



(11) (21) (C) **2,143,652**
(86) 1993/08/31
(87) 1994/03/17
(45) 2000/10/17

(72) Nuttall, David J. H., US

(72) Brehm, Bertram G., US

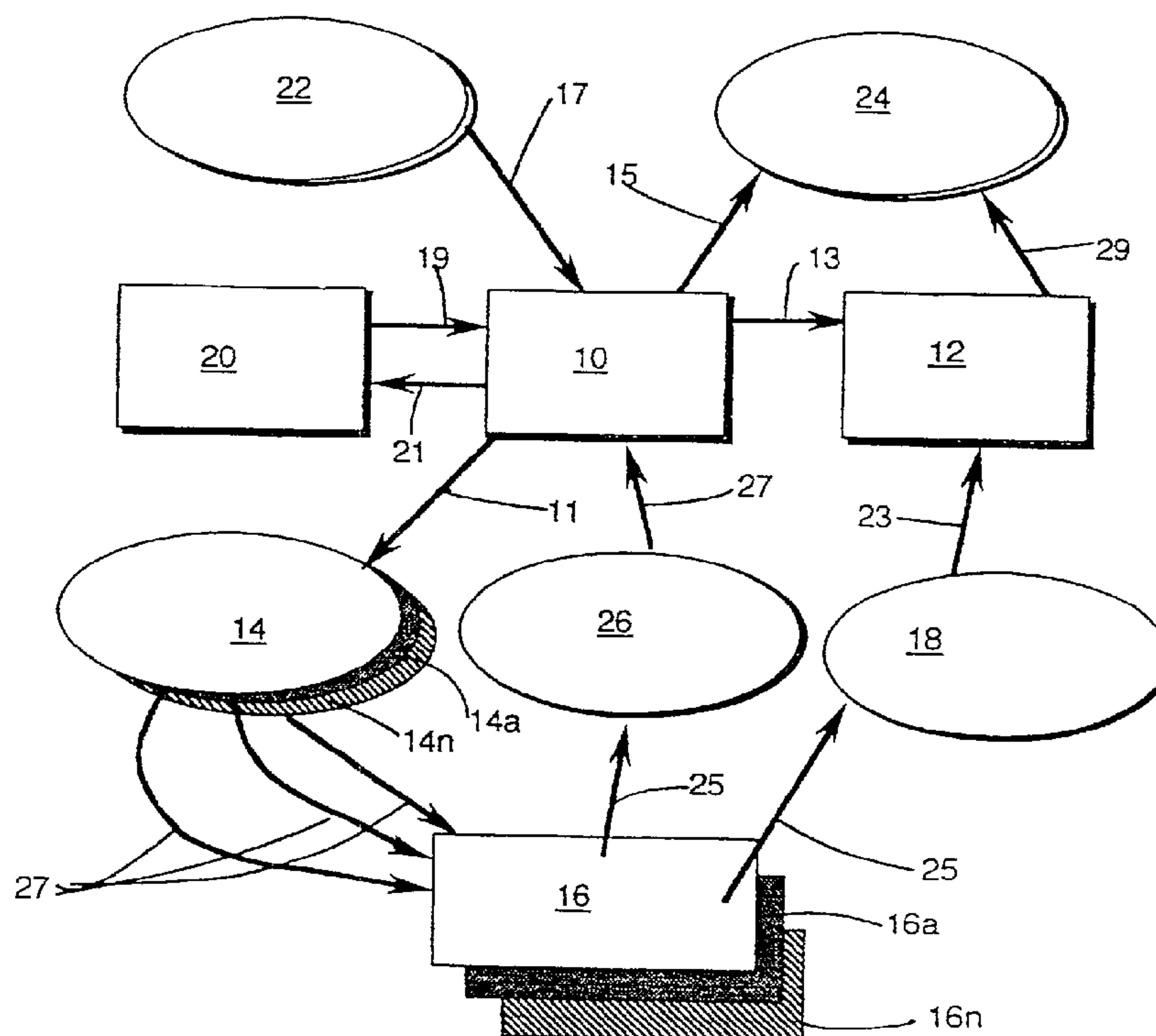
(73) INTERNATIONAL COMPUTERS LIMITED, GB

(51) Int.Cl.⁶ G06F 17/30

(30) 1992/09/01 (07/941,366) US

(54) **MODELE D'INFORMATION BASE SUR UN SYSTEME
PHYSIQUE**

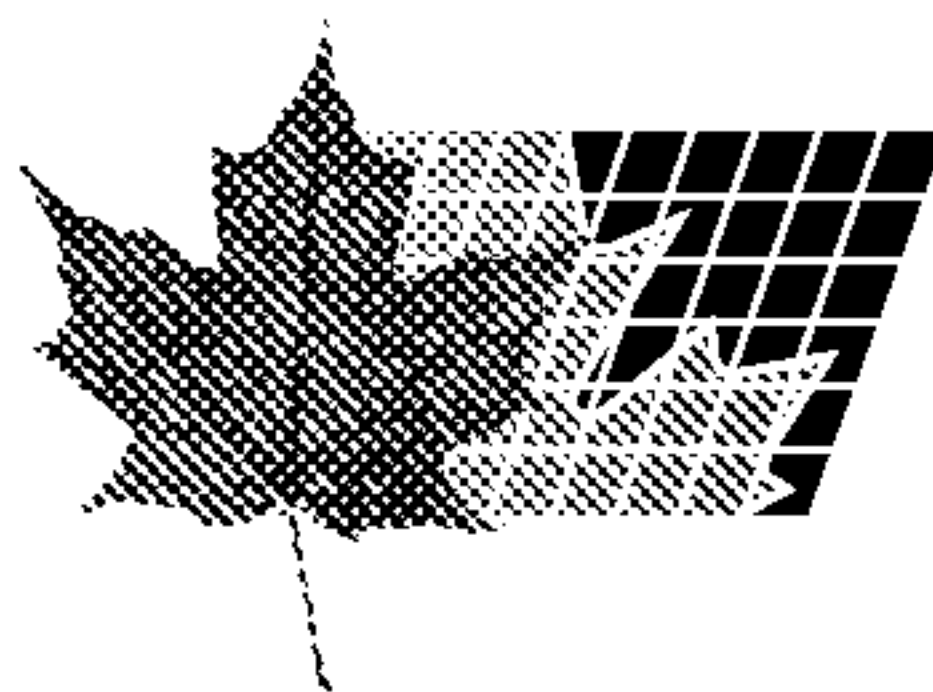
(54) **INFORMATION MODEL BASED ON A PHYSICAL SYSTEM**



(57) Modèle d'information basé sur un système physique, tel que l'équipement physique d'un système d'alimentation électrique. Un modèle d'information orienté sur les objets constitue un modèle de système d'alimentation générique qui peut être appliqué à toute application spécifique. Dans l'invention, les pièces physiques d'équipement sont représentées comme des objets avec attributs qui peuvent être

(57) An information model based on a physical system, such as the physical equipment in a power system. An object-oriented information model provides a generic power system model that may be applied to any of several specific applications. In the invention, physical pieces of equipment are represented as objects with attributes that can be verified (primary data) and relations including connectivity, grouping, and location.





(11) (21) (C) **2,143,652**
(86) 1993/08/31
(87) 1994/03/17
(45) 2000/10/17

vérifiés (donnée principale) et des relations comprenant la collectivité, le groupage et l'emplacement. Le modèle couvre toutes les configurations connues de systèmes d'alimentation et peut s'étendre à de nouvelles configurations. L'entrée d'attributs s'effectue à partir des sources principales et est utilisée pour calculer des données requises par les programmes d'application. Une interface utilisateur graphique du type à fenêtre simplifie de manière unique le fonctionnement de la base de données. Ainsi, l'invention fournit une source unique et facile à utiliser pour toutes les bases de données d'application privée dans un service public.

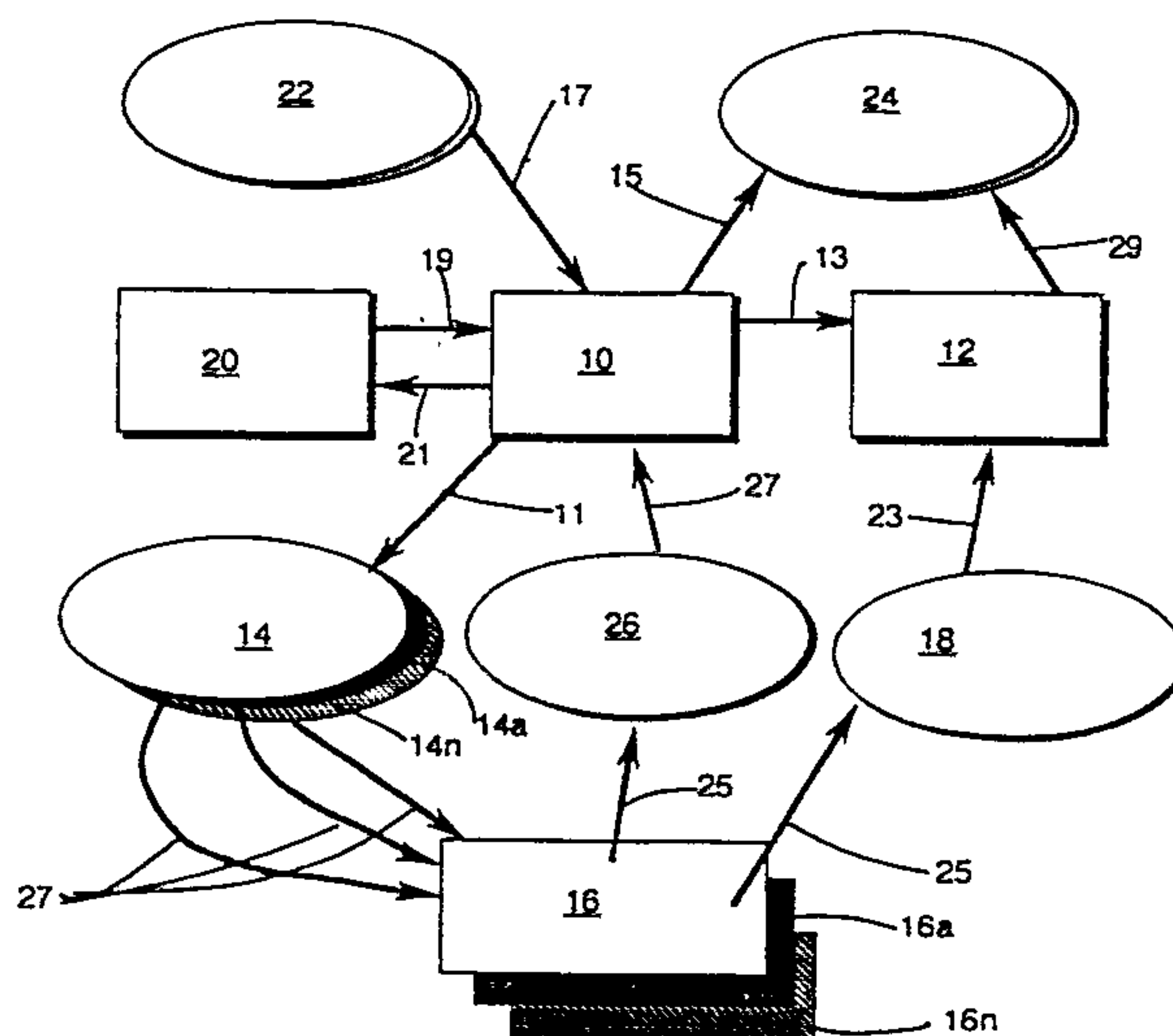
The model handles all known configurations of power systems and is extensible to new configurations. Attribute input is supported from primary sources and is used to calculate data required by applications programs. A window-based graphical user interface uniquely simplifies operation of the database. Thus, the present invention provides a single, easy to use, source for all proprietary application databases at a utility.



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁵ : G06F 15/40		A1	(11) International Publication Number: WO 94/06087
2143652			(43) International Publication Date: 17 March 1994 (17.03.94)
(21) International Application Number: PCT/US93/08233 (22) International Filing Date: 31 August 1993 (31.08.93) (30) Priority data: 07/941,366 1 September 1992 (01.09.92) US (71)(72) Applicants and Inventors: NUTTALL, David, J., H. [GB/US]; 20634 N.E. 181 Place, Woodinville, WA 98072 (US). BREHM, Bertram, G. [US/US]; 25119 N.E. 18 Street, Redmond, WA 98053 (US). (74) Agent: SLEATH, Janet; Stoel Rives Boley Jones & Grey, 600 University Street, 36th Floor, Seattle, WA 98101-3197 (US).			(81) Designated States: AU, CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published <i>With international search report.</i> <i>Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>

(54) Title: INFORMATION MODEL BASED ON A PHYSICAL SYSTEM



(57) Abstract

An information model based on a physical system, such as the physical equipment in a power system. An object-oriented information model provides a generic power system model that may be applied to any of several specific applications. In the invention, physical pieces of equipment are represented as objects with attributes that can be verified (primary data) and relations including connectivity, grouping, and location. The model handles all known configurations of power systems and is extensible to new configurations. Attribute input is supported from primary sources and is used to calculate data required by applications programs. A window-based graphical user interface uniquely simplifies operation of the database. Thus, the present invention provides a single, easy to use, source for all proprietary application databases at a utility.

2143652

INFORMATION MODEL BASED ON A PHYSICAL SYSTEM

5

A portion of the disclosure of this patent document contains material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

15

1. TECHNICAL FIELD

The present invention relates to object-oriented systems analysis. More particularly, the present invention relates to an object-oriented information model based on an underlying physical system, including associated infrastructure and personnel, for example the equipment which makes up an electric utility power system.

25 2. DESCRIPTION OF THE PRIOR ART

Electrical power systems are large complex physical systems with many types of interconnected electrical equipment. Such systems are often modeled for various reasons within the departments that make up the utility which is responsible for operating the power system.

Each department within the utility typically maintains its own specific database, usually employing a proprietary application program. This proprietary application program

35

2143652

F:\WP51\SP\35224.SP
7 September 1994:1530

- 2 -

is tightly coupled to a particular operating system,
computer manufacturer, data structure, etc. That is, the
proprietary application programs employed in the various
departments of the utility are vendor specific and cannot
5 be interfaced with, or exchange data with, the application
programs used in other departments (and often those within
the same department). Thus, in a utility, information is
not freely exchangeable between the various departments of
the utility, such as planning , engineering, operations,
10 etc.

The present state of the art is such that the various
proprietary application programs are, at best, difficult
for a department to use and maintain. That is, the user
15 interface in most such programs requires the manipulation
of strings of raw data, or the entry of data into crude
forms. Thus, highly skilled personnel are required for the
performance of a tedious and repetitive task.

20 Additionally, known database structures as are employed in
such application programs merely provide virtual models of
a power system based on abstract mathematical descriptions
of system equipment. Such models are generic
approximations of actual installed equipment within the
25 power system. Thus, there is an inherent error in the
model. When the error that is present in several separate
databases within the utility is considered, it can be seen
that the cumulative error renders data collection and
reporting within a utility a Tower of Babel.

30 Other workers have attempted to solve these problems by
combining graphics oriented artificial intelligence systems
with analytical dynamic modelling methods. One such system
is an object oriented prototyping program referred to as
35 "G2," that is available from Gensym Corporation of

2143652

- 3 -

F:\WP51\SP\35224.SP
7 September 1994:1530

Cambridge, Massachusetts, and which is described in "Entwicklung von Expertensystemen," Charles J. Kiezulas and Walter Klar, Elektronik, 1991. G2 employs objects as knowledge elements that are related to physical elements such as tanks, pumps, and valves in a beverage filling plant. The objects have attributes, such as flow rate and capacity, and connections with other objects. An attribute table is associated with each object to describe its characteristics. G2 provides the conventional form of object-oriented presentation of knowledge according to the frame concept of Minsky wherein each object belongs to a class and all the classes are organized hierarchically such that objects in lower classes can inherit the attributes of objects in higher classes.

Unfortunately, G2 does not provide all the features necessary to centrally maintain an accurate model of a physical system that has numerous important, but unobservable attributes, some of which are derived from real-time telemetry measurements. G2 also restricts its objects to classes in a hierarchy of classes, a restriction that unduly restricts the modelling of objects to other than hierarchical relationships among the objects.

Therefore, G2 is unable to accurately and completely model the complex equipment inventory, unobservable attributes, nonhierarchical relationships, telemetry measurements, and constantly changing environment of a massive, decentralized facility network such as a power utility.

Accordingly, the utility is faced with occupying its personnel with the many problems attendant with such application programs: redundant entry of the same data in different formats, inconsistent and inaccurate local departmental models, stale data as information within one

- 4 -

department does not track other departments, slow response on a system level to equipment changes, etc., all in the context of many complex application programs that are difficult and expensive to use and maintain.

- 5 The present invention relates to the provision of an object-oriented information model of a physical system (such as an electrical power system), which may include associated infrastructure and personnel, and employs the power, flexibility, and performance of a relational database to implement an
10 accurate, interactive, high-performance information model of the physical system.

The invention provides a computer-implemented method for modelling a physical system using an object-oriented information model in which elements of the physical system are
15 modelled by objects, the information model being implemented using a relational database, the method comprising: (a) storing in the relational database an object table, each entry in the object table containing an object identifier, an object type value, and a plurality of object attributes; (b) storing
20 in the relational database a plurality of type-specific tables, one for each of a number of defined object types, each entry in said type-specific tables containing an object identifier and a plurality of further object attributes; (c) storing in the relational database a supertype table defining a hierarchy
25 of object types; and (d) selectively accessing information stored in the relational database to perform operations involving said objects.

The preferred embodiment of the invention provides a single comprehensive description of the equipment in an electrical
30 power system, including network topology, operational constraints and limits, telemetry and communication details.

- 4a -

The organisation of information in the power system data model provided by the preferred embodiment of the invention is derived from an object-oriented analysis of the power system. Physical pieces of equipment are represented as
5 objects with attributes that can be verified (primary data) and relation, including connectivity, grouping, and location.

In the preferred embodiment, the user interface is based on a windowing environment. Where possible, data input is a matter of selection from a set of recognisable elements
10 contained within the system. In this way, entry error is significantly reduced.

2143652

F:\WP51\SP\35224.SP
7 September 1994:1530

- 5 -

5 In the embodiment, interfaces to existing application programs are also included. Import interfaces allow existing data to be captured. Export interfaces allow continuing support of base application programs. Thus, the present invention may be used to provide a single, easy to use source for all proprietary application program databases at a utility.

10 The invention will be further described in the context of a preferred embodiment, with reference to the accompanying drawings, in which:

15 Fig. 1 is a data flow diagram of an information model provided in accordance with the present invention;

Fig. 2 is a block level hierarchical representation of a data structure suitable for use in the present invention;

20 Fig. 3 is a block level representation of a first level database menu suitable for use in the present invention;

Fig. 4 is a block level representation of a type hierarchy menu suitable for use in the present invention;

25 Fig. 5 is a block level representation of a group hierarchy menu suitable for use in the present invention; and

Fig. 6 is a block level representation of a find object operation suitable for use in the present invention.

30

2143652

F:\WP51\SP\35224.SP
7 September 1994:1530

- 5a -

Power System Data Model

5 The embodiment provides a database that finds application
in the electric utility industry to hold attribute and
connectivity data for electrical power system equipment.
Duplication of data is reduced, and, through import and
export capability, a single point of maintenance is
provided for the operations, planning, engineering,
information services, and other departments of the electric
10 utility. This feature eliminates the need to maintain
multiple application program databases within the utility
by replacing the multiple databases with a single,
comprehensive, point of maintenance.

15 The embodiment allows the utility to define and maintain
connectivity information for electrical equipment. There
is an information flow, as shown in block diagram form in
Fig. 1, that provides a central repository of information
about the equipment which forms the electrical generation,
20 transmission, and distribution network of the electric
utility. This central repository is referred to in Fig. 1
as the power system data model 10.

The power system data model is implemented in a relational
25 database. The power system data model provides a data
transfer path 11 for information transfer, including
attributes and connectivity information. This information
is transferred to a process 14 which generates input files.
These input files are in turn exchanged, as indicated by
30 the information transfer path 27, with the proprietary
application program databases 16 that are used by the
various departments within the utility, e.g. design,
analysis,

2143652

planning, operations, and real-time software applications, including supervisory control and data acquisition (SCADA), distribution automation (DA/DMS), Energy management systems (EMS), and dispatcher training simulators (DTS).

5

The invention replaces the need to maintain multiple proprietary application program databases with a single, comprehensive point of maintenance in the power system data model 10. The external database generator 14 (14a-14n) uses
10 the power system data model 10 to create data exchange files which contain attribute data in an appropriate format to populate any of several proprietary databases 16 (16a-16n) for use by operations, engineering, or other applications.

15 For example, an electrical utility Energy Management System may integrate real-time and analysis applications from several different vendors, each vendor having a different proprietary database. In such system, the maintenance associated with electrical network modeling is performed by
20 the present invention. The data exchange files are transferred from the power system data model (exported) to the host system for the target database.

A data transfer path 25 is provided which allows information
25 from the various proprietary databases to be imported to the power system data model through a data port 26. The data port receives existence, attribute, and connectivity information from the proprietary database and provides this information along a data transfer path 27 to the power system
30 data model.

An historical data storage system 12 is included in the preferred embodiment of the invention that is capable of

2143652

7

storing timed samples (*i.e.* minimum, maximum, and average values every thirty seconds), continuous records (*i.e.* alarm and event logs), and snapshots (*i.e.* current system state) for later analysis. Such samples are provided along a data transfer path 23, which transfers information that is captured by an historical data capture process 18. The historical data capture process is in turn part of a data transfer path 25, which provides information to the historical data capture process from the various application programs 16 (16a-16n). The historical data storage system uses the power system data model 10, as shown by the data transfer path 13, to analyze historical data within the context of the power system.

15 The database design is easily extensible such that the power system data model can be expanded to encompass an enterprise data model. Thus, connections may be provided to external databases for AM/FM, GIS, and corporate computer systems.

20 For example, in some embodiments, the invention may operate in conjunction with an AM/FM/GIS system 20, in which existence and connectivity (described further below) establish one data transfer path 19 to the power system data model, and in which attributes establish another data transfer path 21 with regard to the power system data model.

25

30 The invention may be operated in a client-server environment including data servers, transaction managers, client applications, and user interface servers. Thus, a data server may be used to serve data to all interested users (clients). For example, user clients who are located throughout the power system network, a resource manager for the databases, and a transaction manager for a distributed

2143652

8

database system may all be served by data servers. A key point of the present invention is that the server provides a single (logical) model of the power system. Thus, there is only one comprehensive and consistent model of the power system, but that model may be distributed and/or replicated for better access throughout the power system network.

The invention provides processes both for interactive maintenance 22, having an attribute data transfer path 17, and for decision support 24, having an attribute/connectivity data transfer path 15 and a past performance data transfer path 29.

The present invention is implemented in relational database technology. Thus, relational tools are applied to data maintenance where traditional proprietary databases usually offer a user very little help, that is, in the areas of SQL access, ad-hoc queries, report generation, network access, and decision support. The present invention may be implemented using known database programming techniques. See, for example, Object-Oriented Systems Analysis, Shlaer and Mellor, Yourdon Press (1988) for a discussion of relational database techniques that are applicable in implementing the present invention.

Objects

The present invention is based on extending relational database technology and object-oriented systems analysis to a power system model. In object-oriented systems analysis, abstractions are produced that correspond to sets of physical things. These things are referred to as objects. Each object has a set of attributes which describe the object's

characteristics (discussed more fully below). A specific occurrence of an object, in which the object's attributes are populated with data, is referred to as an instance. All instances within a set of instances have the same characteristics and are subject to and conform to the same rules of behavior. Referential attributes are used to maintain the relationships between different objects.

The information model of the present invention describes:

- objects
- attributes of those objects;
- relationships between objects; and
- behavior of the object.

An object is an abstraction of a set of real-world things, *i.e.* underlying physical equipment, etc., such that:

- all real-world things in the set (the instances) have the same characteristics; and
- all instances are subject to and conform to the same rules and behavior.

Objects can be tangible things, roles, events, interactions, and specifications. Examples include:

- Tangible things: transformers, switches, lines;
- Roles: operators, dispatchers;
- Events: incidents, alarms, trouble calls, plans, schedules;
- Interactions: connections, measurements, groupings, correlations; and
- Specifications: equipment models, validation lists.

2143652

10

Lists of objects may be generated using a grouping mechanism to view the contents of the user-defined interactions, as well as lists of all the groups to which any single object belongs. Lists can also be generated using ad-hoc queries
5 with pattern matching of object names.

An object is named based on the following criteria: common names are preferred, the names should be 'strong' names, the names should have the same dimension and be precise, the
10 names should be based on the object's essential nature, and the name may be content based.

One aspect of the present invention provides a power system data model which is derived using an object-oriented analysis
15 of a power system. The model represents the physical types of equipment and their connectivity. The model contains rules that maintain referential and electrical integrity and which simulate the behavior of the power system.

20 The power system data model defines the existence of and assigns unique keys to equipment in the power system. Based on object-oriented systems analysis, the preferred embodiment of the power system data model describes four main types of equipment (described more fully below): conducting
25 equipment, sensing equipment, computer equipment, and support equipment. Relationships between equipment items are modeled, as are groupings that may be imposed by utility specific conventions.

30

Types

Typing is used in the present invention to enhance the description of an object in the database. For equipment in

2143652

11

the power system, typing is used to define a classification scheme that identifies the exact type of equipment to a degree of resolution defined by the end user. The same technique is used to identify other types of objects in the invention.

In the invention, supertypes and subtypes are used to categorize objects that are usually classified in a hierarchy. Thus, subtypes and supertypes are used to capture similarities between classes of things in the real world, including attributes and behavior. Subtypes and supertypes are used initially to model separate objects and then to bubble up the common attributes. This procedure establishes a type hierarchy which is used as a framework for the power system data model. In establishing subtypes and supertypes a bottom up system analysis is employed. Thus, the starting point is each physical element in the system.

A subtype is a more specific description of a parent supertype. A subtype therefore contains all properties possessed by its supertype. Thus, the attributes of a subtype are a property of a supertype object. For example, if all the subtypes of Switch have attributes describing their most recent state and normal state (Open or Closed), these attributes can be assigned to the switch object. They are then 'inherited' by all the objects that are a subtype of Switch. Other attributes may be common across many types of objects (e.g. manufacturer; location). These attributes may be moved up into classification objects, such as Conducting Equipment.

The present invention provides for three kinds of types:

2143652

12

- system types;
- user application types; and
- user types.

5 System types are pre-defined and may not be modified because
 they are necessary and fundamental to the correct operation
 of the power system model. User application types are used
 by user-written applications to define well known or reserved
 types. User types are defined and modified as necessary to
 10 express the desired depth of classification.

The three kinds of types are best understood as graphically
 depicted as follows:

15	<u>Key Value</u>	<u>Type</u>
	1k-10K	Systems
	10k-20k	Users
	.	
	.	
20	900k-1000k	
	1M-1.01M	User-defined
	.	
	.	
	999.99M-1000M	

25

Each new type must be related to a previously defined type.
 This relationship forms a supertype-type hierarchy. The
 hierarchy supports an unlimited number of levels but each
 type has one and only one supertype.

