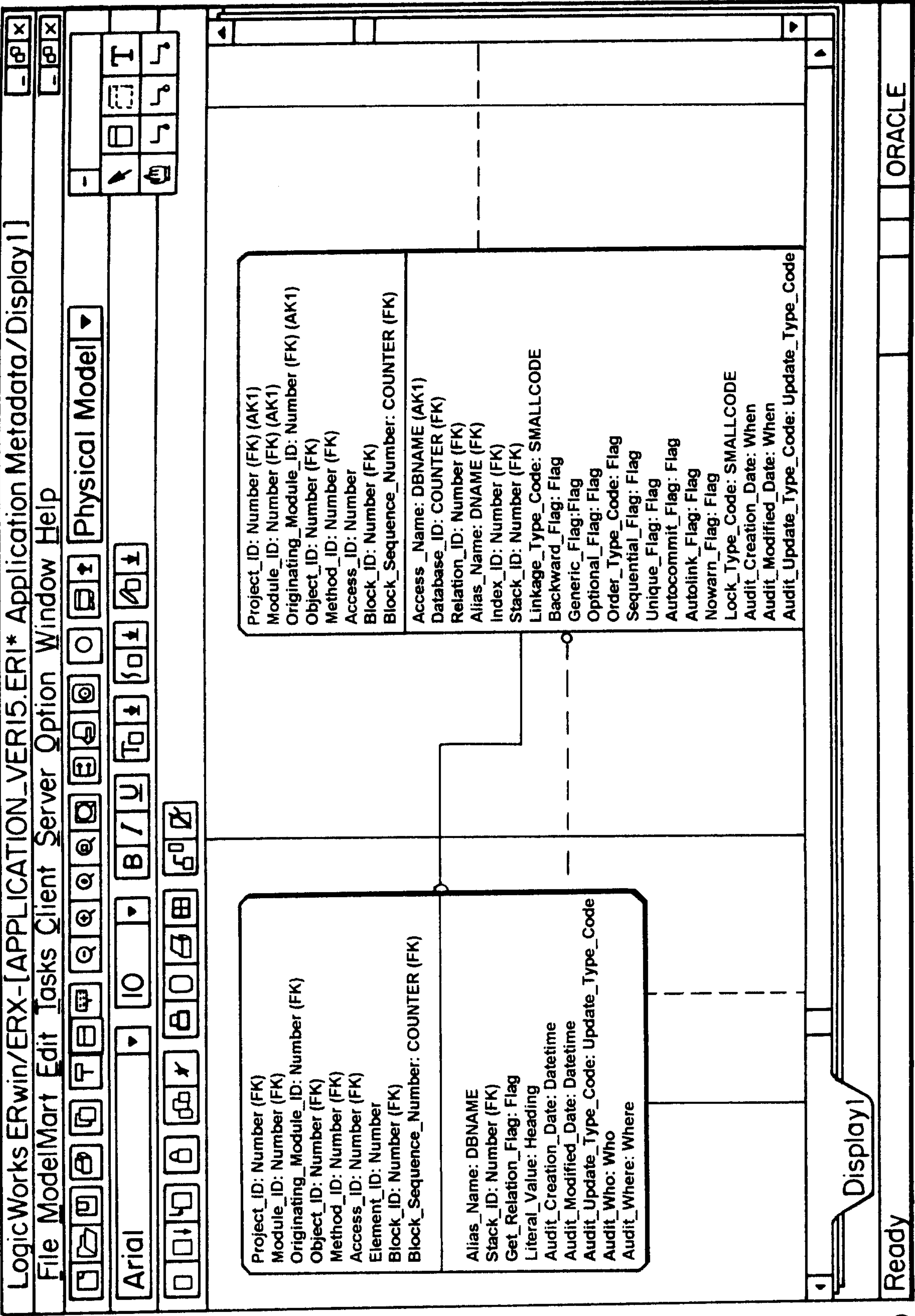


FIG. 3

Ready										ORACLE	
☐ Start	☐ Progra...	☒ Expla...	☐ Logic...	☒ Oracle	☒ My Co...	Control	☐ Channel	☐ Micro	☐ 8:41 AM		