30

For example, a particular type of switch, e.g. a minimum oil
 circuit breaker, would be 'typed' by application of the
 invention as follows:

1. Object All things are objects.
 2. Equipment It is a piece of real equipment...
 3. Conducting Equipment ...that is a part of the power system.
 - 5 4. Switch It is a switch and...
 5. Breaker ...it can break load and...
 6. CB ...it is a circuit breaker and...
 - 10 7. Oil CB ...it is an oil circuit breaker and...
 8. M-OCB ...it is a minimum oil circuit breaker.
- 15 The invention defines types (Object, Equipment, Conducting Equipment, and Switch, above) that are used to allow the power system data model to interpret the type of object that is modeled. The user defined types (Breaker, CB, Oil CB, and Minimum-Oil CB, above) allow interactive users and
- 20 user written applications to alter model behavior based on the type.
- These types are depicted graphically as a hierarchy of types where the lower levels are subtypes. The four main
- 25 types of equipment – conducting equipment, sensing equipment, computer equipment, and support equipment (discussed below) – are supertypes of the other equipment types, which may, in turn, be supertypes of still other types.
- 30 One unique aspect of the present invention also provides for basetypes (or fundamental types), *i.e.* a formalized point in the type hierarchy that encompasses a number of

2143652

14

similar subtypes (similar in attributes and behavior).
The preferred embodiment of the present invention provides
Breaker, Fuse, and Disconnect as base types, for example.
All are subtypes of Switch (which could also be a
5 basetype). Base types are used to map objects for
editors, database tasks, and for procedures that model
behavior. For example:

● Switch

10

- Breaker (capable of interrupting load and reclosing)
 - > OCB, M-OCB, VCB, ABCB, GCB
- Fuse (capable of interrupting load)
 - > Current limiting, power, cutout, vacuum
- 15 - Disconnect (not capable of interrupting load)
 - > Station, transmission, distribution

Attributes

20 Once the components of a power system and the arrangement of
these components is set forth in terms of object-oriented
analysis, the objects are assigned attributes. An attribute
is the abstraction of a single characteristic possessed by
all the things that were abstracted as objects. In the
25 invention, it is preferred that attributes be collected for
each of the objects identified. During this process, some
of the attributes assigned to a group of objects are shared
in common.

30 The invention provides for obtaining a set of attributes for
objects, where the attributes are:

- complete: they capture all the information pertinent to the

object they define;

- fully factored: each attribute captures a separate aspect of the object abstraction; and

- mutually independent: the attributes take on their values independently of one another.

Attributes are assigned to four different categories depending on the type of information they capture:

- descriptive: intrinsic characteristics of an object, which are further divided into two categories, *i.e.*:

- primary: usually physically verifiable by observation, *e.g.* line length, tower type, conductor type; and

- secondary: not usually physically verifiable, but rather derived from others in the database by calculations that model behavior, *e.g.* impedance and capacitance;

- naming: arbitrary names and labels, *e.g.* line name;

- referential: facts that relate an instance of an object to an instance of another object, *e.g.* manufacturer, location (implemented using surrogate keys, discussed below); and

- identifiers: a set of one or more attributes that can uniquely identify an instance of an object, *e.g.* line name, segment name.

Attributes that can be derived from the primary data may be calculated automatically. For example, the primary data for a transformer is entered directly from the nameplate or test report. Individual winding characteristics are derived from this information. Transmission line impedances and admittances are derived from the primary data of conductor characteristics, tower geometry, and line segment lengths. Derived data may be input in lieu of primary data, if desired.

2143652

16

One unique aspect of the present invention is the use of two types of descriptive attributes. Thus, one goal of the present invention is to calculate secondary attributes where possible. Otherwise, direct entry of the attribute is employed. In this way, the present invention allows the modeling of system behavior.

Groups

10

Objects may be related to each other by assigning them to groups. Any particular object may belong to any unlimited number of different groups, although each object has a single primary group. Groupings are arbitrary collections of objects. Groups have meaning only as specified by the user. The objects in a group are related because they are member of (belong to) the same group. Groupings are modeled as relationships that may be imposed by utility specific conventions.

20

Relationships are abstractions of a set of associations that hold systematically between different kinds of things in the real world. There are an infinite number of relationships between types of things in a power system (and a large number of things themselves). Prior art approaches to modeling power systems express each relationship individually, resulting in unwieldy database structures. In the present invention, relationships may be:

- one-to-one;
- one-to-many; and
- many-to-many.

30

For example:

- a substation contains equipment;
- telemetry monitors equipment;
- lines are composed of segments;
- 5 - equipment may have a location;
- electrical equipment is connected to electrical equipment;
- electrical equipment may have ratings;
- transformers have windings.

10 Additionally, relationships may be conditional or unconditional.

One unique feature of the present invention provides for the use of one table for all groupings (i.e. for all
15 relationships, such as 'member of,' 'located with,' and 'shown on'). Typically, a database of the type described herein would rely on a separate table for each grouping relationship between each type of object.

20 Group-Member Relationship

The present invention contains a description of group-member relationship. Both the group and the member are objects in the database. Where a division belongs to a company, the
25 group is an object of the type *COMPANY*, the member is an object of type *DIVISION*, and the relationship is *BELONGS TO*. Other relationships may include: *LOCATED WITH*, which can be used to locate switches on poles that have a known address; *SHOWN ON*, which relates objects to maps and drawings; *RATED*
30 *AS*, which allows a common set of ratings to be held for a set of similar equipment; and *LOAD MODELED BY*, which allows a common load model to be applied to similar consumers.

2143652

18

The *SYSTEM* grouping holds references to all the objects that make up a version of the power system data model. Generational versions record the evolution of the equipment in the power system. Between two versions, the grouping mechanism duplicates only the references to objects common to both, not the objects themselves.

Additionally, objects may be stored as members of groups in at least two ways:

- as an object: e.g. a transformer
- as a network position identifier: e.g. a position or place holder in the power system.

By assigning a dual nature to an object, the object itself may exist in the model, along with its attributes, and a place in the power system may be held into which objects having different attributes may be placed based on historical need. For example, a power system may have a network position identifier for a transformer. An object, i.e. a transformer having a particular rating, manufacturer, etc. may currently occupy that network position. Based on historical data, projected needs, etc., it may be determined that the network position occupied by the particular transformer should be occupied by a transformer having a different rating, etc. In this way, the power system data model (and, accordingly, the power system) is readily modified by equipment interchange based on demand, need, plan, etc. The historical data can also be applied to the object itself, allowing the history of a device to be analyzed.

Normalization Rules

5 The following normalization rules are provided by the invention which are applied to the attributes to ensure that the descriptions are coherent:

- One instance of an object has exactly one value for each attribute;
- Attributes cannot contain an internal structure; and
- 10 - Every non-key attribute must be fully dependent on the primary key.

15 The normalization rules are similar to those normally applied to the design of a relational database. The first rule defines a 'table' in the relational database.

20 The second rule ensures that the attributes are fully factored. This is essential to avoid 'hard-coded' knowledge of the meaning of parts of an attribute (e.g. the first two digits of the equipment number represent the year of purchase).

25 The third rule refers to the use of 'keys.' In a database there must be a way of uniquely identifying anything in the database. In a hierarchical database this is accomplished by the combination of the name of the item plus the names of the parents of the item (e.g. Substation Foo, Transformer T1). In a relational database, each row in a table must have a key of its own. In some applications this is simple (e.g. 30 the name of a substation - Foo); in others compound (e.g. a name constructed from the identifier plus those of its parents - Foo T1).

The fourth rule requires that every attribute that is not part of the identifier must represent a characteristic of the entire object, and not a characteristic of something that is identified by part of the identifier (e.g. the attributes of the transformer Foo T1 cannot include the location of the substation Foo). This rule also prevents the use of attributes that are identified by other attributes within the object (e.g. the transformer may have a manufacturer attribute, but it is incorrect to store the manufacturer's address as an attribute of the transformer).

Surrogate Keys

In the preferred embodiment of the present invention, the actual primary key is modeled as a surrogate key, which is a computer generated numeric key that has no meaning to users. A surrogate key allows the user to change the name of an incidence of an object but not change references. All referential attributes are based on surrogate keys. Other (secondary) keys can be used to access objects by name, and the uniqueness rules for these keys can be defined on a case-by-case basis.

One unique feature of the present invention provides for the use of a balanced binary tree index with sequential keys. Surprisingly, the use of this technique in the present invention, taking advantage of the fact that the database is primarily 'read,' results in a much faster cycle time. Thus, the database is speeded up using a technique that would, by conventional wisdom, slow a database down.

Data Model

5 In summary, one aspect of the present invention provides a data model constructed in accordance with the following procedure:

- Create an extensible type hierarchy, allowing a user to extend types down the hierarchy;
- 10 • Collapse objects into tables, where attributes specific to all objects below base types are collapsed into type tables, and all objects above base types are collapsed into object tables:
 - Use typing to identify an object (not a table name);
 - 15 and
 - Use common object table with surrogate keys; and
- Externalize and collapse relationships, where three main relationships are defined in the preferred embodiment of the present invention, i.e. connectivity, measurements, and groupings:
 - 20 - Use typing to identify the type and relationship.

25 The data model incorporates the following procedures and operations to ensure integrity:

- Referential: to enforce constraints imposed by the database design, e.g. relationships between tables;
- 30 • Validation: to evaluate against real-world criteria, e.g. use engineering rules of thumb to reject bad data; and
- Behavioral: to model behavior of equipment, e.g. calculate

2143652

secondary/primary descriptive attributes, or split up load models.

5 These procedures are triggered when data is changed, e.g. SQL insert, update, delete statements, etc. The procedures cannot be bypassed by using a different user interface. This is vital for integrity and prevents the use of so-called 'back doors' to corrupt the database. These procedures may be turned off, e.g. for maintenance.

10

Triggers make the presence of various procedures transparent. Thus, a user would not be aware of the operation of the database. Rather, values are entered and results displayed. This is discussed in greater detail below.

15

Physical Model

20 An important aspect of the present invention is the use of a physical model. Thus, the descriptive attributes recorded for an object are verifiable. For example, control center applications may require the impedances for a three-winding transformer to be specified on a per-winding basis, corrected for the system base MVA and kV levels. These impedances cannot be validated by inspection. In the prior art, it is
25 common practice to enter these values directly into the databases used by the specific application programs concerned. If the data from the manufacturer's test report are entered into the database of the present invention, then the application data can be calculated at any point in the
30 system. This test data has a known provenance which can be checked if a question arises about the validity of the data.

Another example of the foregoing involves transmission line

impedances. If the characteristics of conductor types and tower geometries are known, it is possible to calculate the positive sequence impedances from the primary data of the line length and construction. By adding more information about the ground resistivity and the relationships to other lines in the same right-of-way, zero sequence impedance can also be calculated.

The power system data model describes four main types of equipment: conducting equipment, sensing equipment, computer equipment, and support equipment. The following relationships are established between these four major types:

1. Sensing equipment -> measures -> conducting equipment;
2. Support equipment -> supports -> conducting equipment;
3. Conducting equipment -> is connected to -> conducting equipment; and
4. Computer equipment -> controls and models -> conducting equipment.

The main types of the conducting equipment are:

Conducting equipment -> capacitor, switch, transformer, generator, conductor, reactor, consumer, etc.

The sensing equipment is assigned two major subtypes:

Sensing equipment ->

- protection: voltage relay, current relay, frequency relay, etc.; and

2143652

- telemetry: analog, digital, counter, control, etc.

The support equipment encompasses the equipment needed to keep the network in place:

5

Support equipment ->

- overhead: tower, pad, pole, etc;
- underground: manhole, pit, vault, etc; and
- etc.

10

Computer equipment may be described as follows:

Computer equipment ->

- hardware: display, printer, etc;
- 15 - software: program.

20

The relationships in the power system data model may be extended by the end user. Groupings are used to aggregate the equipment into larger units that reflect the utility's organization. For example, a utility may choose to define groups of equipment as follows:

Equipment group -> system, company, division, substation, bay, line, circuit, etc.

25

A circuit breaker can belong to a bay that belongs to a substation that belongs to a division that belongs to a company. This is represented graphically as follows:

30

Company -> division -> substation -> bay -> circuit breaker.

The same circuit breaker may also belong to a transmission line which may belong to two different divisions of different

2143652

25

companies.

That is,

- 5 Company 1 -> division 1 ->
 Company 2 -> division 2 ->
 -> transmission line -> circuit breaker

10 These example groupings are specific to a particular utility.
 Each database user can choose different groupings to
 accurately reflect the utility's organization.

15 Fig. 2 is a block level representation of an object 30 as may
 be used in the present invention as a database member based
 on physical equipment having both a location 32 and a rating
 34. The following types of conducting equipment are modeled
 in the power system data model provided in a preferred
 embodiment of the invention:

20 AC Overhead Line: The AC overhead line model constructs a
 line from a number of two terminal conductors 46. Each
 conductor has impedance and capacitance information
 associated with it that can either be entered directly or
 calculated by the power system data model from supplied
25 primary data, e.g. tower geometry 43, conductor
 characteristics 41, and conductor length.

30 Busbar: Busbars 45 are considered subtypes of conductors 46
 and may be modeled as physical devices, if desired. This
 characterization is in contrast to prior art practice of
 modeling a bus bar as an abstraction referred to as a node.

Capacitor: For capacitors 36, the installed MVar is modeled.

2143652

26

Consumer: A consumer 38 is used to model a 'load' 33 on the power system. For a transmission model, consumers may be distribution feeders or major customers. If power system modeling is extended to include the distribution primary, consumers may be commercial enterprises with a high voltage supply or they may be housing developments. If desired, the model may be extended to individual meters. Each consumer has a base load expressed in MW or MVar, plus voltage and frequency variations for the base load. A family of load curves that describe temporal variation of the base load can be associated with the consumer, such as season 35, day type 37, and load value 39.

Generator: Generator 42 MVA, MW, MVar, and kV ratings, resistance and reactance, and parameters for the rate of change of the unit are modeled.

Reactor: For reactors 44, the installed MVar is modeled.

Switch: The normal state of a switch 48 is modeled.

Transformer: The power system data model includes one- and three-phase, two- and three-winding transformers 50. Characteristics of the transformers may be entered directly from the nameplate and manufacturer's test reports as primary data, in which case the per-winding impedance information is calculated 49. The per-winding impedance data may be entered directly, if desired. Additionally, tap setting information 47 is modeled.

Transformer banks may also be supported by the present invention.

Additionally, the power system data model may also model PERSONs 54, COMPUTERs 56, and TELEMETRY 58, as desired.

5

User Interface

In the preferred embodiment of the invention, all equipment attribute data are maintained in a single database using a window-based graphical user interface. Primary data for an
10 object are input directly by a user. The use of windowing in the user interface permits precise control of input data at the point of entry using pull down menus and slider bars having a range bounded by the physical limits of the modeled equipment. Thus, out of range or incorrect settings are not
15 possible and a user can be guided through an interactive data input process with little previous experience or training.

Typically, the user of the invention is an engineer having responsibility for modeling decisions concerning the
20 electrical power system. The user interface also allows efficient data entry for administrative personnel.

Thus, the present invention features a consistent graphical user interface across all applications and for all users.
25 A user enters attribute data via pop-up menus, pull-down menus, scrollable lists, enterable fields, dialog boxes, and mouse pointing device support, generic implementation of which is well known in the art. Data entry fields are color coded to prompt the user for the appropriate type of input.
30 For example, electrical connectivity is defined using point and click operations to reference the physical terminals of the equipment being connected.

2143652

28

The invention allows several editors to be active at the same time. An editor can be started for any object in the database by clicking at any reference to that object. In the equipment editor windows, the user selects from option lists that only offer valid choices. Existing equipment may be used as a template to simplify data entry for new equipment.

The user interface of the present invention is configurable in the preferred embodiment for OSF/MOTIF, Open Look, or Windows. Typical hardware support for the present invention may include workstation type computers, such as VAXstations running VAX/VMS with INGRES or DECstations running ULTRIX/RISC with INGRES, as supplied by Digital Equipment Corporation of Maynard, MA.

Operation

Fig. 3 shows a first level database menu 100 for a specific power system 110. At this level, a user accesses type hierarchy 120, system parameters 130, grouping selection 140, and object lists 150.

A user makes a selection in menu 100, for example type hierarchy 120, and is directed to a type menu 200 for the desired type hierarchy 210. Type hierarchy includes a name field 230 which identifies an established hierarchy for a specified object 240, which is further defined as equipment 250 (subtype) of a type that is conducting 260 (sub-subtype). Additional hierarchy may be provided in the remaining field 270 and several other fields, as desired, beneath it.

Once type and type hierarchy are established, a type selection is made 280, as specifically defined by a name

field 290. The selection is made from the available types 300-350, for example, ground, generator, consumer, capacitor, reactor, or switch; and any additional subtypes 360.

5 A grouping selection menu 400 for a specified grouping 410 is shown in Fig. 5. A first level in the menu displays group hierarchy 420 by type 430 and by name 440. Thus, group hierarchy may be structured as an organization 431 named ORGANIZATION 441, having a system 432 named SYS 442, in which
10 there is a company 433 named CO 443, having a division 434 named DIV 444, containing a substation 435 named SUB 445, and so on.

For the group hierarchy, a group selection 450 is made,
15 including a type 460 having a name 470. The list of objects selected would represent equipment within substation SUB and would be selected by type from a scrollable list 461-469 in which each piece of equipment on the list has a corresponding name 471-479.

20

For each group selected, a user can also identify members of the group 480, equipment details 490, and network connections 495.

25 Objects within the system may be located from the find object menu 500 for a specific object 510, as shown in Fig. 6. Thus, an object type is selected 520 from a list of types 521-524, which may include switch, system, telemetry equipment, etc. The object selected is associated with a
30 group 530 having a relationship 532, such as 'contains,' 'starts with,' 'ends with,' 'matches,' and having a name 534. A corresponding name 540 also has a defined relation 542, such as 'contains' for a name 544. A look-up button 546

2143652

30

also provided to obtain a list of objects that match the search criteria.

5 The objects found are reported by number 550 and, by group 560, type 570, and name 580, each of which includes a corresponding list of matching objects (561-566, 571-576, and 581-586, respectively). Navigation through the menu is enhanced by an apply button 590 and a cancel button 595.

10 Although the invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that additional applications, other than those set forth herein, may be substituted therefor without departing from the spirit and scope of the present invention. For
15 example, the invention may be readily applied to other large, complex systems, including establishing models for process control in manufacturing plants and other physical systems. Accordingly, the invention should only be limited by the claims included below.

20

The following appendix provides source and object code listings and screen printouts for software included in this invention. The listings include database scheme definition, object editors for the power system data manager editor, and
25 general use frames for the power system data manager.

2143652

.....
Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.
.....

2143652

32

```

/*
*   The following domains are used in this model:
*
*   AbsoluteDate      A date in full dd-mm-yyyy hh:mm:ss form
*   Address           Address of component in computer system
*   Amps              Current flow (+ve into bus, -ve out of bus)
*   Bits              Count of bits
*   Boolean            0 = False = No | 1 = True = Yes
*   Comment            Free form text
*   ContactName        Textual name of a contact person
*   Coordinate         Measurement of position on a grid
*   Count              Measurement of quantity
*   Degrees/Step       Degrees of phase shift per tap step
*   Description         Textual description
*   Exponent           Exponentiation amount
*   ForeignKey          Reference to a primary key
*   Hertz              Cycles per second
*   Interpolation       Type of interpolation between data points
*   IntervalTime        A time between two points (usually hhh:mm:ss)
*   kV                 Value representing kV
*   KVLevel            Voltage level in kV
*   KeyName            Textual name of a key
*   KeyValue           Current value of a key
*   LocationLine        Text describing a location
*   LockCombination     Combination for a lock
*   LongLength          Long unit of length (e.g. mile; kilometer)
*   LongLength/ShortLength Ratio of long length to short length
*   MVA                Megavoltamperes
*   MVAR               Megavoltamperes Reactive
*   MW                 Megawatt
*   MW/min             Megawatt rate of change per minute
*   MeasurementSource   Reference to document defining measurement
*   Name               Textual name
*   ObjectName          Textual name of an object
*   Ohm-ShortLength     Resistivity in ohms per-unit short length
*   Ohms/LongLength     Resistance in ohms per unit long length
*   ParticipationFactor Generating unit economic participation factor
*   Percent             Normally 0 - 100 on a defined base
*   PerUnit             Normally 0 - 1 on a defined base
*   PerUnitMW-Sec       MW - second inertia on a defined base
*   PerUnitMW/Freq      MW variation with frequency on a defined base
*   PerUnitkV/MVAR      Voltage variation with reactive power
*   PhaseID             Phase identification (A|B|C|3)
*   PhoneNumber         Telephone number
*   PrimaryKey          Unique reference to a row
*   Problem             Identifier of a problem description
*   RatingType          Type of measurement being rated
*   Reference           Reference to association between two objects
*   SensorValue         Physical value measured by a sensor
*   SequenceNumber       Incrementing number used to order rows
*   SerialNumber        Textual serial number string
*   Severity            Problem severity (F|E|W|I)
*   ShortLength          Short unit of length (e.g. foot; meter)
*   SourceIdentifier     Reference to data source in foreign database
*   TapStep             Transformer tap step position
*   TelemetryValue       Value of reading in telemetry system
*   Temperature         Value of temperature in TemperatureUnits
*   TemperatureUnits    C | F
*   Terminal            Object terminal number
*   TerminalCount       Maximum number of terminals for an object
*   TypeName            Textual name for an object type
*   UserName            Textual name for a user
*   VersionNumber        Object version number
*   WindingName          Textual name for a transformer winding

```

2143652

33

```

/*
 *      Object table. There is an entry in this table for every
 *      Equipment, Organisation and Specification entry.
 */
CREATE TABLE Object
(
    Object          INTEGER          NOT NULL                /* PrimaryKey */
,   Source          INTEGER          WITH NULL              /* SourceIdentifier */
,   SystemVersion   INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
,   BaseType        INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
,   Type            INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
,   PrimaryGrouping INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
,   TypeName        VARCHAR(20)      NOT NULL WITH DEFAULT  /* TypeName */
,   PrimaryGroupingName VARCHAR(20)  NOT NULL WITH DEFAULT  /* ObjectName */
,   Name            VARCHAR(20)      NOT NULL WITH DEFAULT  /* ObjectName */
,   Description      VARCHAR(60)      NOT NULL WITH DEFAULT  /* Description */
,   Comment          VARCHAR(200)     NOT NULL WITH DEFAULT  /* Comment */
,   Terminals        INTEGER          NOT NULL WITH DEFAULT  /* TerminalCount */
,   InService        DATE             WITH NULL              /* AbsoluteDate */
,   OutService        DATE             WITH NULL              /* AbsoluteDate */
,   PositionX        REAL             WITH NULL              /* Coordinate */
,   PositionY        REAL             WITH NULL              /* Coordinate */
,   OwnerNumber      CHAR(20)         NOT NULL WITH DEFAULT  /* SerialNumber */
,   SerialNumber      CHAR(20)         NOT NULL WITH DEFAULT  /* SerialNumber */
,   SpecNumber       CHAR(20)         NOT NULL WITH DEFAULT  /* SerialNumber */
,   Created          DATE             WITH NULL              /* AbsoluteDate */
,   CreatedBy        CHAR(24)         NOT NULL WITH DEFAULT  /* UserName */
,   LastChanged      DATE             WITH NULL              /* AbsoluteDate */
,   LastChangedBy    CHAR(24)         NOT NULL WITH DEFAULT  /* UserName */
);

```

2143652

34

```

/*
 *      Conducting equipment.
 *      Each type of conducting equipment has a table containing
 *      specific details pertaining to that type of equipment alone.
 */
CREATE TABLE Capacitor
(   Object          INTEGER      NOT NULL WITH DEFAULT   /* ForeignKey */
,   NominalMVAR     REAL        WITH NULL                /* MVAR */
,   VoltSensitivity REAL        WITH NULL                /* PerUnit kV/MVAR */
,   AVRDelay        DATE        WITH NULL                /* IntervalTime */
);
/*
 *      Transmission lines are made up of two-terminal segments, as
 *      are jumpers. Busbars have single terminals.
 */
CREATE TABLE Conductor
(   Object          INTEGER      NOT NULL WITH DEFAULT   /* ForeignKey */
,   R               REAL        WITH NULL                /* PerUnit */
,   X               REAL        WITH NULL                /* PerUnit */
,   Bch             REAL        WITH NULL                /* PerUnit */
,   Length          REAL        WITH NULL                /* LongLength */
,   TowerType       INTEGER      NOT NULL WITH DEFAULT   /* ForeignKey */
,   GroundConductorType INTEGER    NOT NULL WITH DEFAULT /* ForeignKey */
,   PhaseConductorType INTEGER    NOT NULL WITH DEFAULT /* ForeignKey */
,   PhaseConductorCount INTEGER    NOT NULL WITH DEFAULT /* Count */
,   PhaseConductorSpacing REAL     NOT NULL WITH DEFAULT /* ShortLength */
);
/*
 *      Consumers (loads) are modeled as a fixed part and a variable part.
 *      The fixed part represents an unchanging portion of the load
 *      (e.g. station load). The variable part is represents the rest of
 *      the load. It is made up of a nominal load, plus voltage and frequency
 *      variations (Dommel eq. 94). This portion of the load is also subject
 *      to time variations defined by the load curve associated with the
 *      consumer. The percentage of the total load in the uppermost load
 *      area for which this consumer is a member is also stored, together
 *      with a reference to this uppermost load area. (Note that load areas
 *      are also modeled as consumers).
 */
CREATE TABLE Consumer
(   Object          INTEGER      NOT NULL WITH DEFAULT   /* ForeignKey */
,   Type            INTEGER      NOT NULL WITH DEFAULT   /* ForeignKey */
,   TopLoadArea     INTEGER      NOT NULL WITH DEFAULT   /* ForeignKey */
,   Pfixed          REAL        WITH NULL                /* MW */
,   Qfixed          REAL        WITH NULL                /* MVAR */
,   Pnom            REAL        WITH NULL                /* MW */
,   Qnom            REAL        WITH NULL                /* MVAR */
,   PowerFactor     REAL        WITH NULL                /* Cosine */
,   PfixedPct       REAL        WITH NULL                /* Percent */
,   QfixedPct       REAL        WITH NULL                /* Percent */
,   PnomPct         REAL        WITH NULL                /* Percent */
,   QnomPct         REAL        WITH NULL                /* Percent */
,   PVexp           REAL        WITH NULL                /* Exponent */
,   QVexp           REAL        WITH NULL                /* Exponent */
,   PFexp           REAL        WITH NULL                /* Exponent */
,   QFexp           REAL        WITH NULL                /* Exponent */
);

```

2143652

35

```

CREATE TABLE Generator
(
  Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
, RatedMVA        REAL         WITH NULL              /* MVA */
, MinimumMW       REAL         WITH NULL              /* MW */
, BaseMW          REAL         WITH NULL              /* MW */
, MaximumMW       REAL         WITH NULL              /* MW */
, MinimumMVAR     REAL         WITH NULL              /* MVAR */
, BaseMVAR        REAL         WITH NULL              /* MVAR */
, MaximumMVAR     REAL         WITH NULL              /* MVAR */
, MinimumKV       REAL         WITH NULL              /* kV */
, MaximumKV       REAL         WITH NULL              /* kV */
, X               REAL         WITH NULL              /* PerUnit */
, R               REAL         WITH NULL              /* PerUnit */
, Inertia         REAL         WITH NULL              /* PerUnit MW-Sec */
, Damping         REAL         WITH NULL              /* PerUnit MW/Freq */
, ManualToAVR     DATE         WITH NULL              /* IntervalTime */
, AVRTToManualLag DATE         WITH NULL              /* IntervalTime */
, AVRTToManualLead DATE        WITH NULL              /* IntervalTime */
, NormalPF        REAL         WITH NULL              /* ParticipationFact */
, ShortPF         REAL         WITH NULL              /* ParticipationFact */
, LongPF          REAL         WITH NULL              /* ParticipationFact */
, DownRampRate    REAL         WITH NULL              /* MW/min */
, UpRampRate      REAL         WITH NULL              /* MW/min */
);

CREATE TABLE Reactor
(
  Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
, NominalMVAR     REAL         WITH NULL              /* MVAR */
, VoltSensitivity REAL         WITH NULL              /* PerUnit kV/MVAR */
, AVRDelay        DATE         WITH NULL              /* IntervalTime */
);

CREATE TABLE Switch
(
  Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
, NormalOpen      INTEGER      NOT NULL WITH DEFAULT /* Boolean */
, CurrentlyOpen   INTEGER      NOT NULL WITH DEFAULT /* Boolean */
, RelayType       INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
);

```

2143652

36

```

/*
 *   The transformer table contains information relating to the transformer as
 *   a whole. These are mostly related to the magnetizing branch characteristics
 *   which are dependent on the core of the transformer.
 */