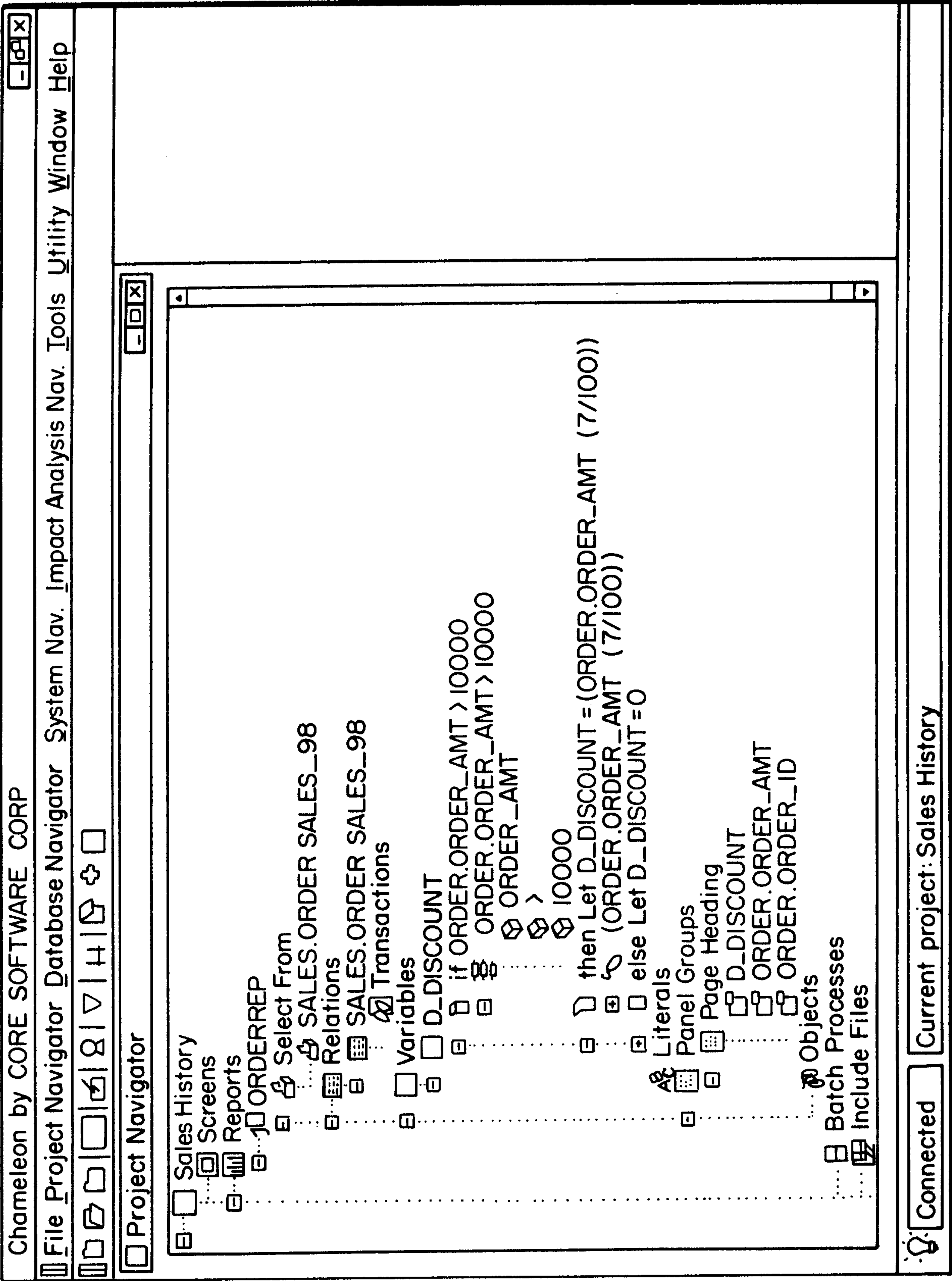


FIG. 5