```

```

CREATE TABLE Transformer

```

```

(   Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Bank            INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Phases          CHAR(1)      NOT NULL WITH DEFAULT /* PhaseID */
,   MagBaseKV       REAL         WITH NULL              /* kV */
,   Gmag            REAL         WITH NULL              /* PerUnit */
,   Bmag            REAL         WITH NULL              /* PerUnit */
,   MagSatFlux      REAL         WITH NULL              /* PerUnit */
,   BmagSat         REAL         WITH NULL              /* PerUnit */
);

```

```

/*
 *   Each transformer is made up of a number of windings
 */

```

```

CREATE TABLE Winding

```

```

(   Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Winding         INTEGER      NOT NULL WITH DEFAULT /* SequenceNumber */
,   Source          INTEGER      NOT NULL WITH DEFAULT /* SourceIdentifier */
,   Name            CHAR(10)     NOT NULL WITH DEFAULT /* WindingName */
,   LoadTapChanger INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   NominalKV       REAL         NOT NULL              /* KVLevel */
,   NeutralKV       REAL         NOT NULL WITH DEFAULT /* kV */
,   BIL             REAL         WITH NULL              /* kV */
,   RatedMVA        REAL         WITH NULL              /* MVA */
,   ShortTermMVA    REAL         WITH NULL              /* MVA */
,   EmergencyMVA    REAL         WITH NULL              /* MVA */
,   LowStep         INTEGER      NOT NULL WITH DEFAULT /* TapStep */
,   HighStep        INTEGER      NOT NULL WITH DEFAULT /* TapStep */
,   NeutralStep     INTEGER      NOT NULL WITH DEFAULT /* TapStep */
,   StepSize        REAL         WITH NULL              /* Percent */
,   PhaseShift      REAL         WITH NULL              /* Degrees/Step */
);

```

```

/*
 * Each transformer may have a number of measurements done at different
 * tap settings. These can have two sources: the primary source is the
 * manufacturers test report, which contains impedance information in perce
 * measured at one winding with a reference winding shorted. The secondary
 * source are the impedances derived from this test report. These are
 * specified on a per-winding basis, corrected for system base MVA and
 * system base kV. These are needed by analytical applications, and are eit
 * derived from tthe primary, or loaded directly if the primary information
 * is not available.
 */

```

```

CREATE TABLE TapSetting

```

```

(
  Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
  Winding1        INTEGER      NOT NULL WITH DEFAULT /* SequenceNumber */
  TapStep1        INTEGER      NOT NULL WITH DEFAULT /* TapStep */
  Winding2        INTEGER      NOT NULL WITH DEFAULT /* SequenceNumber */
  TapStep2        INTEGER      NOT NULL WITH DEFAULT /* TapStep */
  Winding3        INTEGER      NOT NULL WITH DEFAULT /* SequenceNumber */
  TapStep3        INTEGER      NOT NULL WITH DEFAULT /* TapStep */
  Source          CHAR(20)     NOT NULL WITH DEFAULT /* MeasurementSource */
  R0_1_2          REAL         WITH NULL              /* PerCent */
  Z0_1_2          REAL         WITH NULL              /* PerCent */
  R1_1_2          REAL         WITH NULL              /* PerCent */
  Z1_1_2          REAL         WITH NULL              /* PerCent */
  R0_2_3          REAL         WITH NULL              /* PerCent */
  Z0_2_3          REAL         WITH NULL              /* PerCent */
  R1_2_3          REAL         WITH NULL              /* PerCent */
  Z1_2_3          REAL         WITH NULL              /* PerCent */
  R0_3_1          REAL         WITH NULL              /* PerCent */
  Z0_3_1          REAL         WITH NULL              /* PerCent */
  R1_3_1          REAL         WITH NULL              /* PerCent */
  Z1_3_1          REAL         WITH NULL              /* PerCent */
  R0_1            REAL         WITH NULL              /* PerCent */
  Z0_1            REAL         WITH NULL              /* PerCent */
  X0_1            REAL         WITH NULL              /* PerCent */
  R1_1            REAL         WITH NULL              /* PerCent */
  Z1_1            REAL         WITH NULL              /* PerCent */
  X1_1            REAL         WITH NULL              /* PerCent */
  R0_2            REAL         WITH NULL              /* PerCent */
  Z0_2            REAL         WITH NULL              /* PerCent */
  X0_2            REAL         WITH NULL              /* PerCent */
  R1_2            REAL         WITH NULL              /* PerCent */
  Z1_2            REAL         WITH NULL              /* PerCent */
  X1_2            REAL         WITH NULL              /* PerCent */
  R0_3            REAL         WITH NULL              /* PerCent */
  Z0_3            REAL         WITH NULL              /* PerCent */
  X0_3            REAL         WITH NULL              /* PerCent */
  R1_3            REAL         WITH NULL              /* PerCent */
  Z1_3            REAL         WITH NULL              /* PerCent */
  X1_3            REAL         WITH NULL              /* PerCent */
);

```

2143652

38

```

/*
 *      Sensing Equipment.
 */
CREATE TABLE Telemetry
(
    Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   TelemetryMinimum  INTEGER      NOT NULL WITH DEFAULT /* TelemetryValue */
,   TelemetryMaximum  INTEGER      NOT NULL WITH DEFAULT /* TelemetryValue */
,   XORMask          INTEGER      NOT NULL WITH DEFAULT /* TelemetryValue */
,   Input            INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   Output           INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   CardAddress       INTEGER      NOT NULL WITH DEFAULT /* Address */
,   WordAddress       INTEGER      NOT NULL WITH DEFAULT /* Address */
,   Size             INTEGER      NOT NULL WITH DEFAULT /* Bits */
,   Offset           INTEGER      NOT NULL WITH DEFAULT /* Bits */
,   PowerCable        CHAR(20)     NOT NULL WITH DEFAULT /* Name */
,   ReturnCable       CHAR(20)     NOT NULL WITH DEFAULT /* Name */
,   GroundCable       CHAR(20)     NOT NULL WITH DEFAULT /* Name */
,   TerminationCable CHAR(20)     NOT NULL WITH DEFAULT /* Name */
,   Panel            CHAR(20)     NOT NULL WITH DEFAULT /* Name */
);
/*
 *      Computer Equipment.
 */
CREATE TABLE Computer
(
    Object          INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   ObjectAddress    INTEGER      NOT NULL WITH DEFAULT /* Address */
,   Controller       CHAR(20)     NOT NULL WITH DEFAULT /* Name */
,   ControllerAddress INTEGER      NOT NULL WITH DEFAULT /* Address */
,   Cable           CHAR(20)     NOT NULL WITH DEFAULT /* Name */
);

```

2143652

39

```

/*
 *      Specification objects
 */
/*
 *      Conductor details from the manufacturer. Characteristic of the conductor
 *      as supplied by the manufacturer are tabulated.
 */
CREATE TABLE ConductorType
(
    Object          INTEGER          NOT NULL WITH DEFAULT /* ForeignKey */
,   Resistance      REAL            NOT NULL WITH DEFAULT /* Ohms/LongLength */
,   Radius          REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   GMR             REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   Ampacity        REAL            NOT NULL WITH DEFAULT /* Amps */
);
/*
 *      Tower geometry is tabulated for a three phase, single or double circuit,
 *      with 0, 1, or 2 ground wires. Offsets from an arbitrary datum, and
 *      the height above ground are used.
 */
CREATE TABLE TowerType
(
    Object          INTEGER          NOT NULL WITH DEFAULT /* ForeignKey */
,   A1_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   A1_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   B1_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   B1_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   C1_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   C1_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   G1_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   G1_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   A2_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   A2_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   B2_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   B2_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   C2_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   C2_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   G2_Offset       REAL            NOT NULL WITH DEFAULT /* ShortLength */
,   G2_Height       REAL            NOT NULL WITH DEFAULT /* ShortLength */
);
/*
 *      The Person table holds specific attributes for people who need
 *      to be known to the system.
 */
CREATE TABLE Person
(
    Object          INTEGER          NOT NULL WITH DEFAULT /* ForeignKey */
,   LoginName       CHAR(20)        NOT NULL WITH DEFAULT /* Name */
,   Password        CHAR(20)        NOT NULL WITH DEFAULT /* Name */
,   Initials        CHAR(4)         NOT NULL WITH DEFAULT /* Initials */
);

```

2143652

40

```

/*
 * Object typing
 */
/*
 * The LastKey table contains the last key value used for the different
 * key types used in the system. Each key is an integer value that increments
 * with each use. The key values start at 10000 for user-defined entities.
 * System-defined entities are loaded on database initialization with key
 * values that are less than 10000.
 */
CREATE TABLE LastKey
(
    LastKey      CHAR(8)      NOT NULL      /* KeyName */
    , Value      INTEGER      NOT NULL      /* KeyValue */
);
/*
 * List all problems and their descriptions
 */
CREATE TABLE Problem
(
    Problem      CHAR(20)     NOT NULL      /* Problem */
    , Severity    CHAR(1)     NOT NULL WITH DEFAULT /* Severity */
    , Description  CHAR(60)    NOT NULL WITH DEFAULT /* Description */
);
/*
 * Set up the base type table. This is used to set up base type
 * editors and specific table names
 */
CREATE TABLE BaseType
(
    Type          INTEGER      NOT NULL      /* ForeignKey */
    , EditorName   CHAR(20)     WITH NULL     /* Name */
    , TableName    CHAR(20)     WITH NULL     /* Name */
);
/*
 * Set up the type validation table. This table is used to specify the
 * valid type names.
 */
CREATE TABLE Type
(
    Type          INTEGER      NOT NULL      /* PrimaryKey */
    , BaseType     INTEGER      NOT NULL      /* ForeignKey */
    , Name         CHAR(20)     NOT NULL      /* TypeName */
);
/*
 * Table containing type - supertype relationships. This table is used to
 * define the SuperType to SubType heirarchy.
 */
CREATE TABLE SuperType
(
    Type          INTEGER      NOT NULL      /* ForeignKey */
    , SuperType    INTEGER      NOT NULL      /* ForeignKey */
);
/*
 * Table containing extended type - supertype relationships. This table
 * is a flattened version of the SuperType table. Each type is cataloged
 * with all its SuperTypes.
 */
CREATE TABLE ExtendedType
(
    Type          INTEGER      NOT NULL      /* ForeignKey */
    , SuperType    INTEGER      NOT NULL      /* ForeignKey */
);

```

2143652

41

```

/*
 * Parameters relating to the system being modelled are defined here.
 *
 * Frequency is the system base frequency (default 60)
 * MVABase is the system MVA base for per-unit transformations (default 100)
 * KVReference is the kV to use for per-unit transformations (default 1)
 * LengthRatio is the ratio of the Longlength over the ShortLength
 * Typical values are: 1000 for SI units (km / m) (default)
 * 5280 for British-American units (mile / foot)
 * GroundResistivity is used in zero sequence calculations (default 100 ohm)
 */
CREATE TABLE SystemParameters
(
    Frequency          REAL          NOT NULL          /* Hertz */
,
    MVABase            REAL          NOT NULL          /* MVA */
,
    KVReference        REAL          NOT NULL          /* kV */
,
    LengthRatio        REAL          NOT NULL          /* LongLength/ShortL
,
    GroundResistivity  REAL          NOT NULL          /* Ohm-ShortLength */
,
    Temperature        CHAR(1)      NOT NULL          /* TemperatureUnits
);
/*
 * Validation table for system kV levels
 */
CREATE TABLE KVLevel
(
    KVLevel            REAL          NOT NULL          /* KVLevel */
,
    Name               CHAR(20)     NOT NULL WITH DEFAULT /* Name */
,
    Voltage            REAL          NOT NULL          /* KVLevel */
);

```

2143652

42

```

/*
 *   Locations of objects (e.g. address). This table is keyed by the object
 *   which defines the location (typically of type Support Equipment or
 *   Organization), but may be referenced by any object
 */
CREATE TABLE Location
(
  Object          INTEGER          NOT NULL                /* ForeignKey */
, Combination    CHAR(4)          NOT NULL WITH DEFAULT  /* LockCombination */
, Phone          VARCHAR(20)      NOT NULL WITH DEFAULT  /* PhoneNumber */
, Contact        VARCHAR(40)      NOT NULL WITH DEFAULT  /* ContactName */
, L1             VARCHAR(80)      NOT NULL WITH DEFAULT  /* LocationLine */
, L2             VARCHAR(80)      NOT NULL WITH DEFAULT  /* LocationLine */
, L3             VARCHAR(80)      NOT NULL WITH DEFAULT  /* LocationLine */
, L4             VARCHAR(80)      NOT NULL WITH DEFAULT  /* LocationLine */
, Comment        VARCHAR(200)     NOT NULL WITH DEFAULT  /* Comment */
);
/*
 *   Set up the object rating table. Any object can have a set of ratings
 *   associated with it. The rating type defines the characteristic being
 *   rated (e.g. Amps, MVA, MW). Four values can be supplied: it is assumed
 *   that the ordering is Normal < ShortTerm < Emergency < Loadshed.
 */
CREATE TABLE Rating
(
  Object          INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
, Type           CHAR(10)         NOT NULL WITH DEFAULT  /* RatingType */
, Season         INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
, Temperature     REAL            WITH NULL              /* Temperature */
, Normal         REAL            WITH NULL
, ShortTerm      REAL            WITH NULL
, Emergency      REAL            WITH NULL
, Loadshed      REAL            WITH NULL
);
/*
 *   LoadCurves are used to store the time variation of a load class with
 *   respect to a base value. Families of load curves can be constructed for
 *   different seasons and day types (e.g Summer Weekend; Winter Weekday).
 *   The values supplied for the curve are interpolated as directed. Load
 *   curves are constructed for an entire day (00:00:00 to 23:59:59), and
 *   can be put together by applications to generate a profile as required.
 */
CREATE TABLE LoadCurve
(
  Object          INTEGER          WITH NULL              /* ForeignKey */
, DayType        INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
, Season         INTEGER          NOT NULL WITH DEFAULT  /* ForeignKey */
, Temperature     REAL            WITH NULL              /* Temperature */
, LoadCurve      INTEGER          NOT NULL WITH DEFAULT  /* PrimaryKey */
, Source         INTEGER          WITH NULL              /* SourceIdentifier */
, Interpolate     CHAR(6)         NOT NULL WITH DEFAULT  /* Interpolation */
);
CREATE TABLE LoadValue
(
  LoadCurve      INTEGER          NOT NULL WITH DEFAULT  /* PrimaryKey */
, Time           DATE             NOT NULL WITH DEFAULT  /* IntervalTime */
, LoadValue      REAL            NOT NULL WITH DEFAULT  /* PerUnit */
, Source         INTEGER          WITH NULL              /* SourceIdentifier */
);

```

2143652

43

```

/*
 *      Interactions
 */
/*
 *      Equipment terminals. Each object can have a variable number of terminals
 *      which can be connected to other terminals. The node is used as a
 *      common reference between connected terminals.
 */
CREATE TABLE Terminal
(
    Object          INTEGER      NOT NULL          /* ForeignKey */
,   BaseType       INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Terminal       INTEGER      NOT NULL          /* Terminal */
,   Node           INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   KVLevel        REAL         NOT NULL WITH DEFAULT /* KVLevel */
);
/*
 *      Measurements made on the equipment. Each measurement measures
 *      something (the measurand) using an agent (the measurer).
 *
 *      The measurand has a type (e.g. kV, Status, MW, Amps) that defines
 *      the quantity being measured. It refers to the object being measured,
 *      and may be made at no terminal (e.g. fire alarm), at one terminal
 *      (e.g. voltage), or between two terminals (e.g. breaker status).
 *
 *      The measurand may be represented by a sensor. This has a type
 *      (e.g. current, voltage, frequency, position, impulse, contact) and
 *      a minimum and maximum that are expressed in the units of the
 *      measurand.
 *
 *      The measurer has a type (e.g. telemetered, manual, state estimated,
 *      calculated, from other system) that defines the point of entry of the
 *      measurement. It may refer to an object that is present at that point
 *      of entry (e.g. a telemetry input).
 */
CREATE TABLE Measurement
(
    Measurand       INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   MeasurandType   INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Terminal1       INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Terminal2       INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   SensorType      INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   SensorMinimum   REAL         NOT NULL WITH DEFAULT /* SensorValue */
,   SensorMaximum   REAL         NOT NULL WITH DEFAULT /* SensorValue */
,   SensorAccuracy  REAL         NOT NULL WITH DEFAULT /* PerCent */
,   ReversePolarity INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   MeasurerType    INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Measurer        INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Alarm           INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   History         INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   Trigger         INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   EventLog        INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   Input           INTEGER      NOT NULL WITH DEFAULT /* Boolean */
,   Output          INTEGER      NOT NULL WITH DEFAULT /* Boolean */
);
/*
 *      Set up the grouping validation table. This contains the group - member
 *      relationships for objects.
 */
CREATE TABLE Grouping
(
    SystemVersion   INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Grouping        INTEGER      NOT NULL          /* ForeignKey */
,   GroupingType    INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Member          INTEGER      NOT NULL          /* ForeignKey */
,   MemberType      INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Relationship    INTEGER      NOT NULL WITH DEFAULT /* ForeignKey */
,   Reference       CHAR(10)     NOT NULL WITH DEFAULT /* Reference */
);

```

2143652

44

```

/*
 *      Grass catcher table - somewhere to throw all the problems
 */
CREATE TABLE GrassCatcher
(
    Object          INTEGER          NOT NULL          /* ForeignKey */
  , Problem        CHAR(20)         NOT NULL          /* ForeignKey */
  , Text           CHAR(20)         NOT NULL WITH DEFAULT /* Text */
  , Known          INTEGER          NOT NULL WITH DEFAULT /* Boolean */
);
/*
 *      Temporary tables used as workspace for procedures embedded in
 *      the database
 */
/*
 *      List of terminals connected to objects (identical to Terminal table)
 */
CREATE TABLE TEMP$TerminalList
(
    Object          INTEGER          NOT NULL          /* ForeignKey */
  , BaseType       INTEGER          NOT NULL WITH DEFAULT /* ForeignKey */
  , Terminal       INTEGER          NOT NULL          /* Terminal */
  , Node           INTEGER          NOT NULL WITH DEFAULT /* ForeignKey */
  , KVLevel        REAL             NOT NULL WITH DEFAULT /* KVLevel */
);
/*
 *      List of object and their voltage levels.
 */
CREATE TABLE TEMP$ObjectList
(
    Object          INTEGER          NOT NULL          /* ForeignKey */
  , KVLevel         REAL             NOT NULL WITH DEFAULT /* KVLevel */
);
/*
 *      Table to use as an audit trail of where we have been
 */
CREATE TABLE TEMP$AuditTrail
(
    Object          INTEGER          NOT NULL          /* ForeignKey */
);

```

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

MODIFY Object TO BTree UNIQUE
ON Object;

MODIFY Capacitor TO BTree UNIQUE
ON Object;

MODIFY Conductor TO BTree UNIQUE
ON Object;

MODIFY Consumer TO BTree UNIQUE
ON Object;

MODIFY Generator TO BTree UNIQUE
ON Object;

MODIFY Reactor TO BTree UNIQUE
ON Object;

MODIFY Switch TO BTree UNIQUE
ON Object;

MODIFY Transformer TO BTree UNIQUE
ON Object;

MODIFY Winding TO BTree UNIQUE
ON Object, Winding;

MODIFY TapSetting TO BTree UNIQUE
ON Object, Winding1, TapStep1, Winding2, TapStep2;

MODIFY Telemetry TO BTree UNIQUE
ON Object;

MODIFY Computer TO BTree UNIQUE
ON Object;

MODIFY ConductorType TO BTree UNIQUE
ON Object;

MODIFY TowerType TO BTree UNIQUE
ON Object;

MODIFY Person TO BTree UNIQUE
ON Object;

MODIFY Problem TO BTree UNIQUE
ON Problem;

2143652

46

```
MODIFY BaseType TO BTree UNIQUE
    ON Type;

MODIFY Type TO BTree UNIQUE
    ON Type, Name;

MODIFY SuperType TO BTree UNIQUE
    ON Type, SuperType;

MODIFY ExtendedType TO BTree UNIQUE
    ON Type, SuperType;

MODIFY KVLevel TO BTree UNIQUE
    ON KVLevel;

MODIFY Location TO BTree UNIQUE
    ON Object;

MODIFY Rating TO BTree UNIQUE
    ON Object, Type;

MODIFY LoadCurve TO BTree UNIQUE
    ON Object, DayType, Season, Temperature;

MODIFY LoadValue TO BTree UNIQUE
    ON LoadCurve, Time;

MODIFY Terminal TO BTree UNIQUE
    ON Object, Terminal, Node;

MODIFY Measurement TO Btree UNIQUE
    ON Measurand, MeasurandType, Terminal1, Terminal2;

MODIFY Grouping TO BTree UNIQUE
    ON Grouping, Relationship, Member;

MODIFY GrassCatcher TO BTree    /* Also done in GrassCatcher frame */
    ON Object, Problem;
```

2143652

47

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```
CREATE UNIQUE INDEX Object$1
  ON Object (Type, PrimaryGroupingName, Name)
  WITH STRUCTURE = BTREE;

CREATE UNIQUE INDEX Object$2
  ON Object (PrimaryGrouping, Object)
  WITH STRUCTURE = BTREE;

CREATE UNIQUE INDEX LoadCurve$1 ON LoadCurve
  (LoadCurve)
  WITH STRUCTURE = BTree;

CREATE UNIQUE INDEX ExtendedType$1
  ON ExtendedType (SuperType, Type)
  WITH STRUCTURE = BTREE;

CREATE UNIQUE INDEX Terminal$1 ON Terminal
  (Node, Object, Terminal)
  WITH STRUCTURE = BTree;

CREATE UNIQUE INDEX Grouping$1 ON Grouping
  (Member, Relationship, Grouping)
  WITH STRUCTURE = BTree;
```

2143652

48

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 * This file contain rules that fire in order to trigger external
 * notification events to interested parties. The events themselves,
 * plus a generic procedure to raise them, are also defined here.
 *
 * The rules in this file are not required to be active. Therefore
 * every rule that is defined in this file must have a corresponding
 * entry in the file "drop_notification.sql", which can be
 * executed to drop the rules when desired. A awk program,
 * create_to_drop.awk exists for this purpose, and can be run
 * with the command:
 *
 * "awk -f create_to_drop.awk\
 *   create_notification.sql\
 *   > drop_notification.sql"
 */

/*
 * Define the events that can be raised
 */

CREATE DBEVENT Udb_ChangeName;
CREATE DBEVENT Udb_ChangeObject;
CREATE DBEVENT Udb_ChangeElectrical;
CREATE DBEVENT Udb_ChangeConnectivity;

/*
 * Handle changes to the object record
 */
CREATE PROCEDURE Udb_Change$Name
(
  PrimaryGroupingName  VARCHAR(20) NOT NULL
,
  TypeName             VARCHAR(20) NOT NULL
,
  Name                 VARCHAR(20) NOT NULL
) =
  DECLARE
    EventText          VARCHAR(256) NOT NULL;
  BEGIN
    EventText = SQUEEZE (PrimaryGroupingName + ' ' + TypeName + ' ' + Name
    RAISE DBEVENT Udb_ChangeName :EventText;
  END;

CREATE RULE Udb_Change$Name
  AFTER INSERT, UPDATE ON Object
  WHERE   Old.PrimaryGroupingName != New.PrimaryGroupingName OR
         Old.TypeName != New.TypeName OR
         Old.Name != New.Name
  EXECUTE PROCEDURE Udb_Change$Name

```

49

```

(   PrimaryGroupingName = New.PrimaryGroupingName
,   TypeName = New.TypeName
,   Name = New.Name
);

CREATE PROCEDURE Udb_Change$Object
(   Tid          INTEGER
) =
DECLARE
    PrimaryGroupingName VARCHAR(20) NOT NULL;
    TypeName             VARCHAR(20) NOT NULL;
    Name                 VARCHAR(20) NOT NULL;
    EventText            VARCHAR(256) NOT NULL;
BEGIN
    SELECT   :PrimaryGroupingName = O.PrimaryGroupingName,
             :TypeName = O.TypeName,
             :Name = O.Name
    FROM     Object O
    WHERE    :Tid = O.Tid;
    EventText = SQUEEZE (PrimaryGroupingName + ' ' + TypeName + ' ' + Name);
    RAISE DBEVENT Udb_ChangeObject :EventText;
END;

CREATE RULE Udb_Change$Object
AFTER INSERT, UPDATE ON Object
WHERE    Old.Description != New.Description OR
         Old.Comment != New.Comment OR
         Old.Terminals != New.Terminals OR
         Old.InService != New.InService OR
         Old.OutService != New.OutService OR
         Old.PositionX != New.PositionX OR
         Old.PositionY != New.PositionY OR
         Old.OwnerNumber != New.OwnerNumber OR
         Old.SerialNumber != New.SerialNumber OR
         Old.SpecNumber != New.SpecNumber
EXECUTE PROCEDURE Udb_Change$Object
(   Tid = New.Tid
);

/*
*   Conducting plant
*/
CREATE PROCEDURE Udb_Change$Electrical
(   Object          INTEGER
) =
DECLARE
    PrimaryGroupingName VARCHAR(20) NOT NULL;
    TypeName             VARCHAR(20) NOT NULL;
    Name                 VARCHAR(20) NOT NULL;
    EventText            VARCHAR(256) NOT NULL;
BEGIN
    SELECT   :PrimaryGroupingName = O.PrimaryGroupingName,
             :TypeName = O.TypeName,
             :Name = O.Name
    FROM     Object O
    WHERE    :Object = O.Object;
    EventText = SQUEEZE (PrimaryGroupingName + ' ' + TypeName + ' ' + Name);
    RAISE DBEVENT Udb_ChangeElectrical :EventText;
END;

CREATE RULE Udb_Change$Capacitor
AFTER INSERT, UPDATE ON Capacitor
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

CREATE RULE Udb_Change$Conductor

```

2143652

50

```

AFTER INSERT, UPDATE ON Conductor
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$Consumer
AFTER INSERT, UPDATE ON Consumer
WHERE   Old.Pfixed != New.Pfixed OR
        Old.Qfixed != New.Qfixed OR
        Old.Pnom  != New.Pnom  OR
        Old.Qnom  != New.Qnom  OR
        Old.PVexp != New.PVexp OR
        Old.QVexp != New.QVexp OR
        Old.PFexp != New.PFexp OR
        Old.QFexp != New.QFexp
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$Generator
AFTER INSERT, UPDATE ON Generator
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$Reactor
AFTER INSERT, UPDATE ON Reactor
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$Switch
AFTER INSERT, UPDATE ON Switch
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$Transformer
AFTER INSERT, UPDATE ON Transformer
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$Winding
AFTER INSERT, UPDATE ON Winding
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

CREATE RULE Udb_Change$TapSetting
AFTER INSERT, UPDATE ON TapSetting
EXECUTE PROCEDURE Udb_Change$Electrical
(   Object = New.Object
);

```

```

/*
 *   Specifications
 */

```

```

/*
 *   Interactions
 */

```

```

CREATE PROCEDURE Udb_Change$Connectivity
(   Object          INTEGER
) =

```

```

    DECLARE
        PrimaryGroupName VARCHAR(20) NOT NULL;

```

2143652

```

                                     51
      TypeName      VARCHAR(20) NOT NULL;
      Name          VARCHAR(20) NOT NULL;
      EventText     VARCHAR(256) NOT NULL;
BEGIN
      SELECT  :PrimaryGroupingName = O.PrimaryGroupingName,
              :TypeName = O.TypeName,
              :Name = O.Name
      FROM    Object O
      WHERE   :Object = O.Object;
      EventText = SQUEEZE (PrimaryGroupingName + ' ' + TypeName + ' ' + Name
      RAISE DBEVENT Udb_ChangeConnectivity :EventText;
END;

CREATE RULE Udb_Change$Terminal
AFTER INSERT, UPDATE ON Terminal
WHERE   Old.Node != New.Node
EXECUTE PROCEDURE Udb_Change$Connectivity
(      Object = New.Object
);
```

2143652

52

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

```

*****

/*
 *      Generic reject procedure
 */
CREATE PROCEDURE Reject (Problem VARCHAR(20) NOT NULL) =
  BEGIN
    INSERT INTO GrassCatcher (Object, Problem)
      VALUES (0, :Problem);
  END;

/*
 *      Object table
 */
/*
 *      Sort out an object on insert
 */
CREATE PROCEDURE Object$Insert
(
  Tid          INTEGER
,
  Object       INTEGER
,
  SystemVersion INTEGER
,
  Type        INTEGER
,
  PrimaryGrouping INTEGER
,
  Terminals   INTEGER
) =
  DECLARE
    BaseType    INTEGER;
    TableName   VARCHAR(20);
    Loop        INTEGER;
    AddGrouping INTEGER;

  BEGIN
    /*
     *      Get the base type from the type. If neither is good, reject
     *      the insert.
     */
    SELECT :BaseType = BT.Type,
           :TableName = BT.TableName
    FROM   Type T,
           BaseType BT
    WHERE  :Type = T.Type AND
           T.BaseType = BT.Type;

    IF BaseType = 0 THEN
      INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:Object, 'Object_012');
    ENDIF;

    /*
     *      Set up the key for the object if one was not supplied, and ensure
     *      that the object is in its primary group

```

53

```

*/
AddGrouping = 0;

IF Object = -1 THEN
    UPDATE LastKey K
    SET Value = K.Value + 1
    WHERE K.LastKey = 'Object';

    SELECT :Object = K.Value
    FROM LastKey K
    WHERE K.LastKey = 'Object';

    IF PrimaryGrouping != 0 THEN
        AddGrouping = 1;
    ENDIF;

ENDIF;

/*
* Set the object key and base type.
* Get the type name from the type record and
* get the primary grouping name from its object record.
* Set up the creation date and user
*/

UPDATE Object O
FROM Type T,
      Object PG
SET Object = :Object,
    BaseType = :BaseType,
    TypeName = T.Name,
    PrimaryGroupName = PG.Name,
    Created = 'Now',
    CreatedBy = VARCHAR (User)
WHERE :Type = T.Type AND
      :PrimaryGrouping = PG.Object AND
      :Tid = O.Tid;

IF AddGrouping = 1 THEN
    INSERT INTO Grouping
        (SystemVersion
        ,Grouping
        ,Member
        ,Relationship
        )
    VALUES
        (:SystemVersion
        ,:PrimaryGrouping
        ,:Object
        ,52 /* Type$member_of */
        );
ENDIF;

/*
* Insert terminals for the object
*/

Loop = 1;

WHILE Loop <= Terminals DO
    INSERT INTO Terminal
        (Object
        ,BaseType
        ,Terminal
        )
    VALUES
        (:Object
        ,:BaseType
        ,:Loop
        );

```

2143652

54

```

        Loop = Loop + 1;
    ENDWHILE;

/*
 *
 * Insert the associated type specific row (if any)
 */
    IF TableName = '' THEN
        TableName = TableName; /* Do nothing */
    ELSEIF TableName = 'Generator' THEN
        INSERT INTO Generator (Object) VALUES (:Object);
    ELSEIF TableName = 'Consumer' THEN
        INSERT INTO Consumer (Object) VALUES (:Object);
    ELSEIF TableName = 'Capacitor' THEN
        INSERT INTO Capacitor (Object) VALUES (:Object);
    ELSEIF TableName = 'Reactor' THEN
        INSERT INTO Reactor (Object) VALUES (:Object);
    ELSEIF TableName = 'Switch' THEN
        INSERT INTO Switch (Object) VALUES (:Object);
    ELSEIF TableName = 'Transformer' THEN
        INSERT INTO Transformer (Object) VALUES (:Object);
    ELSEIF TableName = 'BreakerSensor' THEN
        TableName = TableName; /*** NYI ***/
    ELSEIF TableName = 'Telemetry' THEN
        INSERT INTO Telemetry (Object) VALUES (:Object);
    ELSEIF TableName = 'TowerType' THEN
        INSERT INTO TowerType (Object) VALUES (:Object);
    ELSEIF TableName = 'ConductorType' THEN
        INSERT INTO ConductorType (Object) VALUES (:Object);
    ELSEIF TableName = 'Conductor' THEN
        INSERT INTO Conductor (Object) VALUES (:Object);
    ELSEIF TableName = 'Computer' THEN
        INSERT INTO Computer (Object) VALUES (:Object);
    ELSEIF TableName = 'Person' THEN
        INSERT INTO Person (Object) VALUES (:Object);
    ENDIF;

END;

/*
 *
 * Do validation on an object
 */
CREATE PROCEDURE Object$Validate
(
    Object      INTEGER
,
    InService   DATE
,
    OutService  DATE
) =
BEGIN
    IF InService > OutService THEN
        INSERT INTO GrassCatcher (Object, Problem)
            VALUES (:Object, 'Object_001');
    ENDIF;

END;

/*
 *
 * Maintain the PrimaryGroupingName in the objects that reference
 * this one as their primary group
 */
CREATE PROCEDURE Object$Name
(
    Tid      INTEGER
,
    BaseType INTEGER
) =
BEGIN
    UPDATE Object O
    FROM   Object PG
    SET    PrimaryGroupingName = PG.Name
    WHERE  :Tid = PG.Tid AND
           PG.Object = O.PrimaryGrouping;

```

2143652

55

```

END;
/*
 *      Maintain the TypeName in the Object
 */
CREATE PROCEDURE Object$TypeName
(
    Tid      INTEGER
) =
BEGIN
    UPDATE Object O
    FROM     Type T
    SET      TypeName = T.Name
    WHERE    T.Type = O.Type AND
             O.Tid = :Tid;

END;
/*
 *      Ensure that the object always belongs to its primary group.
 *      If the object is a Consumer, then update the load model.
 */
CREATE PROCEDURE Object$PrimaryGrouping
(
    Tid              INTEGER
,   Object           INTEGER
,   SystemVersion   INTEGER
,   Old_PrimaryGrouping INTEGER
,   New_PrimaryGrouping INTEGER
) =
DECLARE
    References      INTEGER NOT NULL;
BEGIN
/*
 *      If there is a primary grouping defined, ensure that the
 *      object exists as a member of the group
 */
    IF New_PrimaryGrouping != 0 THEN

        SELECT  :References = COUNT(*)
        FROM    Grouping G
        WHERE    :New_PrimaryGrouping = G.Grouping AND
                 :Object = G.Member;

        IF References = 0 THEN
            INSERT INTO Grouping
                (SystemVersion
                ,Grouping
                ,Member
                ,Relationship
                )
            VALUES
                (:SystemVersion
                ,:New_PrimaryGrouping
                ,:Object
                ,52      /* Type$member_of */
                );
        ENDIF;
    ENDIF;

/*
 *      Ensure that the primary grouping name for the object is correct
 */
    UPDATE Object O
    FROM     Object PG
    SET      PrimaryGroupName = PG.Name
    WHERE    :New_PrimaryGrouping = PG.Object AND
             O.Tid = :Tid;
/*

```

2143652

56

```

*      Blindly update the load model. Nothing will happen unless this object
*      is a consumer and the primary grouping is also a consumer.
*      If this consumer has been detached from a higher level part of the
*      load model, set its percentages to 100.
*/

```

```

UPDATE Consumer LM
FROM Consumer C
SET   Pfixed = IFNULL(LM.PFixed,0) - IFNULL(C.Pfixed,0),
      Pnom   = IFNULL(LM.Pnom,0) - IFNULL(C.Pnom,0),
      Qfixed = IFNULL(LM.QFixed,0) - IFNULL(C.Qfixed,0),
      Qnom   = IFNULL(LM.Qnom,0) - IFNULL(C.Qnom,0)
WHERE :Object = C.Object AND
      :Old_PrimaryGrouping = LM.Object;

```

```

UPDATE Consumer LM
FROM Consumer C
SET   Pfixed = IFNULL(LM.PFixed,0) + IFNULL(C.Pfixed,0),
      Pnom   = IFNULL(LM.Pnom,0) + IFNULL(C.Pnom,0),
      Qfixed = IFNULL(LM.QFixed,0) + IFNULL(C.Qfixed,0),
      Qnom   = IFNULL(LM.Qnom,0) + IFNULL(C.Qnom,0)
WHERE :Object = C.Object AND
      :New_PrimaryGrouping = LM.Object;

```

```

IF IIRowCount = 0 THEN
  UPDATE Consumer C
  SET   PfixedPct = 100,
        PnomPct   = 100,
        QfixedPct = 100,
        QnomPct   = 100
  WHERE :Object = C.Object;
ENDIF;

```

```

END;

```

```

/*
*      Conducting plant validation
*/

```

```

CREATE PROCEDURE Conductor$Validate

```

```

(   Object      INTEGER
  ,   R          REAL
  ,   X          REAL
  ,   Bch        REAL
) =

```

```

  BEGIN

```

```

    IF (X / R) < 5 THEN
      INSERT INTO GrassCatcher (Object, Problem)
      VALUES (:Object, 'Conductor_001');
    ENDIF;

```

```

  END;

```

```

/*
*      Update the load model when a consumer is changed
*/

```

```

CREATE PROCEDURE Consumer$UpdateLoadModel

```

```

(   Object      INTEGER
  ,   Old_Pfixed REAL
  ,   New_Pfixed REAL
  ,   Old_Pnom   REAL
  ,   New_Pnom   REAL
  ,   Old_Qfixed REAL
  ,   New_Qfixed REAL
  ,   Old_Qnom   REAL
  ,   New_Qnom   REAL
) =

```

```

  BEGIN

```

```

/*

```

2143652

57

```

*      Recursively update the hierarchy of consumers in the load model
*      from the bottom up. This will stop when an object that is not a
*      consumer is encountered
*/

```

```

UPDATE Consumer LM
FROM Object O
SET   Pfixed = IFNULL(LM.Pfixed,0) +
           (IFNULL(New_Pfixed,0) - IFNULL(Old_Pfixed,0)),
       Pnom = IFNULL(LM.Pnom,0) +
           (IFNULL(New_Pnom,0) - IFNULL(Old_Pnom,0)),
       Qfixed = IFNULL(LM.Qfixed,0) +
           (IFNULL(New_Qfixed,0) - IFNULL(Old_Qfixed,0)),
       Qnom = IFNULL(LM.Qnom,0) +
           (IFNULL(New_Qnom,0) - IFNULL(Old_Qnom,0))
WHERE :Object = O.Object AND
       O.PrimaryGrouping = LM.Object;

```

```

/*
*      If we have hit the top this consumer represents 100% of the
*      load in its hierarchy. Set this up to trigger a re-calculation
*      of all the child consumer percentages.
*/

```

```

IF IIRowCount = 0 THEN
    UPDATE Consumer C
    SET   PfixedPct = 100,
          PnomPct = 100,
          QfixedPct = 100,
          QnomPct = 100,
          TopLoadArea = C.Object
    WHERE :Object = C.Object;
ENDIF;

```

```

END;

```

```

/*
*      Update the distribution of the loads by percentage from the
*      top down.
*/

```

```

CREATE PROCEDURE Consumer$UpdateLoadPct

```

```

(   Object          INTEGER
  ,   Pfixed         REAL
  ,   Pnom           REAL
  ,   Qfixed         REAL
  ,   Qnom           REAL
  ,   PfixedPct      REAL
  ,   PnomPct        REAL
  ,   QfixedPct      REAL
  ,   QnomPct        REAL
  ,   TopLoadArea    INTEGER
) =

```

```

BEGIN

```

```

/*
*      If there is the possibility of a divide by zero, then use a trash
*      (but safe) value as the divisor, and force the result to zero
*/

```

```

IF ABS(Pfixed) < 0.0000001 THEN
    PFixed = 1;
    PFixedPct = 0;
ENDIF;

```

```

IF ABS(Pnom) < 0.0000001 THEN
    PNom = 1;
    PNomPct = 0;
ENDIF;

```

```

IF ABS(Qfixed) < 0.0000001 THEN
    QFixed = 1;
    QFixedPct = 0;

```

2143652

58

```

ENDIF;

IF ABS(Qnom) < 0.0000001 THEN
    QNom = 1;
    QNomPct = 0;
ENDIF;

/*
 * Recalculate the child percentages
 */
UPDATE Consumer C
FROM Object O
SET
    PfixedPct = (C.Pfixed / :Pfixed) * :PfixedPct,
    PnomPct = (C.Pnom / :Pnom) * :PnomPct,
    QfixedPct = (C.Qfixed / :Qfixed) * :QfixedPct,
    QnomPct = (C.Qnom / :Qnom) * :QnomPct,
    TopLoadArea = :TopLoadArea
WHERE
    :Object = O.PrimaryGrouping AND
    O.Object = C.Object;

END;

/*
 * Ensure that the winding belongs to a transformer and
 * get the next sequence number
 */
CREATE PROCEDURE Winding$Insert
(
    Tid          INTEGER
,   Object       INTEGER
,   KVLevel     REAL
) =
    DECLARE
        BaseType      INTEGER;
        References    INTEGER      NOT NULL;
        MaxSeq        INTEGER;

    BEGIN

        SELECT :BaseType = O.BaseType
        FROM Object O
        WHERE :Object = O.Object;

        IF BaseType != 27 THEN /* Type$Transformer */
            INSERT INTO GrassCatcher (Object, Problem)
            VALUES (:Object, 'Transformer_008');
        ENDIF;

        /*
         * Get a sequence number for the winding
         */
        SELECT MaxSeq = MAX(Winding)
        FROM Winding
        WHERE Object = :Object;

        IF MaxSeq IS NULL THEN
            MaxSeq = 1;
        ELSE
            MaxSeq = MaxSeq + 1;
        ENDIF;

        /*
         * Update the number of terminals for the transformer
         */
        UPDATE Object
        SET Terminals = Terminals + 1
        WHERE Object = :Object;

        INSERT INTO Terminal
        (Object
        ,BaseType
        ,Terminal

```

2143652

59

```

        ,KVLevel
    )
VALUES
    (:Object
    ,:BaseType
    ,:MaxSeq
    ,:KVLevel
    );
/*
 *   Update the winding last cos it can change the Tid
 */
UPDATE Winding
SET     Winding = :MaxSeq
WHERE   Tid = :Tid;

END;
/*
 *   Validate the attributes of a winding
 */
CREATE PROCEDURE Winding$Validate
(
    Object          INTEGER
  , Name           CHAR(10)
  , LoadTapChanger INTEGER
  , NominalKV      REAL
  , NeutralKV      REAL
  , LowStep        INTEGER
  , HighStep       INTEGER
  , NeutralStep    INTEGER
  , StepSize       REAL
) =
DECLARE
    Steps          INTEGER;
    KVDiff         REAL;

BEGIN

    IF LoadTapChanger != 0 AND LoadTapChanger != 1 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_003');
    ENDIF;

    IF LowStep > HighStep THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_004');
    ENDIF;

    IF NeutralStep < LowStep THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_005');
    ENDIF;

    IF NeutralStep > HighStep THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_006');
    ENDIF;

    IF StepSize = 0 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_007');
    ENDIF;

    IF NeutralKV = NominalKV THEN
        Steps = 0;
    ELSEIF NeutralKV > NominalKV THEN
        Steps = LowStep - NeutralStep; /* negative steps to get there */
    ELSEIF NeutralKV < NominalKV THEN

```

2143652

```

                                60
        Steps = HighStep - NeutralStep; /* positive steps to get there */
    ENDIF;

    KVDiff = ABS (Steps * StepSize * NeutralKV / 100);
    KVDiff = KVDiff + (NeutralKV * 0.10); /* Allow 10% overdrive */

    IF KVDiff < ABS(NeutralKV - NominalKV) THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_001');
    ENDIF;

    IF StepSize > 10 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
            VALUES (:Object, Name, 'Transformer_002');
    ENDIF;

END;

/*
 *   Clean up on delete of a winding
 */
CREATE PROCEDURE Winding$Delete
(
    Object          INTEGER
,
    Winding         INTEGER
) =
    BEGIN
/*
 *   Reduce the number of terminals for this transformer
 */
        DELETE FROM Terminal
        WHERE    Object = :Object AND
                Terminal = :Winding;

        UPDATE  Object
        SET      Terminals = Terminals - :IIRowCount
        WHERE    Object = :Object;

    END;

/*
 *   Validate the attributes for a tap setting
 */
CREATE PROCEDURE TapSetting$Validate
(
    Object          INTEGER
,
    Winding1        INTEGER
,
    TapStep1        INTEGER
,
    Winding2        INTEGER
,
    TapStep2        INTEGER
,
    Winding3        INTEGER
,
    TapStep3        INTEGER
) =
    DECLARE
        LowStep      INTEGER      WITH NULL;
        HighStep     INTEGER      WITH NULL;

    BEGIN

        LowStep = NULL;
        HighStep = NULL;

        SELECT  :LowStep = W.LowStep,
                :HighStep = W.HighStep
        FROM    Winding W
        WHERE    W.Object = :Object AND
                W.Winding = :Winding1;

        IF LowStep IS NULL AND HighStep IS NULL THEN
            INSERT INTO GrassCatcher (Object, Problem)

```

2143652

```

                                61
                                VALUES (:Object, 'Transformer_009');
ELSEIF TapStep1 < LowStep OR TapStep1 > HighStep THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:Object, 'Transformer_010');
ENDIF;

LowStep = NULL;
HighStep = NULL;

SELECT  :LowStep = W.LowStep,
        :HighStep = W.HighStep
FROM    Winding W
WHERE   W.Object = :Object AND
        W.Winding = :Winding2;

IF LowStep IS NULL AND HighStep IS NULL THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:Object, 'Transformer_009');
ELSEIF TapStep2 < LowStep OR TapStep2 > HighStep THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:Object, 'Transformer_010');
ENDIF;

LowStep = NULL;
HighStep = NULL;

SELECT  :LowStep = W.LowStep,
        :HighStep = W.HighStep
FROM    Winding W
WHERE   W.Object = :Object AND
        W.Winding = :Winding3;

IF LowStep IS NULL AND HighStep IS NULL THEN
    /* May not be a three way measurement */
    LowStep = HighStep; /* Empty statement blocks not allowed! */
ELSEIF TapStep3 < LowStep OR TapStep3 > HighStep THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:Object, 'Transformer_010');
ENDIF;

END;

/*
 * Specifications
 */
/*
 * Get the surrogate key set up on inserts
 */
CREATE PROCEDURE LoadCurve$Insert
(
    Tid      INTEGER
,
    LoadCurve  INTEGER
) =
BEGIN
    IF LoadCurve = -1 THEN
        UPDATE LastKey
        SET Value = Value + 1
        WHERE LastKey = 'Curves';

        UPDATE LoadCurve C
        FROM LastKey K
        SET LoadCurve = K.Value
        WHERE C.Tid = :Tid AND
              K.LastKey = 'Curves';
    ENDIF;
END;
END;
```

2143652

62

```

/*
 *      Check the referenced types exist on inserts
 */
CREATE PROCEDURE SuperType$Insert
(
    Type INTEGER
    , SuperType INTEGER
) =
    DECLARE
        References          INTEGER NOT NULL;

    BEGIN

        SELECT  :References = COUNT(*)
        FROM    Type
        WHERE   Type = :Type;

        IF References = 0 THEN
            INSERT INTO GrassCatcher (Object, Problem)
            VALUES (0, 'Type_004');
        ENDIF;

        SELECT  :References = COUNT(*)
        FROM    Type
        WHERE   Type = :SuperType;

        IF References = 0 THEN
            INSERT INTO GrassCatcher (Object, Problem)
            VALUES (0, 'Type_005');
        ENDIF;

/*
 *      Insert the type and all its supertypes as
 *      supertypes in the extended type list
 */
        INSERT INTO ExtendedType
            (Type
            , SuperType
            )
        VALUES
            (:Type
            , :Type
            );

        INSERT INTO ExtendedType
            (Type
            , SuperType
            )
        SELECT  :Type,
                SuperType
        FROM    ExtendedType
        WHERE   Type = :SuperType;

    END;

/*
 *      Ensure that there are no references to the sub-type on delete
 */
CREATE PROCEDURE SuperType$Delete
(
    Type      INTEGER
) =
    DECLARE
        References  INTEGER NOT NULL;
        Name        CHAR(20) NOT NULL;

    BEGIN

        SELECT  :Name = T.Name
        FROM    Type T

```

2143652

63

```

WHERE      :Type = T.Type;

SELECT      :References = COUNT(*)
FROM        SuperType ST
WHERE       :Type = ST.SuperType;

IF References != 0 THEN
    INSERT INTO GrassCatcher (Object, Text, Problem)
    VALUES (0, :Name, 'Type_001');
ENDIF;

SELECT      :References = COUNT(*)
FROM        Object O
WHERE       :Type = O.Type;

IF References != 0 THEN
    INSERT INTO GrassCatcher (Object, Text, Problem)
    VALUES (0, :Name, 'Type_002');
ENDIF;

DELETE FROM ExtendedType
WHERE      :Type = Type;

END;
/*
 *      Get the surrogate key set up on inserts
 */
CREATE PROCEDURE Type$Insert
(   Tid      INTEGER
,   Type     INTEGER
) =
BEGIN
    IF Type = -1 THEN
        UPDATE LastKey
        SET     Value = Value + 1
        WHERE  LastKey = 'Type';

        UPDATE Type C
        FROM    LastKey K
        SET     Type = K.Value
        WHERE   C.Tid = :Tid AND
                K.LastKey = 'Type';
    ENDIF;

END;
/*
 *      Prevent delete if the type is referenced in the
 *      super-type - sub-type heirarchy
 */
CREATE PROCEDURE Type$Delete
(   Type     INTEGER
,   Name     VARCHAR(20)
) =
DECLARE
    References INTEGER NOT NULL;
BEGIN
/*
 *      Verify that the Type is not a system type
 */
    IF Type < 10000 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
        VALUES (0, :Name, 'Type_003');
    ENDIF;
/*
 *      Verify that the Type is not a supertype

```

2143652

64

```

SELECT :References = COUNT(*)
FROM   SuperType
WHERE  SuperType = :Type;

IF References != 0 THEN
    INSERT INTO GrassCatcher (Object, Text, Problem)
    VALUES (0, :Name, 'Type_001');
ENDIF;
END;

/*
 *   Propagate any change to the value of a kVLevel
 */
CREATE PROCEDURE kVLevel$update_kVLevel
(   Old_kVLevel      REAL
,   New_kVLevel      REAL
) =
BEGIN
    UPDATE Terminal T
    SET    kVLevel = :New_kVLevel
    WHERE  :Old_kVLevel = T.kVLevel;
END;

/*
 *   Block delete of a kVLevel if there are any
 *   terminals at that voltage
 */
CREATE PROCEDURE kVLevel$Delete
(   kVLevel          REAL
) =
DECLARE
    References  INTEGER NOT NULL;
BEGIN
    SELECT :References = COUNT(*)
    FROM   Terminal T
    WHERE  :kVLevel = T.kVLevel;

    IF References != 0 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
        VALUES (0, VARCHAR(:kVLevel), 'Object_002');
    ENDIF;
END;

/*
 *   Propagate the transformer terminal voltage levels across all the
 *   connected terminals.
 */
CREATE PROCEDURE kVLevel$Propagate
=
DECLARE
    stext          varchar(40) not null;
    Loop           INTEGER NOT NULL;
    MaxIterations  INTEGER NOT NULL;
BEGIN
    DELETE FROM TEMP$TerminalList;
    DELETE FROM TEMP$ObjectList;
    DELETE FROM TEMP$AuditTrail;

/*
 *   Build a list of all the transformer terminals to use as a start point,
 *   and record the fact that they have been processed.
 */
    INSERT INTO TEMP$TerminalList
    SELECT *
    FROM   Terminal
    WHERE  BaseType = 27;          /* Type$Transformer */

    INSERT INTO TEMP$AuditTrail

```

2143652

65

```

SELECT  Object
FROM    TEMP$TerminalList;

/*
 *   Loop, looking for new objects, and set their terminal kV levels up
 *   when they are found
 */
Loop = 1;
MaxIterations = 40;

WHILE Loop <= MaxIterations DO
/*
 *   Get a list of all the objects connected to the terminals in the
 *   terminal list, propagating the voltage from the terminal list
 */
    DELETE FROM TEMP$ObjectList;
    INSERT INTO TEMP$ObjectList
        (Object
        ,kVLevel
        )
    SELECT DISTINCT
        T.Object,
        TL.kVLevel
    FROM    Terminal T,
           TEMP$TerminalList TL
    WHERE   TL.Node = T.Node AND
           TL.Object != T.Object;           /* same node */
                                           /* different object */

/*
 *   Discard any that have already been visited, set the voltage levels
 *   for all the terminals of those that remain, and add them to the
 *   audit trail
 */
    DELETE FROM TEMP$ObjectList OL
    WHERE   OL.Object = (SELECT Object FROM TEMP$AuditTrail);

    UPDATE Terminal T
    FROM    TEMP$ObjectList OL
    SET     kVLevel = OL.kVLevel
    WHERE   T.Object = OL.Object AND
           T.Node >= 10000;                 /* object in object list */
                                           /* not a system node */

    INSERT INTO TEMP$AuditTrail
    SELECT  DISTINCT Object
    FROM    TEMP$ObjectList;

/*
 *   Find all the other terminals for these objects
 */
    DELETE FROM TEMP$TerminalList;
    INSERT INTO TEMP$TerminalList
    SELECT DISTINCT T.*
    FROM    Terminal T,
           TEMP$ObjectList OL
    WHERE   OL.Object = T.Object AND
           T.Node >= 10000;                 /* same object */
                                           /* not a system node */

    IF IIRowCount = 0 THEN
        ENDLOOP;
    ENDIF;

    Loop = Loop + 1;
ENDWHILE;

/*
create table objectlist as
select  object
from    object
where   basetype != 27 and
        terminals > 1;

```

2143652

66

```

select  ol.object
from    objectlist ol,
        terminal t1,
        terminal t2
where   ol.object = TEMP$objectlist.object and
        ol.object = t2.object and
        t1.terminal != t2.terminal and
        t1.node >= 10000 and
        t2.node >= 10000 and
        t1.kvlevel != t2.kvlevel;
*/
END;
/*
*      Interactions
*/
/*
*      Check that not too many terminals are being used
*/
CREATE PROCEDURE Terminal$Validate
(
  Tid INTEGER
, Object INTEGER
) =
  DECLARE
    References      INTEGER;
    MaxTerm         INTEGER;
  BEGIN
    SELECT  :References = COUNT(*)
    FROM    Terminal
    WHERE   Object = :Object AND
            Terminal > 0;

    SELECT  :MaxTerm = Terminals
    FROM    Object
    WHERE   Object = :Object;

    IF References > MaxTerm THEN
      INSERT INTO GrassCatcher (Object, Problem)
      VALUES (:Object, 'Object_003');
    ENDIF;

  END;
/*
*      Check the referenced objects exist on inserts
*/
CREATE PROCEDURE Grouping$Insert
(
  Tid      INTEGER
, Grouping INTEGER
, Member   INTEGER
) =
  DECLARE
    GroupingType    INTEGER    WITH NULL;
    MemberType      INTEGER    WITH NULL;
  BEGIN
    GroupingType = NULL;

    SELECT  :GroupingType = Type
    FROM    Object
    WHERE   Object = :Grouping;

    IF GroupingType IS NULL THEN
      INSERT INTO GrassCatcher (Object, Problem)
      VALUES (0, 'Object_004');
    ENDIF;

    MemberType = NULL;

```

2143652

67

```

SELECT  :MemberType = Type
FROM    Object
WHERE   Object = :Member;

IF MemberType IS NULL THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (0, 'Object_005');
ENDIF;

UPDATE  Grouping
SET     GroupingType = :GroupingType,
        MemberType = :MemberType
WHERE   :Tid = Tid;
END;

CREATE PROCEDURE GrassCatcher$Insert
(Tid      INTEGER
, Object  INTEGER
, Problem CHAR(20)
, Text    CHAR(20)
) =
DECLARE
    References INTEGER NOT NULL;
    Description CHAR(60) NOT NULL;
    Severity    CHAR(1)  NOT NULL;
BEGIN
/*
*   Get rid of this entry if it is a duplicate
*/
    SELECT :References = COUNT(*)
    FROM   GrassCatcher
    WHERE  :Object = Object AND
           :Problem = Problem AND
           :Text = Text;

    IF References > 1 THEN
        DELETE FROM GrassCatcher
        WHERE   :Tid = Tid;
    ENDIF;

/*
*   Show the problem description, and blow the transaction away
*   if the error is fatal
*/
    SELECT :Description = P.Description,
           :Severity = P.Severity
    FROM   Problem P
    WHERE  :Problem = P.Problem;

    IF Severity = 'F' THEN
        RAISE ERROR -1 Description;
    ELSE
        MESSAGE 0 Description;
    ENDIF;
END;

```

2143652

68

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 * This file contain rules that fire in order to ensure referential
 * integrity. Typically, there is one for an insert on every table:
 * (TableName$Insert1) and one for a deletion on every table:
 * (TableName$Delete). There also may be a rule that fires on an
 * update of a specific column (TableName$update_ColumnName) if that
 * change requires a check on the REFERENTIAL validity of the
 * modification.
 *
 * The rules in this file are required to be active.
 */
/*
 * Object table
 */
CREATE RULE Object$Insert1
AFTER INSERT ON Object
EXECUTE PROCEDURE Object$Insert
(
  Tid = New.Tid
,
  Object = New.Object
,
  SystemVersion = New.SystemVersion
,
  Type = New.Type
,
  PrimaryGrouping = New.PrimaryGrouping
,
  Terminals = New.Terminals
);
CREATE RULE Object$Delete
AFTER DELETE ON Object
EXECUTE PROCEDURE Reject
(
  Problem = 'Object_006'
);
/*
 * Conducting plant
 */
/*
 * Transformer components
 */
CREATE RULE Winding$Insert1
AFTER INSERT ON Winding
EXECUTE PROCEDURE Winding$Insert
(
  Tid = New.Tid
,
  Object = New.Object
,
  KVLevel = New.NominalKV
);
CREATE RULE Winding$Delete
AFTER DELETE ON Winding
EXECUTE PROCEDURE Winding$Delete
(
  Object = Old.Object
,
  Winding = Old.Winding

```

2143652

69

```

    );
/*
 *      Specifications
 */
CREATE RULE LoadCurve$Insert1
AFTER INSERT ON LoadCurve
EXECUTE PROCEDURE LoadCurve$Insert
(
    Tid = New.Tid
    ,
    LoadCurve = New.LoadCurve
);
CREATE RULE LastKey$Delete
AFTER DELETE ON LastKey
EXECUTE PROCEDURE Reject
(
    Problem = 'Object_006'
);
CREATE RULE SuperType$Insert1
AFTER INSERT ON SuperType
EXECUTE PROCEDURE SuperType$Insert
(
    Type = New.Type
    ,
    SuperType = New.SuperType
);
CREATE RULE SuperType$Delete
AFTER DELETE ON SuperType
EXECUTE PROCEDURE SuperType$Delete
(
    Type = Old.Type
);
CREATE RULE Type$Insert1
AFTER INSERT ON Type
EXECUTE PROCEDURE Type$Insert
(
    Tid = New.Tid
    ,
    Type = New.Type
);
CREATE RULE Type$Delete
AFTER DELETE ON Type
EXECUTE PROCEDURE Type$Delete
(
    Type = Old.Type
    ,
    Name = Old.Name
);
CREATE RULE kvLevel$update_kvLevel
AFTER UPDATE (kvLevel) ON kvLevel
EXECUTE PROCEDURE kvLevel$update_kvLevel
(
    Old_kvLevel = Old.kvLevel
    ,
    New_kvLevel = New.kvLevel
);
CREATE RULE kvLevel$Delete
AFTER DELETE ON kvLevel
EXECUTE PROCEDURE kvLevel$Delete
(
    kvLevel = Old.kvLevel
);
/*
 *      Interactions
 */
CREATE RULE Grouping$Insert1
AFTER INSERT ON Grouping
EXECUTE PROCEDURE Grouping$Insert
(
    Tid = New.Tid
    ,
    Grouping = New.Grouping
    ,
    Member = New.Member
);

```

2143652

70

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 * This file contain rules that fire in order to ensure internal
 * validity of the data in a row. Typically, there is one for an
 * insert on every table: (TableName$Insert2). There also may be a
 * rule that fires on an update of a specific column:
 * (TableName$update_ColumnName) if that change requires a check
 * on the internal validity of the modification.
 *
 * The rules in this file are not required to be active. Although
 * they ensure that the data in a row is valid (often electrically);
 * they may be dropped to speed up bulk loading of data. Therefore
 * every rule that is defined in this file must have a corresponding
 * entry in the file "drop_rules_validation.sql", which can be
 * executed to drop the rules when desired. A awk program,
 * create_to_drop.awk exists for this purpose, and can be run
 * with the command:
 *
 *   "awk -f create_to_drop.awk\
 *     create_rules_validation.sql\
 *     > drop_rules_validation.sql"
 */
/*
 * Object table
 */
CREATE RULE Object$Insert2
AFTER INSERT ON Object
EXECUTE PROCEDURE Object$Validate
(
  Object = New.Object
  ,
  InService = New.InService
  ,
  OutService = New.OutService
);
CREATE RULE Object$update_Name
AFTER UPDATE (Name) ON Object
WHERE Old.Name != New.Name
EXECUTE PROCEDURE Object$Name
(
  Tid = New.Tid
  ,
  BaseType = New.BaseType
);
CREATE RULE Object$update_InService
AFTER UPDATE (InService) ON Object
WHERE Old.InService != New.InService
EXECUTE PROCEDURE Object$Validate
(
  Object = New.Object
  ,
  InService = New.InService
  ,
  OutService = New.OutService
);

```

2143652

71

```

CREATE RULE Object$update_OutService
  AFTER UPDATE (OutService) ON Object
  WHERE Old.OutService != New.OutService
  EXECUTE PROCEDURE Object$Validate
  (
    Object = New.Object
    ,
    InService = New.InService
    ,
    OutService = New.OutService
  );
CREATE RULE Object$update_Type
  AFTER UPDATE (Type) ON Object
  WHERE Old.Type != New.Type
  EXECUTE PROCEDURE Object$TypeName
  (
    Tid = New.Tid
  );
CREATE RULE Object$update_PG
  AFTER UPDATE (PrimaryGrouping) ON Object
  WHERE Old.PrimaryGrouping != New.PrimaryGrouping
  EXECUTE PROCEDURE Object$PrimaryGrouping
  (
    Tid = New.Tid
    ,
    Object = New.Object
    ,
    SystemVersion = New.SystemVersion
    ,
    Old_PrimaryGrouping = Old.PrimaryGrouping
    ,
    New_PrimaryGrouping = New.PrimaryGrouping
  );
/*
*      Conducting plant
*/
CREATE RULE Conductor$insert2
  AFTER INSERT ON Conductor
  EXECUTE PROCEDURE Conductor$Validate
  (
    Object = New.Object
    ,
    R = New.R
    ,
    X = New.X
    ,
    Bch = New.Bch
  );
CREATE RULE Conductor$update
  AFTER UPDATE ON Conductor
  EXECUTE PROCEDURE Conductor$Validate
  (
    Object = New.Object
    ,
    R = New.R
    ,
    X = New.X
    ,
    Bch = New.Bch
  );
CREATE RULE Consumer$insert2
  AFTER INSERT ON Consumer
  EXECUTE PROCEDURE Consumer$updateLoadModel
  (
    Object = New.Object
    ,
    Old_Pfixed = 0
    ,
    New_Pfixed = New.Pfixed
    ,
    Old_Pnom = 0
    ,
    New_Pnom = New.Pnom
    ,
    Old_Qfixed = 0
    ,
    New_Qfixed = New.Qfixed
    ,
    Old_Qnom = 0
    ,
    New_Qnom = New.Qnom
  );
CREATE RULE Consumer$updatePQ
  AFTER UPDATE ON Consumer
  WHERE Old.Pfixed != New.Pfixed OR
        Old.Pnom != New.Pnom OR
        Old.Qfixed != New.Qfixed OR
        Old.Qnom != New.Qnom
  EXECUTE PROCEDURE Consumer$updateLoadModel
  (
    Object = New.Object
    ,
    Old_Pfixed = Old.Pfixed
    ,
    New_Pfixed = New.Pfixed
  );

```

2143652

72

```

        Old_Pnom = Old.Pnom
        New_Pnom = New.Pnom
        Old_Qfixed = Old.Qfixed
        New_Qfixed = New.Qfixed
        Old_Qnom = Old.Qnom
        New_Qnom = New.Qnom
    );
CREATE RULE Consumer$UpdatePQPct
AFTER UPDATE (PfixedPct) ON Consumer
WHERE Old.Type = 91 /* Type$LoadArea */
EXECUTE PROCEDURE Consumer$UpdateLoadPct
(
    Object = New.Object
    , Pfixed = New.Pfixed
    , Pnom = New.Pnom
    , Qfixed = New.Qfixed
    , Qnom = New.Qnom
    , PfixedPct = New.PfixedPct
    , PnomPct = New.PnomPct
    , QfixedPct = New.QfixedPct
    , QnomPct = New.QnomPct
    , TopLoadArea = New.TopLoadArea
);
/*
 * Transformer components
 */
CREATE RULE Winding$Insert2
AFTER INSERT ON Winding
EXECUTE PROCEDURE Winding$Validate
(
    Object = New.Object
    , Name = New.Name
    , LoadTapChanger = New.LoadTapChanger
    , NominalKV = New.NominalKV
    , NeutralKV = New.NeutralKV
    , LowStep = New.LowStep
    , HighStep = New.HighStep
    , NeutralStep = New.NeutralStep
    , StepSize = New.StepSize
);
CREATE RULE Winding$Update
AFTER UPDATE ON Winding
EXECUTE PROCEDURE Winding$Validate
(
    Object = New.Object
    , Name = New.Name
    , LoadTapChanger = New.LoadTapChanger
    , NominalKV = New.NominalKV
    , NeutralKV = New.NeutralKV
    , LowStep = New.LowStep
    , HighStep = New.HighStep
    , NeutralStep = New.NeutralStep
    , StepSize = New.StepSize
);
CREATE RULE TapSetting$Insert2
AFTER INSERT ON TapSetting
EXECUTE PROCEDURE TapSetting$Validate
(
    Object = New.Object
    , Winding1 = New.Winding1
    , TapStep1 = New.TapStep1
    , Winding2 = New.Winding2
    , TapStep2 = New.TapStep2
    , Winding3 = New.Winding3
    , TapStep3 = New.TapStep3
);
CREATE RULE TapSetting$Update
AFTER UPDATE ON TapSetting
EXECUTE PROCEDURE TapSetting$Validate
(
    Object = New.Object

```

2143652

73

```
,      Winding1 = New.Winding1
,      TapStep1 = New.TapStep1
,      Winding2 = New.Winding2
,      TapStep2 = New.TapStep2
,      Winding3 = New.Winding3
,      TapStep3 = New.TapStep3
);
/*
 *      Specifications
 */
/*
 *      Interactions
 */
CREATE RULE Terminal$Insert2
AFTER INSERT ON Terminal
EXECUTE PROCEDURE Terminal$Validate
(      Tid = New.Tid
,      Object = New.Object
);

CREATE RULE GrassCatcher$Insert2
AFTER INSERT ON GrassCatcher
EXECUTE PROCEDURE GrassCatcher$Insert
(      Tid = New.Tid
,      Object = New.Object
,      Problem = New.Problem
,      Text = New.Text
);
```

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 * A base type is a position in the type hierarchy that identifies a number of
 * subtypes that have similar attributes and behavior. Note that the type is a
 * forward reference to an entry in the type table that may not be defined
 * yet. This is preferable to allowing types to be defined with no checking on
 * the validity of the base type. For each base type the name of the editor
 * frame and the name of any detail table that is required may be defined.
 */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (01, '', ''); /* Type$Object */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (02, 'ObjectGeneric', ''); /* Type$Organization */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (18, 'ObjectGeneric', ''); /* Type$Ground */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (19, 'Conductor', 'Conductor'); /* Type$Busbar */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (20, 'Generator', 'Generator'); /* Type$Generator */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (21, 'Consumer', 'Consumer'); /* Type$Consumer */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (22, 'Conductor', 'Conductor'); /* Type$ACLineSegment */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (23, 'Conductor', 'Conductor'); /* Type$Jumper */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (24, 'Capacitor', 'Capacitor'); /* Type$Capacitor */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (25, 'Reactor', 'Reactor'); /* Type$Reactor */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (26, 'Switch', 'Switch'); /* Type$Switch */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (27, 'Transformer', 'Transformer'); /* Type$Transformer */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (31, 'BreakerSensor', ''); /* Type$Relay */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (35, 'Telemetry', 'Telemetry'); /* Type$Telemetry */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (43, 'TowerType', 'TowerType'); /* Type$TowerType */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (44, 'ConductorType', 'ConductorType'); /* Type$ConductorType */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (45, 'ObjectGeneric', ''); /* Type$RatingFamily */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (46, 'ObjectGeneric', ''); /* Type$LoadFamily */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (47, 'ObjectGeneric', ''); /* Type$Drawing */
INSERT INTO BaseType (Type, EditorName, TableName)

```

75

```

VALUES (48, 'ObjectGeneric', ''); /* Type$Manufacturer */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (49, 'ObjectGeneric', ''); /* Type$Season */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (50, 'ObjectGeneric', ''); /* Type$DayType */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (59, 'Conductor', 'Conductor'); /* Type$DCLineSegment */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (60, 'Switch', 'Switch'); /* Type$Breaker */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (61, 'Switch', 'Switch'); /* Type$Disconnect */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (62, 'Switch', 'Switch'); /* Type$Fuse */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (63, 'Computer', 'Computer'); /* Type$Computer */
INSERT INTO BaseType (Type, EditorName, TableName)
VALUES (90, 'Person', 'Person'); /* Type$Person */

```

```

/*
 * The following statements define the system types. Any entry added to the
 * type table must be duplicated with an entry of a constant in the
 * PowerSystemModel application with the name "Type$xxxx" and the integer
 * value set to the value defined here.
 *
 * The type table contains the type, the base type and the type name for
 * each type. The type must be unique; the name should be unique but this
 * is not enforced. A base type of 0 implies that the type is for
 * classification only: an object of that type cannot be created or edited.
 * If the base type is non-zero the type of the object can be freely changed
 * amongst others which are subtypes of the base type. Note that it is
 * possible to create a heirarchy of types that can only be descended.
 *
 * New entries must be added at the end of the list, with the next highest
 * type number. For each new entry in the type table, its immediate supertype
 * must be defined in the supertype table. These definitions are grouped
 * after the type table definitions.
 */

```

```

INSERT INTO Type (Type, BaseType, Name) VALUES (00, 01, '');
INSERT INTO Type (Type, BaseType, Name) VALUES (01, 01, 'Object');
INSERT INTO Type (Type, BaseType, Name) VALUES (02, 02, 'Organization');
INSERT INTO Type (Type, BaseType, Name) VALUES (03, 02, 'Company');
INSERT INTO Type (Type, BaseType, Name) VALUES (04, 02, 'Division');
INSERT INTO Type (Type, BaseType, Name) VALUES (05, 02, 'Substation');
INSERT INTO Type (Type, BaseType, Name) VALUES (06, 02, 'Bay');
INSERT INTO Type (Type, BaseType, Name) VALUES (07, 02, 'System');
INSERT INTO Type (Type, BaseType, Name) VALUES (08, 02, 'Spares');
INSERT INTO Type (Type, BaseType, Name) VALUES (09, 63, 'RTU');
INSERT INTO Type (Type, BaseType, Name) VALUES (10, 02, 'Line');
INSERT INTO Type (Type, BaseType, Name) VALUES (11, 02, 'Circuit');
INSERT INTO Type (Type, BaseType, Name) VALUES (12, 02, 'TransformerBank');
INSERT INTO Type (Type, BaseType, Name) VALUES (13, 02, 'CapacitorBank');
INSERT INTO Type (Type, BaseType, Name) VALUES (14, 01, 'Equipment');
INSERT INTO Type (Type, BaseType, Name) VALUES (15, 00, 'Template');
INSERT INTO Type (Type, BaseType, Name) VALUES (16, 01, 'Support');
INSERT INTO Type (Type, BaseType, Name) VALUES (17, 01, 'Conducting');
INSERT INTO Type (Type, BaseType, Name) VALUES (18, 18, 'Ground');
INSERT INTO Type (Type, BaseType, Name) VALUES (19, 19, 'Busbar');
INSERT INTO Type (Type, BaseType, Name) VALUES (20, 20, 'Generator');
INSERT INTO Type (Type, BaseType, Name) VALUES (21, 21, 'Consumer');
INSERT INTO Type (Type, BaseType, Name) VALUES (22, 22, 'ACLineSegment');
INSERT INTO Type (Type, BaseType, Name) VALUES (23, 23, 'Jumper');
INSERT INTO Type (Type, BaseType, Name) VALUES (24, 24, 'Capacitor');
INSERT INTO Type (Type, BaseType, Name) VALUES (25, 25, 'Reactor');
INSERT INTO Type (Type, BaseType, Name) VALUES (26, 26, 'Switch');
INSERT INTO Type (Type, BaseType, Name) VALUES (27, 27, 'Transformer');
INSERT INTO Type (Type, BaseType, Name) VALUES (28, 27, 'PowerTransformer');
INSERT INTO Type (Type, BaseType, Name) VALUES (29, 27, 'PhaseShifter');

```

2143652

76

```

INSERT INTO Type (Type, BaseType, Name) VALUES (30, 01, 'Sensing');
INSERT INTO Type (Type, BaseType, Name) VALUES (31, 01, 'Relay');
INSERT INTO Type (Type, BaseType, Name) VALUES (32, 31, 'VoltageRelay');
INSERT INTO Type (Type, BaseType, Name) VALUES (33, 31, 'FrequencyRelay');
INSERT INTO Type (Type, BaseType, Name) VALUES (34, 31, 'CurrentRelay');
INSERT INTO Type (Type, BaseType, Name) VALUES (35, 35, 'Telemetry');
INSERT INTO Type (Type, BaseType, Name) VALUES (36, 35, 'Analog');
INSERT INTO Type (Type, BaseType, Name) VALUES (37, 35, 'Digital');
INSERT INTO Type (Type, BaseType, Name) VALUES (38, 35, 'Counter');
INSERT INTO Type (Type, BaseType, Name) VALUES (39, 35, 'Control');
INSERT INTO Type (Type, BaseType, Name) VALUES (40, 21, 'Feeder');
INSERT INTO Type (Type, BaseType, Name) VALUES (41, 21, 'Load');
INSERT INTO Type (Type, BaseType, Name) VALUES (42, 01, 'Specification');
INSERT INTO Type (Type, BaseType, Name) VALUES (43, 43, 'TowerType');
INSERT INTO Type (Type, BaseType, Name) VALUES (44, 44, 'ConductorType');
INSERT INTO Type (Type, BaseType, Name) VALUES (45, 45, 'RatingFamily');
INSERT INTO Type (Type, BaseType, Name) VALUES (46, 46, 'LoadFamily');
INSERT INTO Type (Type, BaseType, Name) VALUES (47, 47, 'Drawing');
INSERT INTO Type (Type, BaseType, Name) VALUES (48, 48, 'Manufacturer');
INSERT INTO Type (Type, BaseType, Name) VALUES (49, 49, 'Season');
INSERT INTO Type (Type, BaseType, Name) VALUES (50, 50, 'DayType');
INSERT INTO Type (Type, BaseType, Name) VALUES (51, 00, 'Interaction');
INSERT INTO Type (Type, BaseType, Name) VALUES (52, 00, 'Member of');
INSERT INTO Type (Type, BaseType, Name) VALUES (53, 00, 'Rated as');
INSERT INTO Type (Type, BaseType, Name) VALUES (54, 00, 'Load modeled by');
INSERT INTO Type (Type, BaseType, Name) VALUES (55, 00, 'Shown on');
INSERT INTO Type (Type, BaseType, Name) VALUES (56, 00, 'Made by');
INSERT INTO Type (Type, BaseType, Name) VALUES (57, 00, 'Located with');
INSERT INTO Type (Type, BaseType, Name) VALUES (58, 00, 'Conductor');
INSERT INTO Type (Type, BaseType, Name) VALUES (59, 59, 'DCLineSegment');
INSERT INTO Type (Type, BaseType, Name) VALUES (60, 60, 'Breaker');
INSERT INTO Type (Type, BaseType, Name) VALUES (61, 61, 'Disconnect');
INSERT INTO Type (Type, BaseType, Name) VALUES (62, 62, 'Fuse');
INSERT INTO Type (Type, BaseType, Name) VALUES (63, 63, 'Computer');
INSERT INTO Type (Type, BaseType, Name) VALUES (64, 00, 'Measurement');
INSERT INTO Type (Type, BaseType, Name) VALUES (65, 00, 'Sensor');
INSERT INTO Type (Type, BaseType, Name) VALUES (66, 00, 'Current');
INSERT INTO Type (Type, BaseType, Name) VALUES (67, 00, 'Voltage');
INSERT INTO Type (Type, BaseType, Name) VALUES (68, 00, 'Frequency');
INSERT INTO Type (Type, BaseType, Name) VALUES (69, 00, 'Position');
INSERT INTO Type (Type, BaseType, Name) VALUES (70, 00, 'Impulse');
INSERT INTO Type (Type, BaseType, Name) VALUES (71, 00, 'Contact');
INSERT INTO Type (Type, BaseType, Name) VALUES (72, 00, 'Power');
INSERT INTO Type (Type, BaseType, Name) VALUES (73, 00, 'Measurer');
INSERT INTO Type (Type, BaseType, Name) VALUES (74, 00, 'Telemetered');
INSERT INTO Type (Type, BaseType, Name) VALUES (75, 00, 'Manual');
INSERT INTO Type (Type, BaseType, Name) VALUES (76, 00, 'Calculated');
INSERT INTO Type (Type, BaseType, Name) VALUES (77, 00, 'StateEstimated');
INSERT INTO Type (Type, BaseType, Name) VALUES (78, 00, 'RemoteSystem');
INSERT INTO Type (Type, BaseType, Name) VALUES (79, 00, 'External');
INSERT INTO Type (Type, BaseType, Name) VALUES (80, 00, 'Measurand');
INSERT INTO Type (Type, BaseType, Name) VALUES (81, 00, 'Amps');
INSERT INTO Type (Type, BaseType, Name) VALUES (82, 00, 'kV');
INSERT INTO Type (Type, BaseType, Name) VALUES (83, 00, 'Hertz');
INSERT INTO Type (Type, BaseType, Name) VALUES (84, 00, 'Setting');
INSERT INTO Type (Type, BaseType, Name) VALUES (85, 00, 'Counts');
INSERT INTO Type (Type, BaseType, Name) VALUES (86, 00, 'MW');
INSERT INTO Type (Type, BaseType, Name) VALUES (87, 00, 'MVar');
INSERT INTO Type (Type, BaseType, Name) VALUES (88, 00, 'MVA');
INSERT INTO Type (Type, BaseType, Name) VALUES (89, 00, 'Status');
INSERT INTO Type (Type, BaseType, Name) VALUES (90, 90, 'Person');
INSERT INTO Type (Type, BaseType, Name) VALUES (91, 21, 'LoadArea');
INSERT INTO Type (Type, BaseType, Name) VALUES (92, 21, 'Loss');

```

```

/*
 *      Set up the type heirarchy. The entries in this section must be
 *      ordered by supertype within supertype to ensure that the

```

77

* ExtendedType tables gets set up correctly. There should be one
 * entry for each Type defined above, and that Type's immediate
 * SuperType must be defined.
 */

INSERT INTO SuperType (SuperType, Type) VALUES (00, 01);

INSERT INTO SuperType (SuperType, Type) VALUES (01, 02);

INSERT INTO SuperType (SuperType, Type) VALUES (01, 14);

INSERT INTO SuperType (SuperType, Type) VALUES (01, 42);

INSERT INTO SuperType (SuperType, Type) VALUES (01, 51);

INSERT INTO SuperType (SuperType, Type) VALUES (01, 64);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 03);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 04);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 05);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 06);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 07);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 08);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 10);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 11);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 12);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 13);

INSERT INTO SuperType (SuperType, Type) VALUES (02, 15);

INSERT INTO SuperType (SuperType, Type) VALUES (14, 16);

INSERT INTO SuperType (SuperType, Type) VALUES (14, 17);

INSERT INTO SuperType (SuperType, Type) VALUES (14, 30);

INSERT INTO SuperType (SuperType, Type) VALUES (14, 63);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 18);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 20);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 21);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 24);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 25);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 26);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 27);

INSERT INTO SuperType (SuperType, Type) VALUES (17, 58);

INSERT INTO SuperType (SuperType, Type) VALUES (21, 40);

INSERT INTO SuperType (SuperType, Type) VALUES (21, 41);

INSERT INTO SuperType (SuperType, Type) VALUES (21, 91);

INSERT INTO SuperType (SuperType, Type) VALUES (21, 92);

INSERT INTO SuperType (SuperType, Type) VALUES (26, 60);

INSERT INTO SuperType (SuperType, Type) VALUES (26, 61);

INSERT INTO SuperType (SuperType, Type) VALUES (26, 62);

INSERT INTO SuperType (SuperType, Type) VALUES (27, 28);

INSERT INTO SuperType (SuperType, Type) VALUES (27, 29);

INSERT INTO SuperType (SuperType, Type) VALUES (58, 19);

INSERT INTO SuperType (SuperType, Type) VALUES (58, 22);

INSERT INTO SuperType (SuperType, Type) VALUES (58, 23);

INSERT INTO SuperType (SuperType, Type) VALUES (58, 59);

INSERT INTO SuperType (SuperType, Type) VALUES (30, 31);

INSERT INTO SuperType (SuperType, Type) VALUES (30, 35);

INSERT INTO SuperType (SuperType, Type) VALUES (31, 32);

INSERT INTO SuperType (SuperType, Type) VALUES (31, 33);

INSERT INTO SuperType (SuperType, Type) VALUES (31, 34);

INSERT INTO SuperType (SuperType, Type) VALUES (35, 36);

INSERT INTO SuperType (SuperType, Type) VALUES (35, 37);

INSERT INTO SuperType (SuperType, Type) VALUES (35, 38);

INSERT INTO SuperType (SuperType, Type) VALUES (35, 39);

INSERT INTO SuperType (SuperType, Type) VALUES (63, 09);

INSERT INTO SuperType (SuperType, Type) VALUES (42, 43);

2143652

78

```
INSERT INTO SuperType (SuperType, Type) VALUES (42, 44);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 45);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 46);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 47);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 48);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 49);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 50);
INSERT INTO SuperType (SuperType, Type) VALUES (42, 90);

INSERT INTO SuperType (SuperType, Type) VALUES (51, 52);
INSERT INTO SuperType (SuperType, Type) VALUES (51, 53);
INSERT INTO SuperType (SuperType, Type) VALUES (51, 54);
INSERT INTO SuperType (SuperType, Type) VALUES (51, 55);
INSERT INTO SuperType (SuperType, Type) VALUES (51, 56);
INSERT INTO SuperType (SuperType, Type) VALUES (51, 57);

INSERT INTO SuperType (SuperType, Type) VALUES (64, 65);
INSERT INTO SuperType (SuperType, Type) VALUES (64, 73);
INSERT INTO SuperType (SuperType, Type) VALUES (64, 80);

INSERT INTO SuperType (SuperType, Type) VALUES (65, 66);
INSERT INTO SuperType (SuperType, Type) VALUES (65, 67);
INSERT INTO SuperType (SuperType, Type) VALUES (65, 68);
INSERT INTO SuperType (SuperType, Type) VALUES (65, 69);
INSERT INTO SuperType (SuperType, Type) VALUES (65, 70);
INSERT INTO SuperType (SuperType, Type) VALUES (65, 71);
INSERT INTO SuperType (SuperType, Type) VALUES (65, 72);

INSERT INTO SuperType (SuperType, Type) VALUES (73, 74);
INSERT INTO SuperType (SuperType, Type) VALUES (73, 75);
INSERT INTO SuperType (SuperType, Type) VALUES (73, 76);
INSERT INTO SuperType (SuperType, Type) VALUES (73, 77);
INSERT INTO SuperType (SuperType, Type) VALUES (73, 78);
INSERT INTO SuperType (SuperType, Type) VALUES (73, 79);

INSERT INTO SuperType (SuperType, Type) VALUES (80, 81);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 82);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 83);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 84);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 85);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 86);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 87);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 88);
INSERT INTO SuperType (SuperType, Type) VALUES (80, 89);
```

2143652

79

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 *      This file contains statements to set up all the system defined
 *      nodes and objects.
 */
/*
 *      First, set up the keys. System defined keys range in value from
 *      1 - 10,000; User defined keys range in value from 10,001 to 20,000.
 *      Automatically generated keys (for things added in the general use
 *      of the system) start at 20,001.
 */
INSERT INTO LastKey
  (LastKey
   ,Value
  )
VALUES
  ('Object'
   ,20000
  );
INSERT INTO LastKey
  (LastKey
   ,Value
  )
VALUES
  ('Node'
   ,20000
  );
INSERT INTO LastKey
  (LastKey
   ,Value
  )
VALUES
  ('Type'
   ,20000
  );
INSERT INTO LastKey
  (LastKey
   ,Value
  )
VALUES
  ('Curves'
   ,20000
  );
/*
 *
 *      Objects that are created in this section must have a primary key
 *      that is equal to the base type of the object.
 */

```

2143652

80

```

/*
 *      Set up a "Null" object
 */
INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
   ,Description
  )
VALUES
  (0
   ,0
   ,0
   ,0
   ,''
   , 'Null object'
  );

INSERT INTO Location
  (Object
  )
VALUES
  (0
   , /* Type$Null */
  );

/*
 *      Create the top groupings that everything will belong to
 */
INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
   ,Description
  )
VALUES
  (2
   ,0
   ,2
   ,0
   , 'Organization'
   , 'Top level organization'
  );

INSERT INTO Grouping
  (SystemVersion
   ,Grouping
   ,Member
   ,Relationship
  )
VALUES
  (0
   ,0
   ,2
   ,52
   , /* Type$Null */
   , /* Type$Null */
   , /* Type$Organization */
   , /* Type$Member_of */
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
   ,Description
  )

```

2143652

81

```

)
VALUES
  (42          /* Type$Specification */
  ,0           /* Type$Null */
  ,42          /* Type$Specification */
  ,0
  , 'Specification'
  , 'Top level specification'
  );

INSERT INTO Grouping
  (SystemVersion
  ,Grouping
  ,Member
  ,Relationship
  )
VALUES
  (0           /* Type$Null */
  ,0           /* Type$Null */
  ,42          /* Type$Specification */
  ,52          /* Type$Member_of */
  );

/*
 *      Set up a single terminal ground for the entire system.
 */
INSERT INTO Object
  (Object
  ,SystemVersion
  ,Type
  ,Terminals
  ,Name
  ,Description
  ,InService
  )
VALUES
  (18          /* Type$Ground */
  ,0           /* Type$Null */
  ,18          /* Type$Ground */
  ,1
  , 'Ground'
  , 'Single terminal ground for entire system'
  , '1-Jan-1900'
  );

UPDATE Terminal T
SET   Node = 18          /* Type$Ground */
WHERE T.Object = 18;    /* Type$Ground */

INSERT INTO Grouping
  (SystemVersion
  ,Grouping
  ,Member
  ,Relationship
  )
VALUES
  (0           /* Type$Null */
  ,42          /* Type$Specification */
  ,18          /* Type$Ground */
  ,52          /* Type$Member_of */
  );

/*
 *      Create the template object for each base type with a key of
 *      BaseType + 5000.
 */
INSERT INTO Object
  (Object

```

2143652

82

```

      ,SystemVersion
      ,Type
      ,Terminals
      ,Name
    )
VALUES
    (02 + 5000 /* Type$Organization */
    ,0 /* Type$Null */
    ,02 /* Type$Organization */
    ,0
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (19 + 5000 /* Type$Busbar */
    ,0 /* Type$Null */
    ,19 /* Type$Busbar */
    ,1
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (20 + 5000 /* Type$Generator */
    ,0 /* Type$Null */
    ,20 /* Type$Generator */
    ,1
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (21 + 5000 /* Type$Consumer */
    ,0 /* Type$Null */
    ,21 /* Type$Consumer */
    ,1
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES

```

2143652

83

```

      (22 + 5000 /* Type$ACLineSegment */
      ,0        /* Type$Null */
      ,22      /* Type$ACLineSegment */
      ,2
      , 'Template'
      );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (23 + 5000 /* Type$Jumper */
  ,0        /* Type$Null */
  ,23      /* Type$Jumper */
  ,2
  , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (24 + 5000 /* Type$Capacitor */
  ,0        /* Type$Null */
  ,24      /* Type$Capacitor */
  ,2
  , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (25 + 5000 /* Type$Reactor */
  ,0        /* Type$Null */
  ,25      /* Type$Reactor */
  ,2
  , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (26 + 5000 /* Type$Switch */
  ,0        /* Type$Null */
  ,26      /* Type$Switch */
  ,2
  , 'Template'
  );

```

2143652

84

```

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (27 + 5000 /* Type$Transformer */
   ,0 /* Type$Null */
   ,27 /* Type$Transformer */
   ,0
   , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (31 + 5000 /* Type$BreakerSensor */
   ,0 /* Type$Null */
   ,31 /* Type$BreakerSensor */
   ,2
   , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (35 + 5000 /* Type$Telemetry */
   ,0 /* Type$Null */
   ,35 /* Type$Telemetry */
   ,0
   , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals
   ,Name
  )
VALUES
  (43 + 5000 /* Type$TowerType */
   ,0 /* Type$Null */
   ,43 /* Type$TowerType */
   ,0
   , 'Template'
  );

INSERT INTO Object
  (Object
   ,SystemVersion
   ,Type
   ,Terminals

```

2143652

85

```

        ,Name
    )
VALUES
    (44 + 5000 /* Type$ConductorType */
    ,0 /* Type$Null */
    ,44 /* Type$ConductorType */
    ,0
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (45 + 5000 /* Type$RatingFamily */
    ,0 /* Type$Null */
    ,45 /* Type$RatingFamily */
    ,0
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (46 + 5000 /* Type$LoadFamily */
    ,0 /* Type$Null */
    ,46 /* Type$LoadFamily */
    ,0
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (47 + 5000 /* Type$Drawing */
    ,0 /* Type$Null */
    ,47 /* Type$Drawing */
    ,0
    , 'Template'
    );

INSERT INTO Object
    (Object
    ,SystemVersion
    ,Type
    ,Terminals
    ,Name
    )
VALUES
    (48 + 5000 /* Type$Manufacturer */
    ,0 /* Type$Null */
    ,48 /* Type$Manufacturer */

```

2143652

86

```

    ,0
    , 'Template'
    );

INSERT INTO Object
    (Object
    , SystemVersion
    , Type
    , Terminals
    , Name
    )
VALUES
    (49 + 5000 /* Type$Season */
    , 0 /* Type$Null */
    , 49 /* Type$Season */
    , 0
    , 'Template'
    );

INSERT INTO Object
    (Object
    , SystemVersion
    , Type
    , Terminals
    , Name
    )
VALUES
    (50 + 5000 /* Type$DayType */
    , 0 /* Type$Null */
    , 50 /* Type$DayType */
    , 0
    , 'Template'
    );

INSERT INTO Object
    (Object
    , SystemVersion
    , Type
    , Terminals
    , Name
    )
VALUES
    (59 + 5000 /* Type$DCLineSegment */
    , 0 /* Type$Null */
    , 59 /* Type$DCLineSegment */
    , 2
    , 'Template'
    );

INSERT INTO Object
    (Object
    , SystemVersion
    , Type
    , Terminals
    , Name
    )
VALUES
    (60 + 5000 /* Type$Breaker */
    , 0 /* Type$Null */
    , 60 /* Type$Breaker */
    , 2
    , 'Template'
    );

INSERT INTO Object
    (Object

```

2143652

87

```
,SystemVersion
,Type
,Terminals
,Name
)
VALUES
(61 + 5000 /* Type$Disconnect */
,0 /* Type$Null */
,61 /* Type$Disconnect */
,2
,'Template'
);

INSERT INTO Object
(Object
,SystemVersion
,Type
,Terminals
,Name
)
VALUES
(62 + 5000 /* Type$Fuse */
,0 /* Type$Null */
,62 /* Type$Fuse */
,2
,'Template'
);

INSERT INTO Object
(Object
,SystemVersion
,Type
,Terminals
,Name
)
VALUES
(63 + 5000 /* Type$Computer */
,0 /* Type$Null */
,63 /* Type$Computer */
,0
,'Template'
);

INSERT INTO Object
(Object
,SystemVersion
,Type
,Terminals
,Name
)
VALUES
(90 + 5000 /* Type$Person */
,0 /* Type$Null */
,90 /* Type$Person */
,0
,'Template'
);
```

2143652

88

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

```

*****
/*
*      Parameters relating to the system being modelled are defined here.
*/
INSERT INTO SystemParameters
  (Frequency
  ,MVABase
  ,KVReference
  ,LengthRatio
  ,GroundResistivity
  ,Temperature
  )
VALUES
  (60
  ,100
  ,1
  ,1000
  ,100
  , 'C'
  );

```

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 * The following statements define the problem descriptions that are
 * recognised in the Power System Data Model
 */
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_001', 'E', 'In Service must precede Out of Service');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_002', 'F', 'There is equipment at this voltage');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_003', 'F', 'This device has no more terminals');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_004', 'F', 'Group object does not exist');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_005', 'F', 'Member object does not exist');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_006', 'F', 'Delete not permitted');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_007', 'F', 'Base type of an object may not be changed');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_008', 'F', 'Please select a terminal to connect to');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_009', 'F', 'Please select a terminal to be connected');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_010', 'F', 'Please select a terminal to remove');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_011', 'F', 'There is already a relationship with this object');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_012', 'F', 'Objects of this type may not be created');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_013', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_014', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_015', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_016', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_017', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_018', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_019', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Object_020', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)

```

2143652

90

```

VALUES ('Capacitor_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Capacitor_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_001', 'W', 'X/R ratio is low');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Conductor_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Consumer_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)

```

2143652

91

```

VALUES ('Generator_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Generator_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Reactor_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Switch_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_001', 'W', 'Nominal kV is out of range of the taps' );
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_002', 'W', 'Step Size is greater than 10%');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_003', 'E', 'Load tap changer must be "Yes" or "No"');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_004', 'E', 'Low Step must be <= High Step');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_005', 'E', 'Neutral Step must be >= low step');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_006', 'E', 'Neutral step must be <= high step');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_007', 'E', 'Step size must be non-zero');
INSERT INTO Problem (Problem, Severity, Description)

```

2143652

92

```

VALUES ('Transformer_008', 'F', 'Windings must belong to transformers' ;
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_009', 'E', 'Tap Setting winding does not exist');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_010', 'E', 'Tap Setting tap step is out of range');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_011', 'E', 'Tap Setting windings 1 and 2 must be specif
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_012', 'E', 'Tap Setting windings may be used once only'
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_013', 'E', 'Rated MVA is not specified for any winding'
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_014', 'E', 'Tap Setting impedance is not in the expecte
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_015', 'F', 'Winding is referenced by a Tap Setting');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_016', 'F', 'There are Tap Settings defined');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_017', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_018', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_019', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Transformer_020', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Telemetry_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('ConductorType_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)

```

2143652

93

```

VALUES ('TowerType_001', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_002', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_003', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_004', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_005', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_006', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_007', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('TowerType_010', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_001', 'F', 'Type is in use as a super-type');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_002', 'F', 'There are objects of this type');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_003', 'F', 'Type is system defined and cannot be deleted');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_004', 'F', 'Sub-type does not exist');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_005', 'F', 'Super-type does not exist');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_006', 'F', 'No editor defined for this type of object');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_007', 'F', 'A type of that name already exists');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_008', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_009', '', '');
INSERT INTO Problem (Problem, Severity, Description)
VALUES ('Type_010', '', '');

```

2143652

94

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

*****
DROP PROCEDURE kvLevel$Propagate;
CREATE PROCEDURE kvLevel$Propagate
=
    DECLARE
        i                INTEGER NOT NULL;
        Loop             INTEGER NOT NULL;
        MaxIterations    INTEGER NOT NULL;
    BEGIN
        DELETE FROM TEMP$TerminalList;
        DELETE FROM TEMP$ObjectList;
        DELETE FROM TEMP$AuditTrail;
/*
*   Build a list of all the transformer terminals to use as a start point,
*   and record the fact that they have been processed.
*/
        INSERT INTO TEMP$TerminalList
        SELECT *
        FROM   Terminal
        WHERE  BaseType = 27;           /* Type$Transformer */

        INSERT INTO TEMP$AuditTrail
        SELECT Object
        FROM   TEMP$TerminalList;
/*
*   Loop, looking for new objects, and set their terminal kV levels up
*   when they are found
*/
        Loop = 1;
        MaxIterations = 40;

        WHILE Loop <= MaxIterations DO
            message loop 'Loops';
            SELECT :i = count(*) from temp$terminalList;
            message i 'Terminals';

/*
*   Get a list of all the objects connected to the terminals in the
*   terminal list, propagating the voltage from the terminal list. Force
*   the voltage propagated to be that of the highest voltage terminal
*   (in case some are grounded). Note that transformers have already
*   been dealt with.
*/
            DELETE FROM TEMP$ObjectList;
            INSERT INTO TEMP$ObjectList
                (Object
                 ,kVLevel
                )

```

2143652

95

```

SELECT
    T.Object,
    MAX(TL.kVLevel)
FROM    Terminal T,
        TEMP$TerminalList TL
WHERE    TL.Node = T.Node AND                /* same node */
        TL.Object != T.Object                /* different object */
GROUP BY T.Object;
message irowcount 'Objects';

/*
 * Discard any that have already been visited, set the voltage levels
 * for all the terminals of those that remain, and add them to the
 * audit trail
 */

DELETE FROM TEMP$ObjectList OL
WHERE    OL.Object = (SELECT Object FROM TEMP$AuditTrail);
message irowcount 'Revisited';

UPDATE    Terminal T
FROM      TEMP$ObjectList OL
SET        kVLevel = OL.kVLevel
WHERE      T.Object = OL.Object AND          /* object in object list */
        T.Node >= 10000;                    /* not a system node */
message irowcount 'Updated';

INSERT INTO TEMP$AuditTrail
SELECT    Object
FROM      TEMP$ObjectList;
message irowcount 'Audited';

/*
 * Find all the other terminals for these objects
 */

DELETE FROM TEMP$TerminalList;
INSERT INTO TEMP$TerminalList
SELECT    T.*
FROM      Terminal T,
        TEMP$ObjectList OL
WHERE      OL.Object = T.Object AND          /* same object */
        T.Node >= 10000;                    /* not a system node */

IF IIRowCount = 0 THEN
    ENDLOOP;
ENDIF;

Loop = Loop + 1;
ENDWHILE;

/*
create table objectlist as
select    object
from      object
where     basetype != 27 and
        terminals > 1;

select    ol.object
from      objectlist ol,
        terminal t1,
        terminal t2
where     ol.object = TEMP$objectlist.object and
        ol.object = t2.object and
        t1.terminal != t2.terminal and
        t1.node >= 10000 and
        t2.node >= 10000 and
        t1.kvlevel != t2.kvlevel;

*/
END;

```

2143652

96

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

*****
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Waxwing');
UPDATE ConductorType SET
  Resistance=0.238, Radius=0.008, GMR=0.006, Ampacity=440.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Partridge');
UPDATE ConductorType SET
  Resistance=0.236, Radius=0.008, GMR=0.007, Ampacity=440.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Merlin');
UPDATE ConductorType SET
  Resistance=0.189, Radius=0.009, GMR=0.141, Ampacity=500.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Linnet');
UPDATE ConductorType SET
  Resistance=0.187, Radius=0.009, GMR=0.137, Ampacity=510.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Ibis');
UPDATE ConductorType SET
  Resistance=0.159, Radius=0.010, GMR=0.134, Ampacity=570.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Pelican');
UPDATE ConductorType SET
  Resistance=0.133, Radius=0.010, GMR=0.134, Ampacity=625.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Flicker');
UPDATE ConductorType SET
  Resistance=0.133, Radius=0.011, GMR=0.132, Ampacity=635.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Hawk');
UPDATE ConductorType SET
  Resistance=0.132, Radius=0.011, GMR=0.131, Ampacity=640.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Hen');
UPDATE ConductorType SET
  Resistance=0.131, Radius=0.011, GMR=0.129, Ampacity=645.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Osprey');

```

2143652

97

```
UPDATE ConductorType SET
  Resistance=0.115, Radius=0.011, GMR=0.132, Ampacity=690.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Parakeet');
UPDATE ConductorType SET
  Resistance=0.114, Radius=0.012, GMR=0.129, Ampacity=700.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Dove');
UPDATE ConductorType SET
  Resistance=0.113, Radius=0.012, GMR=0.128, Ampacity=710.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Rook');
UPDATE ConductorType SET
  Resistance=0.100, Radius=0.012, GMR=0.126, Ampacity=765.000
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 44, 'Grosbeak');
UPDATE ConductorType SET
  Resistance=0.099, Radius=0.013, GMR=0.126, Ampacity=775.000
  WHERE Object = (Select value from lastkey where lastkey='Object');

INSERT INTO Grouping (SystemVersion, Grouping, GroupingType, Member, MemberType,
  SELECT 0, 42, 42, Object, 44, 52
  FROM Object where Type=44 AND Object != 44;
```

2143652

98

```

INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 43, 'BPA 500');
Update TowerType SET
a1_offset=-6.707, a1_height=21.951,
b1_offset=0.000, b1_height=31.402,
c1_offset=6.707, c1_height=21.951,
g1_offset=-3.811, g1_height=38.567,
a2_offset=0.000, a2_height=0.000,
b2_offset=0.000, b2_height=0.000,
c2_offset=0.000, c2_height=0.000,
g2_offset=3.811, g2_height=38.567
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 43, 'TVA 500');
Update TowerType SET
a1_offset=-12.195, a1_height=33.232,
b1_offset=0.000, b1_height=33.232,
c1_offset=12.195, c1_height=33.232,
g1_offset=-10.061, g1_height=42.988,
a2_offset=0.000, a2_height=0.000,
b2_offset=0.000, b2_height=0.000,
c2_offset=0.000, c2_height=0.000,
g2_offset=10.061, g2_height=42.988
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Object (Object, Type, Name)
  VALUES (-1, 43, 'APL 500');
Update TowerType SET
a1_offset=-9.223, a1_height=28.049,
b1_offset=0.000, b1_height=28.049,
c1_offset=9.223, c1_height=28.049,
g1_offset=-6.174, g1_height=37.957,
a2_offset=0.000, a2_height=0.000,
b2_offset=0.000, b2_height=0.000,
c2_offset=0.000, c2_height=0.000,
g2_offset=6.174, g2_height=37.957
  WHERE Object = (Select value from lastkey where lastkey='Object');
INSERT INTO Grouping (SystemVersion, Grouping, GroupingType, Member, MemberType,
  SELECT 0, 42, 42, Object, 43, 52
  FROM Object where Type=43 AND Object != 43;

```

2143652

AE Substation MCTN_AE

File Edit

Primary group: AE

In service: 09-Sep-1978 00:00

Position X: 206340.00

Owner #: MC206135TN

Description: 115kV switchyard

Comment:

Type: Substation

Out of service:

Position Y: 135135.00

Spec #: 78-135246

Name: MCTN_AE

Last edit:

Serial #:

Add

Replace

Network

Measurements

2143652

100

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                               /* Object being edited */
  OG = ARRAY OF GroupingClass;                     /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                  /* Measurements made on it */
  RO = ObjectSummaryClass;                         /* Exchange replacement object */
  i = INTEGER;                                     /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;

On UserEvent 'T.GetAttributes'=
  BEGIN
    ROLLBACK;
  END;

On UserEvent 'T.Update'=
  BEGIN
    COMMIT;
  END;
```

101

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

ON UserEvent 'O.GetAttributes' =
/*
*      Get the object attributes, then get the type specific attributes
*/
BEGIN
    REPEATED
    SELECT :O.Object = O.Object,
           :O.BaseType = O.BaseType,
           :O.Type = O.Type,
           :O.PrimaryGrouping = O.PrimaryGrouping,
           :O.TypeName = O.TypeName,
           :O.PrimaryGroupingName = O.PrimaryGroupingName,
           :O.Name = O.Name,
           :O.Description = O.Description,
           :O.Comment = O.Comment,
           :O.Terminals = O.Terminals,
           :O.InService = O.InService,
           :O.OutService = O.OutService,
           :O.PositionX = O.PositionX,
           :O.PositionY = O.PositionY,
           :O.OwnerNumber = O.OwnerNumber,
           :O.SerialNumber = O.SerialNumber,
           :O.SpecNumber = O.SpecNumber,
           :O.LastChanged = O.LastChanged,
           :O.LastChangedBy = O.LastChangedBy
    FROM   Object O
    WHERE  :Object = O.Object;

    INQUIRE_SQL( :i = ERRORNO );

    IF i != 0 THEN
        ROLLBACK;
        RESUME;
    ENDIF;

    CurFrame.WindowTitle =
        Squeeze (O.PrimaryGroupingName + ' ' + O.TypeName + ' ' + O.Name);

/*
*      If the object has no terminals, disallow access to the Network button
*/
    IF O.Terminals <= 0 THEN
        FIELD(Network).CurBias = FB_Dimmed;
    ENDIF;

/*
*      Mark the identifiers as unchanged
*/
    FIELD(O.PrimaryGroupingName).HasDataChanged = FALSE;

```

2143652

102

```

FIELD(O.TypeName).HasDataChanged = FALSE;
FIELD(O.Name).HasDataChanged = FALSE;
/*
 *
 * Get the groupings and measurements also
 */
OG.Clear();
i = 1;
REPEATED
SELECT :OG[i].SystemVersion = G.SystemVersion,
       :OG[i].Grouping = G.Grouping,
       :OG[i].GroupingType = G.GroupingType,
       :OG[i].Relationship = G.Relationship,
       :OG[i].Reference = G.Reference

FROM   Grouping G
WHERE  :Object = G.Member
BEGIN
    i = i + 1;
END;

OM.Clear();
i = 1;
REPEATED
SELECT :OM[i].MeasurandType = M.MeasurandType,
       :OM[i].Terminal1 = M.Terminal1,
       :OM[i].Terminal2 = M.Terminal2,
       :OM[i].SensorType = M.SensorType,
       :OM[i].SensorMinimum = M.SensorMinimum,
       :OM[i].SensorMaximum = M.SensorMaximum,
       :OM[i].SensorAccuracy = M.SensorAccuracy,
       :OM[i].ReversePolarity = M.ReversePolarity,
       :OM[i].MeasurerType = M.MeasurerType,
       :OM[i].Measurer = M.Measurer,
       :OM[i].Alarm = M.Alarm,
       :OM[i].History = M.History,
       :OM[i].Trigger = M.Trigger,
       :OM[i].EventLog = M.EventLog,
       :OM[i].Input = M.Input,
       :OM[i].Output = M.Output

FROM   Measurement M
WHERE  :Object = M.Measurand
BEGIN
    i = i + 1;
END;

CurFrame.SendUserEvent (EventName = 'T.GetAttributes');
END;

```

103

```

ON Click Menu.File.Add,
ON Click Add,
ON UserEvent 'O.Insert' =
/*
*      Set up the object, get the key set up by the insert, and
*      update the object associated with it as part of the same transaction.
*/
  BEGIN
    REPEATED
      INSERT INTO Object
        (Object
         ,Name
         ,Type
         ,TypeName
         ,PrimaryGrouping
         ,PrimaryGroupingName
         ,Terminals
        )
      VALUES
        (-1
         ,:O.Name
         ,:O.Type
         ,:O.TypeName
         ,:O.PrimaryGrouping
         ,:O.PrimaryGroupingName
         ,:O.Terminals
        );

    INQUIRE_SQL( :i = ERRORNO );

    IF i != 0 THEN
      ROLLBACK;
      RESUME;
    ENDIF;

    REPEATED
      SELECT :O.Object = K.Value
      FROM   LastKey K
      WHERE  K.LastKey = 'Object';

    Object = O.Object;
  /*
  *      Mark the identifiers as unchanged to prevent warnings
  *      on the update.
  */
    FIELD(O.PrimaryGroupingName).HasDataChanged = FALSE;
    FIELD(O.TypeName).HasDataChanged = FALSE;
    FIELD(O.Name).HasDataChanged = FALSE;

    CurFrame.SendUserEvent (EventName = 'O.Update');
  END;

ON Click Menu.File.Replace,
ON Click Replace,
ON UserEvent 'O.Update' =
  BEGIN
  /*
  *      Check if any of the identifier fields have changed. If so, warn
  *      the user, as this may be a hit on Replace when Add was intended
  */
    IF (FIELD(O.PrimaryGroupingName).HasDataChanged = TRUE OR
        FIELD(O.TypeName).HasDataChanged = TRUE OR
        FIELD(O.Name).HasDataChanged = TRUE) THEN
  /****
  Replace with message read from Problem ****/
  /****
  (V2.0 also use defaulting of buttons) ****/

```

2143652

104

```

IF (CurFrame.ConfirmPopup
    (MessageText = 'You will overwrite an existing object') =
    PU_Cancel) THEN
    ROLLBACK;
    RESUME;
ENDIF;

ENDIF;

/*
* Update the enterable parts of the object record in the database,
* then update the type specific details.
*/
REPEATED
UPDATE Object O
SET
    Type = :O.Type,
    PrimaryGrouping = :O.PrimaryGrouping,
    TypeName = :O.TypeName,
    PrimaryGroupingName = :O.PrimaryGroupingName,
    Name = :O.Name,
    Description = :O.Description,
    Comment = :O.Comment,
    InService = :O.InService,
    OutService = :O.OutService,
    PositionX = :O.PositionX,
    PositionY = :O.PositionY,
    OwnerNumber = :O.OwnerNumber,
    SerialNumber = :O.SerialNumber,
    SpecNumber = :O.SpecNumber,
    LastChanged = Date ('now'),
    LastChangedBy = DBMSInfo ('Username')
WHERE :Object = O.Object;

/*
* Trash all the groupings in the database and replace them
*/
REPEATED
DELETE FROM Grouping
WHERE :Object = Member;

i = 1;
WHILE i <= OG.LastRow() DO
    REPEATED
    INSERT INTO Grouping
        (SystemVersion
        ,Grouping
        ,GroupingType
        ,Member
        ,MemberType
        ,Relationship
        ,Reference
        )
    VALUES
        (:OG[i].SystemVersion
        ,:OG[i].Grouping
        ,:OG[i].GroupingType
        ,:Object
        ,:O.Type
        ,:OG[i].Relationship
        ,:OG[i].Reference
        );
    i = i + 1;
ENDWHILE;

/*
* Trash all the measurements in the database and replace them
*/
REPEATED
DELETE FROM Measurement

```

2143652

105

```

WHERE      :Object = Measurand;

i = 1;
WHILE i <= OM.LastRow() DO
  REPEATED
  INSERT INTO Measurement
    (Measurand
    ,MeasurandType
    ,Terminal1
    ,Terminal2
    ,SensorType
    ,SensorMinimum
    ,SensorMaximum
    ,SensorAccuracy
    ,ReversePolarity
    ,MeasurerType
    ,Measurer
    ,Alarm
    ,History
    ,Trigger
    ,EventLog
    ,Input
    ,Output
    )
  VALUES
    (:Object
    ,:OM[i].MeasurandType
    ,:OM[i].Terminal1
    ,:OM[i].Terminal2
    ,:OM[i].SensorType
    ,:OM[i].SensorMinimum
    ,:OM[i].SensorMaximum
    ,:OM[i].SensorAccuracy
    ,:OM[i].ReversePolarity
    ,:OM[i].MeasurerType
    ,:OM[i].Measurer
    ,:OM[i].Alarm
    ,:OM[i].History
    ,:OM[i].Trigger
    ,:OM[i].EventLog
    ,:OM[i].Input
    ,:OM[i].Output
    );
  i = i + 1;
ENDWHILE;

COMMIT;

CurFrame.SendUserEvent (EventName = 'T.Update');
CurFrame.SendUserEvent (EventName = 'O.GetAttributes');
END;

```

2143652

106

```

ON Click Menu.File.Exchange =
  BEGIN
/*
*   Go and get the piece of equipment to replace this one
*/
    RO = CALLFRAME FindObject
        (SuperType = O.BaseType
        ) WITH
        WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;

    IF RO IS NULL THEN
        ROLLBACK;
        RESUME;
    ENDIF;
/*
*   Check that the replacement has the same number of terminals as the
*   current object.
*/
    SELECT :i = COUNT(*)
    FROM    Object O,
           Object R
    WHERE   :O.Object = O.Object AND
           :RO.Object = R.Object AND
           O.Terminals = R.Terminals;

    IF i = 0 THEN
        MESSAGE 'The replacement has a different number of terminals.';
        ROLLBACK;
        RESUME;
    ENDIF;
/*
*   Exchange all the terminals, groupings and measurements. Negate
*   the keys to prevent duplicates during the exchange.
*/
    UPDATE Terminal T1
    FROM    Terminal T2
    SET     Object = - T2.Object
    WHERE   (:O.Object = T1.Object AND :RO.Object = T2.Object) OR
           (:RO.Object = T1.Object AND :O.Object = T2.Object);

    UPDATE Terminal
    SET     Object = - Object
    WHERE   Object < 0;

    UPDATE Grouping G1
    FROM    Grouping G2
    SET     Member = - G2.Member,
           MemberType = G2.MemberType
    WHERE   (:O.Object = G1.Member AND :RO.Object = G2.Member) OR
           (:RO.Object = G1.Member AND :O.Object = G2.Member);

    UPDATE Grouping
    SET     Member = - Member
    WHERE   Member < 0;

    UPDATE Measurement M1
    FROM    Measurement M2
    SET     Measurand = - M2.Measurand
    WHERE   (:O.Object = M1.Measurand AND :RO.Object = M2.Measurand) OR
           (:RO.Object = M1.Measurand AND :O.Object = M2.Measurand);

    UPDATE Measurement
    SET     Measurand = - Measurand
    WHERE   Measurand < 0;
/*

```

2143652

107

```
*      Now change the primary grouping and name
*/
UPDATE  Object O1
FROM    Object O2
SET     Type = - O2.Type,
        PrimaryGrouping = O2.PrimaryGrouping,
        Name = O2.Name
WHERE   (:O.Object = O1.Object AND :RO.Object = O2.Object) OR
        (:RO.Object = O1.Object AND :O.Object = O2.Object);

UPDATE  Object
SET     Type = - Type
WHERE   Type < 0;

COMMIT;

/*
*      Stay with the same network position - so change to the new object
*/
Object = RO.Object;
CurFrame.SendUserEvent (EventName = 'O.GetAttributes');
END;
```

2143652

108

```
ON Click Menu.Edit.Type,
ON Entry O.TypeName =
/*
 *      Show the user the types that this object can belong to, so
 *      that one may be selected
 */
BEGIN
    i = O.Type;
    CALLFRAME SetType (Object = O);

    IF i != O.Type THEN
        FIELD(O.TypeName).HasDataChanged = TRUE;
    ENDIF;

    RESUME;          /* Keep input focus out of this field */
END;

ON Click Menu.Edit.Groupings,
ON Entry O.PrimaryGroupingName =
/*
 *      Show the user the groups that this object belongs to, so
 *      that they can be edited
 */
BEGIN
    i = O.PrimaryGrouping;
    CALLFRAME SetGrouping (Object = O, OG = OG);

    IF i != O.PrimaryGrouping THEN
        FIELD(O.PrimaryGroupingName).HasDataChanged = TRUE;
    ENDIF;

    RESUME;          /* Keep input focus out of this field */
END;
```

2143652

109

```
ON Click Menu.Edit.Network_1,
ON Click Network =

    BEGIN
        GlobalFrame[Frame$ConnectivityEditor].Frame.SendUserEvent
            (EventName = 'LoadObject_D1', MessageInteger = Object);
    END;

ON Click Menu.Edit.Network_2,
ON Details Network =

    BEGIN
        GlobalFrame[Frame$ConnectivityEditor].Frame.SendUserEvent
            (EventName = 'LoadObject_D2', MessageInteger = Object);
    END;

ON Click Menu.Edit.Measurements,
ON Click Measurements =
/*
 *      Show the user the measurements made on this object, so
 *      that they can be edited
 */
    BEGIN
        CALLFRAME SetMeasurement (Object = O, OM = OM);
    END;
```

2143652

110

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

```

/*
**      Application:  PowerSystemModel
**      Procedure:    ObjectEditor
*/
PROCEDURE ObjectEditor
(
  OS = ObjectSummaryClass;                                /* Object summary */
  FrameName = VARCHAR(20) NOT NULL;                       /* Editor frame name */
  FrameTitle = VARCHAR(80) NOT NULL;                      /* Editor frame title */
) =
BEGIN
  REPEATED
  SELECT  :FrameName = BT.EditorName
  FROM    BaseType BT
  WHERE   :OS.BaseType = BT.Type;

  IF FrameName = '' THEN
    INSERT INTO GrassCatcher (Object, Text, Problem)
      VALUES (0, OS.TypeName, 'Type_006');
    ROLLBACK;
    RETURN;
  ENDIF;

  FrameTitle = OS.PrimaryGroupingName + ' ' + OS.TypeName + ' ' + OS.Name;
  FrameTitle = Squeeze (FrameTitle);
  OPENFRAME :FrameName (Object = OS.Object) WITH
    WindowTitle = FrameTitle,
    ParentFrame = NULL;                                /* Should be parent frame of procedure
  RETURN;
END;

```

2143652

111

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
**      Application:  PowerSystemModel
**      Procedure:    LoadTypeList
*/
PROCEDURE LoadTypeList
(
  EF = EnumField;                                /* Enumerated field to be set up
  StartType = INTEGER;                          /* Type to start with */
  Type = TypeClass;                             /* Scratch type */
  CList = ChoiceList;                          /* Scratch choice list entry */
  i = INTEGER NOT NULL;                        /* Scratch integer */
) =
BEGIN
/*
*      Set up the values in the choice list
*/
  CList = EF.ValueList;
  CList.ChoiceItems.Clear();
  CList.ChoiceItems[1].EnumValue = 0;
  CList.ChoiceItems[1].EnumText = '';
  i = 2;

  REPEATED
  SELECT  :Type.Type = T.Type,
          :Type.Name = T.Name
  FROM    Type T,
          ExtendedType ET
  WHERE   :StartType = ET.SuperType AND
          ET.Type = T.Type
  ORDER BY Name
  BEGIN
    CList.ChoiceItems[i].EnumValue = Type.Type;
    CList.ChoiceItems[i].EnumText = Type.Name;
    i = i + 1;
  END;
/*
*      Ensure it is updated on the screen
*/
  EF.UpdChoiceList();
END;

```

2143652

112

/*****

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

*****/

```
/*
**      Application:      PowerSystemModel
**      Frame:           Menu.File.CloseButton
*/
ON Click Menu.File.CloseButton =

    BEGIN
        RETURN;
    END;
```

2143652

BUXM Capacitor HATF			
File Edit			
Primary group:	BUXM	Type:	Capacitor
In service:	23-Jul-1967 00:00	Out of service:	
Position X:		Position Y:	
Owner #:	GE345123	Spec #:	67-gpc-123
Description:	Buxmallian Substation 11.5 Bank		
Comment:	Blows cans frequently - proximity to smelter		
Nominal MVAR:		10.80	Voltage sensitivity:
		0.95	AVR delay:
			01:30
Add		Replace	Network
		Measurements	

2143652

114

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                               /* Object being edited */
  OG = ARRAY OF GroupingClass;                     /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                  /* Measurements made on it */
  RO = ObjectSummaryClass;                         /* Exchange replacement object */
  i = INTEGER;                                     /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
  SELECT :C.Object = C.Object,
         :C.NominalMVar = C.NominalMVar,
         :C.VoltSensitivity = C.VoltSensitivity,
         :C.AVRDelay = C.AVRDelay
  FROM   Capacitor C
  WHERE  :Object = C.Object;

  ROLLBACK;
END;
```

ON UserEvent 'T.Update' =

```
BEGIN
  REPEATED
  UPDATE Capacitor C
  SET    NominalMVar = :C.NominalMVar,
         VoltSensitivity = :C.VoltSensitivity,
         AVRDelay = :C.AVRDelay
  WHERE  :Object = C.Object;

  COMMIT;
END;
```

2143652

RTU SUBU-1

File Edit

Primary group:

In service:

Position X:

Owner #:

Description:

Comment:

Type: RTU

Out of service:

Position Y:

Spec #:

Name: SUBU-1

Last edit: 05-Apr-1992 14:55

Serial #:

Cable: 123BL456

Controller: LTA24

Object address: 12

Controller address: 0

Add

Replace

Network

Measurements

2143652

116

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER;                                      /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
    SELECT :C.Object = C.Object,
           :C.ObjectAddress = C.ObjectAddress,
           :C.Controller = C.Controller,
           :C.ControllerAddress = C.ControllerAddress,
           :C.Cable = C.Cable
    FROM   Computer C
    WHERE  :Object = C.Object;

  ROLLBACK;
END;
```

ON UserEvent 'T.Update' =

```
BEGIN
  REPEATED
    UPDATE Computer C
    SET   ObjectAddress = :C.ObjectAddress,
         Controller = :C.Controller,
         ControllerAddress = :C.ControllerAddress,
         Cable = :C.Cable
    WHERE :Object = C.Object;

  COMMIT;
END;
```

2143652

DCLineSegment FLAR-BRUM

FileEdit

Primary group:

In service:

04-Apr-1978 00:00

Position X:

Owner #:

FL123543

Description:

Burlington line

Comment:

PUD has their padlock on switch 3046

Type:

DCLineSegment

Out of service:

Position Y:

Spec #:

78-gpc-1342

Name:

FLAR-BRUM

Last edit:

05-Apr-1992 16:36

Serial #:

Ground conductor:

Flicker

Tower type:

APL 500

Length:

50.00

Phase conductor:

Grosbeak

Bundle count:

1

Bundle spacing:

0.0000

R%:

0.0000

X%:

0.0000

Bch%:

0.0000

Add

Replace

Network

Measurements

2143652

118

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                /* Object being edited */
  OG = ARRAY OF GroupingClass;     /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;  /* Measurements made on it */
  RO = ObjectSummaryClass;         /* Exchange replacement object */
  i = INTEGER;                     /* Scratch integer */

  N = INTEGER NOT NULL;            /* Count */
  A = FLOAT NOT NULL;              /* Scratch */
  B = FLOAT NOT NULL;              /* Scratch */
  kV = FLOAT;                      /* Voltage */
  Dab = FLOAT;                     /* GMD a phase to b phase */
  Dbc = FLOAT;                     /* GMD b phase to c phase */
  Dca = FLOAT;                     /* GMD c phase to a phase */
  Deq = FLOAT;                     /* Equivalent GMD */
  GMReq = FLOAT;                   /* Equivalent GMR for bundle */
  Radiuseq = FLOAT;                /* Equivalent radius for bundle */
  R = FLOAT;                       /* Series resistance */
  X = FLOAT;                       /* Series inductive reactance */
  Bch = FLOAT;                     /* Shunt susceptance */
  ZBase = FLOAT;                   /* Impedance base */
  Omega = FLOAT;                   /* Angular frequency */
  Pi = FLOAT NOT NULL;             /* Pi */
)=
BEGIN
  Pi = 3.141592654;
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

2143652

119

```

ON UserEvent 'T.GetAttributes'=
  BEGIN
    REPEATED
      SELECT :S.Object = S.Object,
             :S.R = S.R,
             :S.X = S.X,
             :S.Bch = S.Bch,
             :S.Length = S.Length,
             :S.TowerType = S.TowerType,
             :S.GroundConductorType = S.GroundConductorType,
             :S.PhaseConductorType = S.PhaseConductorType,
             :S.PhaseConductorCount = S.PhaseConductorCount,
             :S.PhaseConductorSpacing = S.PhaseConductorSpacing
      FROM   Conductor S
      WHERE  :Object = S.Object;

    REPEATED
      SELECT :kV = T.KVLevel
      FROM   Terminal T
      WHERE  :S.Object = T.Object;

    CurFrame.SendUserEvent (EventName = 'GetTowerType');
    CurFrame.SendUserEvent (EventName = 'GetGroundConductorType');
    CurFrame.SendUserEvent (EventName = 'GetPhaseConductorType');

    ROLLBACK;
  END;

On UserEvent 'GetTowerType'=
  BEGIN
    REPEATED
      SELECT :S.TT.Object = TT.Object,
             :S.TT.Name = TTO.Name,
             :S.TT.A1_Offset = TT.A1_Offset,
             :S.TT.A1_Height = TT.A1_Height,
             :S.TT.B1_Offset = TT.B1_Offset,
             :S.TT.B1_Height = TT.B1_Height,
             :S.TT.C1_Offset = TT.C1_Offset,
             :S.TT.C1_Height = TT.C1_Height,
             :S.TT.G1_Offset = TT.G1_Offset,
             :S.TT.G1_Height = TT.G1_Height,
             :S.TT.A2_Offset = TT.A2_Offset,
             :S.TT.A2_Height = TT.A2_Height,
             :S.TT.B2_Offset = TT.B2_Offset,
             :S.TT.B2_Height = TT.B2_Height,
             :S.TT.C2_Offset = TT.C2_Offset,
             :S.TT.C2_Height = TT.C2_Height,
             :S.TT.G2_Offset = TT.G2_Offset,
             :S.TT.G2_Height = TT.G2_Height
      FROM   TowerType TT,
             Object TTO
      WHERE  :S.TowerType = TT.Object AND
             :S.TowerType = TTO.Object;

    ROLLBACK;
  END;

On UserEvent 'GetGroundConductorType'=
  BEGIN
    REPEATED
      SELECT :S.GCT.Object = GCT.Object,
             :S.GCT.Name = GCTO.Name,
             :S.GCT.Resistance = GCT.Resistance,
             :S.GCT.Radius = GCT.Radius,
             :S.GCT.GMR = GCT.GMR,

```

2143652

120

```

        :S.GCT.Ampacity = GCT.Ampacity
FROM      ConductorType GCT,
          Object GCTO
WHERE     :S.GroundConductorType = GCT.Object AND
          :S.GroundConductorType = GCTO.Object;

ROLLBACK;
END;

On UserEvent 'GetPhaseConductorType'=
BEGIN
    REPEATED
    SELECT :S.PCT.Object = PCT.Object,
           :S.PCT.Name = PCTO.Name,
           :S.PCT.Resistance = PCT.Resistance,
           :S.PCT.Radius = PCT.Radius,
           :S.PCT.GMR = PCT.GMR,
           :S.PCT.Ampacity = PCT.Ampacity
FROM      ConductorType PCT,
          Object PCTO
WHERE     :S.PhaseConductorType = PCT.Object AND
          :S.PhaseConductorType = PCTO.Object;

ROLLBACK;
END;

ON UserEvent 'T.Update' =
BEGIN
    REPEATED
    UPDATE Conductor S
    SET     R = :S.R,
           X = :S.X,
           Bch = :S.Bch,
           Length = :S.Length,
           TowerType = :S.TowerType,
           GroundConductorType = :S.GroundConductorType,
           PhaseConductorType = :S.PhaseConductorType,
           PhaseConductorCount = :S.PhaseConductorCount,
           PhaseConductorSpacing = :S.PhaseConductorSpacing
WHERE     :Object = S.Object;

    COMMIT;
END;

ON Entry S.TT.Name =
/*
 *      Show the user the list of possible tower types
 */
BEGIN
    i = CALLFRAME SetObject (Type = Type$TowerType
                           ,DefaultObject = S.TowerType);

    IF i IS NOT NULL THEN
        S.TowerType = i;
        CurFrame.SendUserEvent (EventName = 'GetTowerType');
        CurFrame.SendUserEvent (EventName = 'Recalculate');
    ENDIF;

    RESUME;          /* Keep input focus out of this field */
END;

ON Entry S.GCT.Name =
/*
 *      Show the user the list of possible conductor types
 */
BEGIN

```

2143652

```

                                121
i = CALLFRAME SetObject (Type = Type$ConductorType
                        ,DefaultObject = S.GroundConductorType);

IF i IS NOT NULL THEN
    S.GroundConductorType = i;
    CurFrame.SendUserEvent (EventName = 'GetGroundConductorType');
    CurFrame.SendUserEvent (EventName = 'Recalculate');
ENDIF;

RESUME;          /* Keep input focus out of this field */
END;

ON Entry S.PCT.Name =
/*
*   Show the user the list of possible conductor types
*/
BEGIN
    i = CALLFRAME SetObject (Type = Type$ConductorType
                        ,DefaultObject = S.PhaseConductorType);

    IF i IS NOT NULL THEN
        S.PhaseConductorType = i;
        CurFrame.SendUserEvent (EventName = 'GetPhaseConductorType');
        CurFrame.SendUserEvent (EventName = 'Recalculate');
    ENDIF;

    RESUME;          /* Keep input focus out of this field */
END;
```

2143652

122

```

ON UserEvent 'Recalculate',
ON ChildSetValue S =
  BEGIN
    IF S.TowerType != 0 AND
       S.PhaseConductorType != 0 AND
       S.Length IS NOT NULL THEN
/*
**      Equivalent GMR and radius for bundles (from Dommel eq. 56)
**
*/
      N = S.PhaseConductorCount;
      IF S.PhaseConductorSpacing > (2.0 * S.PCT.Radius) THEN
        A = S.PhaseConductorSpacing / (2.0 * SIN (PI / N));
      ELSE
        A = (2.0 * S.PCT.Radius) / (2.0 * SIN (PI / N));
      ENDIF;

      GMReq = (N * S.PCT.GMR * (A ** (N - 1))) ** (1.0 / N);
      Radiuseq = (N * S.PCT.Radius * (A ** (N - 1))) ** (1.0 / N);
/*
**      Get the GMD for each phase pair and the equivalent GMD
**      (from Stevenson eq. 3.52)
**
*/
      A = ABS (S.TT.A1_Height - S.TT.B1_Height) ** 2;
      B = ABS (S.TT.A1_Offset - S.TT.B1_Offset) ** 2;
      Dab = (A + B) ** 0.5;

      IF S.TT.B2_Height != 0 THEN
        A = ABS (S.TT.A1_Height - S.TT.B2_Height) ** 2;
        B = ABS (S.TT.A1_Offset - S.TT.B2_Offset) ** 2;
        Dab = (Dab * ((A + B) ** 0.5)) ** 0.5;
      ENDIF;

      A = ABS (S.TT.B1_Height - S.TT.C1_Height) ** 2;
      B = ABS (S.TT.B1_Offset - S.TT.C1_Offset) ** 2;
      Dbc = (A + B) ** 0.5;

      IF S.TT.C2_Height != 0 THEN
        A = ABS (S.TT.B1_Height - S.TT.C2_Height) ** 2;
        B = ABS (S.TT.B1_Offset - S.TT.C2_Offset) ** 2;
        Dbc = (Dbc * ((A + B) ** 0.5)) ** 0.5;
      ENDIF;

      A = ABS (S.TT.C1_Height - S.TT.A1_Height) ** 2;
      B = ABS (S.TT.C1_Offset - S.TT.A1_Offset) ** 2;
      Dca = (A + B) ** 0.5;

      IF S.TT.A2_Height != 0 THEN
        A = ABS (S.TT.C1_Height - S.TT.A2_Height) ** 2;
        B = ABS (S.TT.C1_Offset - S.TT.A2_Offset) ** 2;
        Dca = (Dca * ((A + B) ** 0.5)) ** 0.5;
      ENDIF;

      Deq = (Dab * Dbc * Dca) ** (1.0 / 3.0);
/*
**      Get the series reactance value in Ohms / Km (from Dommel eq. 54)
**      and convert it to per cent for the segment length.
**
*/
      Omega = 2.0 * Pi * SP.Frequency;
      ZBase = (kV ** 2) / SP.MVABase;

      X = Omega * (2.0 * (10.0**-4)) * LOG (Deq / GMReq);
      S.X = X / ZBase * S.Length * 100.0;
/*
**      Get the shunt susceptance in Siemens / Km (from Dommel eq. 58)
**      and convert it to per cent for the segment length.

```

2143652

123

```
*/
      Bch = Omega * (10.0**-6) / (18.0 * LOG (Deq / Radiuseq));
      S.Bch = Bch * ZBase * S.Length * 100.0;
/*
**      Get the resistance for the bundle in Ohms / Unit length
**      and convert it to per cent for the segment length.
**      Add 2% for spiralling.
*/
      S.R = S.PCT.Resistance / S.PhaseConductorCount;
      S.R = S.R / ZBase * S.Length * 100.0;
      S.R = S.R * 1.02;

      ENDIF;

END;
```

2113652

ConductorType Flicker

File Edit

Primary group:

In service:

Position X:

Owner #:

Description:

Comment:

Type:

Out of service:

Position Y:

Spec #:

ConductorType

Name:

Last edit:

Serial #:

Flicker

Ampacity: 635

Resistance: 0.133000

Radius: 0.011000

GMR: 0.132000

Add

Replace

Network

Measurements

2143652

125

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
**      Application:      PowerSystemModel
**      Frame:           ConductorType
*/
INITIALIZE
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER;                                       /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;

ON UserEvent 'T.GetAttributes' =
  BEGIN
/*
**      Get the conductor details
*/
    SELECT  :C.Ampacity = C.Ampacity,
            :C.Resistance = C.Resistance,
            :C.Radius = C.Radius,
            :C.GMR = C.GMR
    FROM    ConductorType C
    WHERE   :Object = C.Object;

    ROLLBACK;
  END;

ON UserEvent 'T.Update' =
  BEGIN
    UPDATE  ConductorType C
    SET     Ampacity = :C.Ampacity,
            Resistance = :C.Resistance,
            Radius = :C.Radius,
            GMR = :C.GMR
    WHERE   :Object = C.Object;

    COMMIT;
  END;

```

2143652

ABIN_WE Load ABIN

File Edit

Primary group: ABIN_WE

In service:

Position X:

Owner #:

Description:

Comment:

Type: Load

Out of service:

Position Y:

Spec #:

Name: ABIN

Last edit:

Serial #:

Fixed

Nominal

Voltage dependence

Frequency dependence

Real power:

Reactive power:

0.000

0.000

259.851

-79.740

0.000

0.000

Add

Replace

Network

Measurements

2143652

127

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                      /* Object being edited */
  OG = ARRAY OF GroupingClass;           /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;        /* Measurements made on it */
  RO = ObjectSummaryClass;               /* Exchange replacement object */
  OS = ObjectSummaryClass;               /**** Temp ****/
  i = INTEGER;                           /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
    SELECT :C.Object = C.Object,
           :C.Pfixed = C.Pfixed,
           :C.Qfixed = C.Qfixed,
           :C.Pnom = C.Pnom,
           :C.Qnom = C.Qnom,
           :C.PowerFactor = C.PowerFactor,
           :C.PfixedPct = C.PfixedPct,
           :C.QfixedPct = C.QfixedPct,
           :C.PnomPct = C.PnomPct,
           :C.QnomPct = C.QnomPct,
           :C.PVexp = C.PVexp,
           :C.QVexp = C.QVexp,
           :C.PFexp = C.PFexp,
           :C.QFexp = C.QFexp
    FROM   Consumer C
    WHERE  :Object = C.Object;

  ROLLBACK;
END;
```

ON UserEvent 'T.Update' =

```
BEGIN
  REPEATED
    UPDATE Consumer C
    SET   Type = :O.Type,                 /**** Temporary - to speed rules ****/
          Pfixed = :C.Pfixed,
          Qfixed = :C.Qfixed,
          Pnom = :C.Pnom,
          Qnom = :C.Qnom,
          PowerFactor = :C.PowerFactor,
          PVexp = :C.PVexp,
```

2143652

128

```

        QVexp = :C.QVexp,
        PFexp = :C.PFexp,
        QFexp = :C.QFexp
WHERE    :Object = C.Object;

        COMMIT;
END;

/**** temporary special for consumers - make it easy to chase up load model ****
ON Properties O.PrimaryGroupName =
BEGIN
    IF O.PrimaryGrouping != Object$Null THEN
        REPEATED
        SELECT  :OS.Object = O.Object,
                :OS.BaseType = O.BaseType,
                :OS.Type = O.Type,
                :OS.PrimaryGrouping = O.PrimaryGrouping,
                :OS.TypeName = O.TypeName,
                :OS.Name = O.Name,
                :OS.PrimaryGroupName = O.PrimaryGroupName
        FROM    Object O
        WHERE   :O.PrimaryGrouping = O.Object;

        CALLPROC ObjectEditor (OS = OS);
    ENDIF;
END;
```

2143652

ALLE Generator COTU

File Edit

Primary group: ALLE

In service: 01-Jan-1945 00:00

Position X:

Owner #: WEST3455

Description: Westinghouse 72 MVA Unit

Comment: Last rewInd: 6/30/75

Type: Generator

Out of service:

Position Y:

Spec #: 45-gac-984578

Name: COTU

Last edit:

Serial #: 112233

Rated MVA: 72.00

Inertia: 2.22

Damping: 0.72

Minimum:

Base:

Maximum:

0.00

0.00

0.00

Ramp rate

Up: 5.00

Down: 5.00

AVR to manual lead:

AVR to manual lag:

Manual to AVR:

0.00

0.00

0.00

Participation factor: 0.00

MW

MVAR

kV

Add

Replace

Network

Measurements

2143652

130

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER;                                      /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
  SELECT :G.Object = G.Object,
         :G.RatedMVA = G.RatedMVA,
         :G.MinimumMW = G.MinimumMW,
         :G.BaseMW = G.BaseMW,
         :G.MaximumMW = G.MaximumMW,
         :G.MinimumMVA = G.MinimumMVA,
         :G.BaseMVA = G.BaseMVA,
         :G.MaximumMVA = G.MaximumMVA,
         :G.MinimumkV = G.MinimumkV,
         :G.MaximumkV = G.MaximumkV,
         :G.X = G.X,
         :G.R = G.R,
         :G.Inertia = G.Inertia,
         :G.Damping = G.Damping,
         :G.ManualToAVR = G.ManualToAVR,
         :G.AVRTToManualLag = G.AVRTToManualLag,
         :G.AVRTToManualLead = G.AVRTToManualLead,
         :G.NormalPF = G.NormalPF,
         :G.ShortPF = G.ShortPF,
         :G.LongPF = G.LongPF,
         :G.DownRampRate = G.DownRampRate,
         :G.UpRampRate = G.UpRampRate
  FROM   Generator G
  WHERE  :Object = G.Object;

  ROLLBACK;
END;
```

2143652

131

```
ON UserEvent 'T.Update' =
  BEGIN
    REPEATED
      UPDATE Generator G
      SET
        RatedMVA = :G.RatedMVA,
        MinimumMW = :G.MinimumMW,
        BaseMW = :G.BaseMW,
        MaximumMW = :G.MaximumMW,
        MinimumMVar = :G.MinimumMVar,
        BaseMVar = :G.BaseMVar,
        MaximumMVar = :G.MaximumMVar,
        MinimumkV = :G.MinimumkV,
        MaximumkV = :G.MaximumkV,
        X = :G.X,
        R = :G.R,
        Inertia = :G.Inertia,
        Damping = :G.Damping,
        ManualToAVR = :G.ManualToAVR,
        AVRToManualLag = :G.AVRToManualLag,
        AVRToManualLead = :G.AVRToManualLead,
        NormalPF = :G.NormalPF,
        ShortPF = :G.ShortPF,
        LongPF = :G.LongPF,
        DownRampRate = :G.DownRampRate,
        UpRampRate = :G.UpRampRate
      WHERE :Object = G.Object;

    COMMIT;
  END;
```

2143652

132

Dispatcher Brewer, Kathy

File Edit

Primary group:	Type:	Name:
	Dispatcher	Brewer, Kathy
In service:	Out of service:	Last edit:
26-Jun-1972 00:00		04-Apr-1992 14:18
Position X:	Position Y:	
Owner #:	Spec #:	Serial #:
81891		
Description:		
Comment:	Normally assigned to Eastern Division; has worked Western Division	

Initials: Login name: Password:

2143652

132.5

.....

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

*****/

INITIALIZE

```
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER;                                      /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =
BEGIN

```
  REPEATED
  SELECT :P.Object = P.Object,
         :P.Initials = P.Initials,
         :P.LoginName = P.LoginName,
         :P.Password = P.Password
  FROM   Person P
  WHERE  :Object = P.Object;
```

ROLLBACK;

END;

ON UserEvent 'T.Update' =
BEGIN

```
  REPEATED
  UPDATE Person P
  SET    Initials = :P.Initials,
         LoginName = :P.LoginName,
         Password = :P.Password
  WHERE  :Object = P.Object;
```

COMMIT;

END;

2143652

AMOS_WE Reactor 1

FileEdit

Primary group:AMOS_WE

In service:15-Jul-1989 00:00

Position X:

Owner #:79df709

Description:150 MVAR Reactor

Comment:

Type:Reactor

Out of service:

Position Y:

Spec #:89-acd-709

Name:1

Last edit:05-Apr-1992 16:47

Serial #:8904582

Nominal MVAR:150.00

Voltage sensitivity:0.85

AVR delay:00:30

Add

Replace

Network

Measurements

2143652

134

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER;                                      /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
  SELECT :R.Object = R.Object,
         :R.NominalMVar = R.NominalMVar,
         :R.VoltSensitivity = R.VoltSensitivity,
         :R.AVRDelay = R.AVRDelay
  FROM   Reactor R
  WHERE  :Object = R.Object;

  ROLLBACK;
END;
```

ON UserEvent 'T.Update' =

```
BEGIN
  REPEATED
  UPDATE Reactor R
  SET    NominalMVar = :R.NominalMVar,
         VoltSensitivity = :R.VoltSensitivity,
         AVRDelay = :R.AVRDelay
  WHERE  :Object = R.Object;

  COMMIT;
END;
```

2143652

135

File		Edit
Primary group:	ACTP	Type: AB
Name:	EXCH_1-NLND_SE	
In service:	30-Aug-1985 00:00	Last edit: 03-Apr-1992 15:34
Position X:		Position Y:
Owner #:	RPH1819	Spec #: GE135
Description:	GE 13.0 air blast circuit breaker	
Serial #:	9745763	
Comment:	Last overhauled 8/91	
☐ Normal open		
Add	Replace	Network
Measurements		

2143652

136

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

INITIALIZE

```
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER;                                      /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
    SELECT :S.Object = S.Object,
           :S.NormalOpen = S.NormalOpen
    FROM   Switch S
    WHERE  :Object = S.Object;

  ROLLBACK;
END;
```

ON UserEvent 'T.Update' =

```
BEGIN
  REPEATED
    UPDATE Switch S
    SET    NormalOpen = :S.NormalOpen
    WHERE  :Object = S.Object;

  COMMIT;
END;
```

2143652

SUBU-1 Analog 12.2

File

Edit

Primary group: SUBU-1

Type: Analog

Name: 12.2

In service:

Out of service:

Last edit: 04-Apr-1992 15:35

Position X:

Position Y:

Owner #:

Spec #:

Serial #:

Description:

Comment:

☒ Input

☐ Output

Card address: 12

Word address: 2

Size (blts):

12

Offset (blts):

0

XOR mask: 0

Power cable: 123LR324

Return cable: 423RL321

Termination cable: BLR432876

Ground cable: GR678411

Panel: RPH42

Add

Replace

Network

Measurements

2143652

138

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                      /* Object being edited */
  OG = ARRAY OF GroupingClass;           /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;        /* Measurements made on it */
  RO = ObjectSummaryClass;               /* Exchange replacement object */
  i = INTEGER;                           /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
  SELECT :T.Object = T.Object,
         :T.TelemetryMinimum = T.TelemetryMinimum,
         :T.TelemetryMaximum = T.TelemetryMaximum,
         :T.XORMask = T.XORMask,
         :T.Input = T.Input,
         :T.Output = T.Output,
         :T.CardAddress = T.CardAddress,
         :T.WordAddress = T.WordAddress,
         :T.Size = T.Size,
         :T.Offset = T.Offset,
         :T.PowerCable = T.PowerCable,
         :T.ReturnCable = T.ReturnCable,
         :T.GroundCable = T.GroundCable,
         :T.TerminationCable = T.TerminationCable,
         :T.Panel = T.Panel
  FROM   Telemetry T
  WHERE  :Object = T.Object;
```

```
ROLLBACK;
END;
```

ON UserEvent 'T.Update' =

```
BEGIN
  REPEATED
  UPDATE Telemetry T
  SET    TelemetryMinimum = :T.TelemetryMinimum,
         TelemetryMaximum = :T.TelemetryMaximum,
         XORMask = :T.XORMask,
         Input = :T.Input,
         Output = :T.Output,
         CardAddress = :T.CardAddress,
         WordAddress = :T.WordAddress,
```

2143652

139

```
Size = :T.Size,  
Offset = :T.Offset,  
PowerCable = :T.PowerCable,  
ReturnCable = :T.ReturnCable,  
GroundCable = :T.GroundCable,  
TerminationCable = :T.TerminationCable,  
Panel = :T.Panel  
WHERE :Object = T.Object;  
COMMIT;  
END;
```

2143652

TowerType APL 500

File Edit

Primary group: Specification

In service: ☐

Position X:

Owner #:

Description:

Comment:

Type: TowerType

Out of service: ☐

Position Y:

Spec #:

Name: APL 500

Last edit:

Serial #:

Circuit 1 A phase B phase C phase Ground

Height:

Offset:

Circuit 2 A phase B phase C phase Ground

Height:

Offset:

Add

Replace

Network

Measurements

2143652

141

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
**      Application:      PowerSystemModel
**      Frame:           TowerType
*/
INITIALIZE
(
  Object = INTEGER;                      /* Object being edited */
  OG = ARRAY OF GroupingClass;           /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;        /* Measurements made on it */
  RO = ObjectSummaryClass;               /* Exchange replacement object */
  i = INTEGER;                           /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;

ON UserEvent 'T.GetAttributes'=
  BEGIN
/*
**      Get the tower details
*/
    REPEATED
    SELECT :TT.A1_Offset = TT.A1_Offset,
           :TT.A1_Height = TT.A1_Height,
           :TT.B1_Offset = TT.B1_Offset,
           :TT.B1_Height = TT.B1_Height,
           :TT.C1_Offset = TT.C1_Offset,
           :TT.C1_Height = TT.C1_Height,
           :TT.G1_Offset = TT.G1_Offset,
           :TT.G1_Height = TT.G1_Height,
           :TT.A2_Offset = TT.A2_Offset,
           :TT.A2_Height = TT.A2_Height,
           :TT.B2_Offset = TT.B2_Offset,
           :TT.B2_Height = TT.B2_Height,
           :TT.C2_Offset = TT.C2_Offset,
           :TT.C2_Height = TT.C2_Height,
           :TT.G2_Offset = TT.G2_Offset,
           :TT.G2_Height = TT.G2_Height
    FROM   TowerType TT
    WHERE  :Object = TT.Object;

    ROLLBACK;
  END;

```

2143652

142

ON UserEvent 'T.Update' =

```
BEGIN
  UPDATE TowerType TT
  SET
    A1_Offset = :TT.A1_Offset,
    A1_Height = :TT.A1_Height,
    B1_Offset = :TT.B1_Offset,
    B1_Height = :TT.B1_Height,
    C1_Offset = :TT.C1_Offset,
    C1_Height = :TT.C1_Height,
    G1_Offset = :TT.G1_Offset,
    G1_Height = :TT.G1_Height,
    A2_Offset = :TT.A2_Offset,
    A2_Height = :TT.A2_Height,
    B2_Offset = :TT.B2_Offset,
    B2_Height = :TT.B2_Height,
    C2_Offset = :TT.C2_Offset,
    C2_Height = :TT.C2_Height,
    G2_Offset = :TT.G2_Offset,
    G2_Height = :TT.G2_Height
  WHERE :Object = TT.Object;

  COMMIT;
END;
```

2143652

Primary group:		Type:		Name:	
ALLE		Transformer		COTU	
In service:		Out of service:		Last edit:	
08-Aug-1978 00:00				05-Apr-1992 16:03	
Position X:		Position Y:			
Owner #:		Spec #:		Serial #:	
HIT25130		78-gvt-907800		9786709	
Description: Hitachi 12.5/138 kV OAFA					
Comment: 					

Phases:		G mag:		B mag:	
3 ☒		0.00		0.00	
		Mag sat flux:		B mag sat:	

Windings			Tap Setting Measurements					
Name	LTC	Nominal kV	Winding 1	Step	Winding 2	Step	Winding 3	Step
TRSL	0	12.50	TRSL	0	TRSH	0		0
TRSH	0	138.00		0		0		0
	0	0.00						

Add
Replace
Network
Measurements

2143652

144

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

INITIALIZE

```
(
  Object = INTEGER;                                /* Object being edited */
  OG = ARRAY OF GroupingClass;                      /* Groupings it belongs to */
  OM = ARRAY OF MeasurementClass;                   /* Measurements made on it */
  RO = ObjectSummaryClass;                          /* Exchange replacement object */
  i = INTEGER NOT NULL;                             /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent(EventName = 'O.GetAttributes');
END;
```

ON UserEvent 'T.GetAttributes' =

```
BEGIN
  REPEATED
  SELECT :T.Object = T.Object,
         :T.Phases = T.Phases,
         :T.GMag = T.GMag,
         :T.BMag = T.BMag,
         :T.MagSatFlux = T.MagSatFlux,
         :T.BMagSat = T.BMagSat
  FROM   Transformer T
  WHERE  :Object = T.Object;
```

```
/*
 * The number of terminals is determined by the number of windings,
 * and can be changed. Set the terminal count in the object to 0 to
 * prevent any confusion (especially if this is used as a template
 * for an Add)
 */
```

O.Terminals = 0;

```
CurFrame.SendUserEvent(EventName = 'W.GetAttributes');
CurFrame.SendUserEvent(EventName = 'TS.GetAttributes');
ROLLBACK;
```

END;

ON UserEvent 'W.GetAttributes' =

BEGIN

```
/*
 ** Get the winding data
 */
```

```
i = 1;
W.Clear();
```

```
REPEATED
SELECT :W[i].Object = W.Object,
       :W[i].Winding = W.Winding,
```

2143652

145

```
:W[i].Source = W.Source,
:W[i].Name = W.Name,
:W[i].LoadTapChanger = W.LoadTapChanger,
:W[i].NominalKV = W.NominalKV,
:W[i].NeutralKV = W.NeutralKV,
:W[i].BIL = W.BIL,
:W[i].RatedMVA = W.RatedMVA,
:W[i].ShortTermMVA = W.ShortTermMVA,
:W[i].EmergencyMVA = W.EmergencyMVA,
:W[i].LowStep = W.LowStep,
:W[i].HighStep = W.HighStep,
:W[i].NeutralStep = W.NeutralStep,
:W[i].StepSize = W.StepSize,
:W[i].PhaseShift = W.PhaseShift
FROM Winding W
WHERE :Object = W.Object
BEGIN
    i = i + 1;
END;

ROLLBACK;
END;
```

2143652

146

```

ON UserEvent 'TS.GetAttributes' =
  BEGIN
    /*
    **      Get the tap setting measurements
    */
    i = 1;
    TS.Clear();

    REPEATED
    SELECT :TS[i].Object = TS.Object,
           :TS[i].Winding1 = TS.Winding1,
           :TS[i].TapStep1 = TS.TapStep1,
           :TS[i].Winding2 = TS.Winding2,
           :TS[i].TapStep2 = TS.TapStep2,
           :TS[i].Winding3 = TS.Winding3,
           :TS[i].TapStep3 = TS.TapStep3,
           :TS[i].Source = TS.Source,
           :TS[i].R0_1_2 = TS.R0_1_2,
           :TS[i].Z0_1_2 = TS.Z0_1_2,
           :TS[i].R1_1_2 = TS.R1_1_2,
           :TS[i].Z1_1_2 = TS.Z1_1_2,
           :TS[i].R0_2_3 = TS.R0_2_3,
           :TS[i].Z0_2_3 = TS.Z0_2_3,
           :TS[i].R1_2_3 = TS.R1_2_3,
           :TS[i].Z1_2_3 = TS.Z1_2_3,
           :TS[i].R0_3_1 = TS.R0_3_1,
           :TS[i].Z0_3_1 = TS.Z0_3_1,
           :TS[i].R1_3_1 = TS.R1_3_1,
           :TS[i].Z1_3_1 = TS.Z1_3_1,
           :TS[i].R0_1 = TS.R0_1,
           :TS[i].Z0_1 = TS.Z0_1,
           :TS[i].X0_1 = TS.X0_1,
           :TS[i].R1_1 = TS.R1_1,
           :TS[i].Z1_1 = TS.Z1_1,
           :TS[i].X1_1 = TS.X1_1,
           :TS[i].R0_2 = TS.R0_2,
           :TS[i].Z0_2 = TS.Z0_2,
           :TS[i].X0_2 = TS.X0_2,
           :TS[i].R1_2 = TS.R1_2,
           :TS[i].Z1_2 = TS.Z1_2,
           :TS[i].X1_2 = TS.X1_2,
           :TS[i].R0_3 = TS.R0_3,
           :TS[i].Z0_3 = TS.Z0_3,
           :TS[i].X0_3 = TS.X0_3,
           :TS[i].R1_3 = TS.R1_3,
           :TS[i].Z1_3 = TS.Z1_3,
           :TS[i].X1_3 = TS.X1_3
    FROM   TapSetting TS
    WHERE  :Object = TS.Object
    BEGIN

      IF TS[i].Winding1 != 0 THEN
        TS[i].Winding1Name = W[TS[i].Winding1].Name;
      ENDIF;

      IF TS[i].Winding2 != 0 THEN
        TS[i].Winding2Name = W[TS[i].Winding2].Name;
      ENDIF;

      IF TS[i].Winding3 != 0 THEN
        TS[i].Winding3Name = W[TS[i].Winding3].Name;
      ENDIF;

      i = i + 1;
    END;

```

2143652

147

ROLLBACK;
END;

2143652

148

```

/*
**      Edit a transformer winding
**
ON ChildProperties W =
  BEGIN
    IF FIELD(W).ActiveRow = 0 THEN
      RESUME;
    ENDIF;

    CALLFRAME SetWinding (Winding = W[]);
  END;

ON DeleteRow W =
  BEGIN
    IF FIELD(W).ActiveRow = 0 THEN
      RESUME;
    ENDIF;
  /*
  *      Check that the winding is not referenced by a tap setting
  */
  i = 1;
  WHILE i <= TS.LastRow() DO

    IF TS[i].Winding1 = W[].Winding OR
       TS[i].Winding2 = W[].Winding OR
       TS[i].Winding3 = W[].Winding THEN
      FIELD(TS).ActiveRow = i;
      INSERT INTO GrassCatcher (Object, Text, Problem)
        VALUES (:W[].Object, :W[].Name, 'Transformer_015');
      ROLLBACK;
      RESUME;
    ENDIF;

    i = i + 1;
  ENDWHILE;
/*
*      Back up any references to windings beyond this one
*/
  i = 1;
  WHILE i <= TS.LastRow() DO

    IF TS[i].Winding1 > W[].Winding THEN
      TS[i].Winding1 = TS[i].Winding1 - 1;
    ENDIF;

    IF TS[i].Winding2 > W[].Winding THEN
      TS[i].Winding2 = TS[i].Winding2 - 1;
    ENDIF;

    IF TS[i].Winding3 > W[].Winding THEN
      TS[i].Winding3 = TS[i].Winding3 - 1;
    ENDIF;

    i = i + 1;
  ENDWHILE;

  W.RemoveRow(RowNumber = FIELD(W).CurRow);

END;

ON ClearTable W =
  BEGIN
    IF TS.LastRow() != 0 THEN
      INSERT INTO GrassCatcher (Object, Problem)

```

2143652

```

                                149
                                VALUES (:W[]Object, 'Transformer_016');
                                ROLLBACK;
                                RESUME;
                                ENDIF;
                                W.Clear();
                                END;
```

2143652

150

```
        Edit a transformer tap setting measurement
ChildProperties TS =
  BEGIN
    IF FIELD(TS).ActiveRow = 0 THEN
      RESUME;
    ENDIF;

    OPENFRAME SetTapSetting (TapSetting = TS[], W = W) WITH
      WindowTitle = CurFrame.WindowTitle;
  END;
```

2143652

151

```
ON UserEvent 'T.Update' =
  BEGIN
    REPEATED
      UPDATE Transformer T
      SET
        Phases = :T.Phasess,
        GMag = :T.GMag,
        BMag = :T.BMag,
        MagSatFlux = :T.MagSatFlux,
        BMagSat = :T.BMagSat
      WHERE :Object = T.Object;

      CurFrame.SendUserEvent (EventName = 'W.Update');
      CurFrame.SendUserEvent (EventName = 'TS.Update');
    COMMIT;
  END;
```

2143652

152

```

ON UserEvent 'W.Update' =
  BEGIN
/*
*   Preserve any terminals that are defined for the transformer by
*   altering the terminal number so that it is not the same as the
*   winding number, then delete all the windings.
*/
    REPEATED
    UPDATE Terminal T
    SET Terminal = -Terminal
    WHERE :Object = T.Object;

    REPEATED
    DELETE FROM Winding W
    WHERE :Object = W.Object;
/*
*   Loop through the table field, inserting each winding in turn.
*/
    i = 1;

    WHILE i <= W.LastRow() DO
      REPEATED
      INSERT INTO Winding
        (Object
        ,Name
        ,LoadTapChanger
        ,NominalKV
        ,NeutralKV
        ,BIL
        ,RatedMVA
        ,ShortTermMVA
        ,EmergencyMVA
        ,LowStep
        ,HighStep
        ,NeutralStep
        ,StepSize
        ,PhaseShift
        )
      VALUES
        (:Object
        ,:W[i].Name
        ,:W[i].LoadTapChanger
        ,:W[i].NominalKV
        ,:W[i].NeutralKV
        ,:W[i].BIL
        ,:W[i].RatedMVA
        ,:W[i].ShortTermMVA
        ,:W[i].EmergencyMVA
        ,:W[i].LowStep
        ,:W[i].HighStep
        ,:W[i].NeutralStep
        ,:W[i].StepSize
        ,:W[i].PhaseShift
        );

      IF IIErrorNumber != 0 THEN
        FIELD(W).ActiveRow = i;
        CurFrame.PurgeUserEvent();
        ROLLBACK;
        RESUME;
      ENDIF;

      i = i + 1;
    ENDWHILE;
/*

```

2143652

153

* Update the new terminal associated with the winding with the
 * connectivity information from the old terminal that was
 * preserved above. The match is done on the KVlevel - this means
 * that is the NominalKV of a winding is changed the terminal
 * will get disconnected automatically and will have to be
 * reconnected by hand. Then blow the old terminals away and ensure
 * that the object's maximum terminal count is correct.
 */

```

REPEATED
UPDATE Terminal NT
FROM Terminal OT
SET Node = OT.Node
WHERE :Object = NT.Object AND
      NT.Terminal > 0 AND
      :Object = OT.Object AND
      OT.Terminal < 0 AND
      OT.KVLevel = NT.KVLevel;

```

```

REPEATED
DELETE FROM Terminal
WHERE :Object = Object AND
      Terminal < 0;

```

```

i = W.LastRow();
REPEATED
UPDATE Object O
SET Terminals = :i
WHERE :Object = O.Object;

```

```

COMMIT;
END;

```

2143652

154

```

ON UserEvent 'TS.Update' =
  BEGIN
/*
*   Blow away all the current measurements, and replace
*   them with a new set (too many things can be edited to allow
*   clean identification of the old row)
*/
    REPEATED
    DELETE FROM TapSetting TS
    WHERE   :Object = TS.Object;

    i = 1;

    WHILE i <= TS.LastRow() DO
/*
**   Insert the new measurements
*/
        REPEATED
        INSERT INTO TapSetting
        (
            Object,
            Winding1,
            TapStep1,
            Winding2,
            TapStep2,
            Winding3,
            TapStep3,
            Source,
            R1_1_2,
            Z1_1_2,
            R0_1_2,
            Z0_1_2,
            R1_2_3,
            Z1_2_3,
            R0_2_3,
            Z0_2_3,
            R1_3_1,
            Z1_3_1,
            R0_3_1,
            Z0_3_1,
            R1_1,
            Z1_1,
            X1_1,
            R0_1,
            Z0_1,
            X0_1,
            R1_2,
            Z1_2,
            X1_2,
            R0_2,
            Z0_2,
            X0_2,
            R1_3,
            Z1_3,
            X1_3,
            R0_3,
            Z0_3,
            X0_3
        )
        VALUES
        (
            :Object,
            :TS[i].Winding1,
            :TS[i].TapStep1,
            :TS[i].Winding2,

```

2143652

155

```

:TS[i].TapStep2,
:TS[i].Winding3,
:TS[i].TapStep3,
:TS[i].Source,
:TS[i].R1_1_2,
:TS[i].Z1_1_2,
:TS[i].R0_1_2,
:TS[i].Z0_1_2,
:TS[i].R1_2_3,
:TS[i].Z1_2_3,
:TS[i].R0_2_3,
:TS[i].Z0_2_3,
:TS[i].R1_3_1,
:TS[i].Z1_3_1,
:TS[i].R0_3_1,
:TS[i].Z0_3_1,
:TS[i].R1_1,
:TS[i].Z1_1,
:TS[i].X1_1,
:TS[i].R0_1,
:TS[i].Z0_1,
:TS[i].X0_1,
:TS[i].R1_2,
:TS[i].Z1_2,
:TS[i].X1_2,
:TS[i].R0_2,
:TS[i].Z0_2,
:TS[i].X0_2,
:TS[i].R1_3,
:TS[i].Z1_3,
:TS[i].X1_3,
:TS[i].R0_3,
:TS[i].Z0_3,
:TS[i].X0_3
);

IF IIErrNumber != 0 THEN
    FIELD(TS).ActiveRow = i;
    CurFrame.PurgeUserEvent();
    ROLLBACK;
    RESUME;
ENDIF;

    i = i + 1;
ENDWHILE;

COMMIT;
END;

```

2143652

[illegible]

2143652

157

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

INITIALIZE

```
(
  Winding = WindingClass;          /* Passed winding */
)=
BEGIN
  W = Winding.Duplicate();
END;
```

On Entry W.NominalKV =

```
BEGIN
  W.NominalKV = CALLFRAME SetKVLevel (KVLevel = W.NominalKV);
  RESUME; /* Keep input focus out of this field */
END;
```

ON Click Apply_Button =

```
BEGIN
  Winding.Name = W.Name;
  Winding.LoadTapChanger = W.LoadTapChanger;
  Winding.NominalKV = W.NominalKV;
  Winding.NeutralKV = W.NeutralKV;
  Winding.BIL = W.BIL;
  Winding.RatedMVA = W.RatedMVA;
  Winding.ShortTermMVA = W.ShortTermMVA;
  Winding.EmergencyMVA = W.EmergencyMVA;
  Winding.LowStep = W.LowStep;
  Winding.HighStep = W.HighStep;
  Winding.NeutralStep = W.NeutralStep;
  Winding.StepSize = W.StepSize;
  Winding.PhaseShift = W.PhaseShift;
  RETURN TRUE;
END;
```

ON Click Cancel_Button =

```
BEGIN
  RETURN FALSE;
END;
```

2143652

[illegible]

2143652

159

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

 INITIALIZE

```
(
  TapSetting = TapSettingClass;          /* Passed tap setting */
  W = ARRAY OF WindingClass;             /* Passed windings */

  i = INTEGER NOT NULL;                  /* Scratch integer */

  MinValue = INTEGER;                    /* Min value for slider */
  MaxValue = INTEGER;                    /* Max value for slider */
  NameplateMVA = FLOAT;                  /* Rated MVA */
  KVCorrection = FLOAT;                  /* kV correction for tap setting */
  NameplateKV = FLOAT;                  /* kV of measurement */
  CF1 = FLOAT;                          /* Winding 1 correction factor */
  CF2 = FLOAT;                          /* Winding 2 correction factor */
  CF3 = FLOAT;                          /* Winding 3 correction factor */
)=
BEGIN
  FIELD(TS.Winding1).ValueList.ChoiceItems.Clear();
  FIELD(TS.Winding1).ValueList.ChoiceItems[1].EnumValue = 0;
  FIELD(TS.Winding1).ValueList.ChoiceItems[1].EnumText = '';
  i = 1;

  WHILE i <= W.LastRow() DO
    FIELD(TS.Winding1).ValueList.ChoiceItems[i+1].EnumValue = i;
    FIELD(TS.Winding1).ValueList.ChoiceItems[i+1].EnumText = W[i].Name;
    i = i + 1;
  ENDWHILE;

  FIELD(TS.Winding2).ValueList = FIELD(TS.Winding1).ValueList;
  FIELD(TS.Winding3).ValueList = FIELD(TS.Winding1).ValueList;

  TS = TapSetting.Duplicate();

  CurFrame.SendUserEvent(EventName = 'TS.Winding1');
  CurFrame.SendUserEvent(EventName = 'TS.Winding2');
  CurFrame.SendUserEvent(EventName = 'TS.Winding3');
END;
```

2143652

160

```
ON UserEvent 'TS.Winding1',  
ON SetValue TS.Winding1 =
```

```
BEGIN
```

```
IF TS.Winding1 = 0 THEN  
    MinValue = 0;  
    MaxValue = 0;
```

```
ELSE  
    MinValue = W[TS.Winding1].LowStep;  
    MaxValue = W[TS.Winding1].HighStep;  
ENDIF;
```

```
IF TS.TapStep1 < MinValue OR  
   TS.TapStep1 > MaxValue THEN  
    TS.TapStep1 = (MinValue + MaxValue) / 2;  
ENDIF;
```

```
IF MinValue = MaxValue THEN  
    FIELD(TS.TapStep1).MinValue = MinValue;  
    FIELD(TS.TapStep1).MaxValue = MaxValue + 1;  
    FIELD(TS.TapStep1).UpdateBias = FB_Visible;  
ELSE  
    FIELD(TS.TapStep1).MinValue = MinValue;  
    FIELD(TS.TapStep1).MaxValue = MaxValue;  
    FIELD(TS.TapStep1).UpdateBias = FB_Changeable;  
ENDIF;
```

```
END;
```

```
ON UserEvent 'TS.Winding2',  
ON SetValue TS.Winding2 =
```

```
BEGIN
```

```
IF TS.Winding2 = 0 THEN  
    MinValue = 0;  
    MaxValue = 0;
```

```
ELSE  
    MinValue = W[TS.Winding2].LowStep;  
    MaxValue = W[TS.Winding2].HighStep;  
ENDIF;
```

```
IF TS.TapStep2 < MinValue OR  
   TS.TapStep2 > MaxValue THEN  
    TS.TapStep2 = (MinValue + MaxValue) / 2;  
ENDIF;
```

```
IF MinValue = MaxValue THEN  
    FIELD(TS.TapStep2).MinValue = MinValue;  
    FIELD(TS.TapStep2).MaxValue = MaxValue + 1;  
    FIELD(TS.TapStep2).UpdateBias = FB_Visible;  
ELSE  
    FIELD(TS.TapStep2).MinValue = MinValue;  
    FIELD(TS.TapStep2).MaxValue = MaxValue;  
    FIELD(TS.TapStep2).UpdateBias = FB_Changeable;  
ENDIF;
```

```
END;
```

```
ON UserEvent 'TS.Winding3',  
ON SetValue TS.Winding3 =
```

```
BEGIN
```

2143652

161

```
IF TS.Winding3 = 0 THEN
    MinValue = 0;
    MaxValue = 0;
ELSE
    MinValue = W[TS.Winding3].LowStep;
    MaxValue = W[TS.Winding3].HighStep;
ENDIF;

IF TS.TapStep3 < MinValue OR
   TS.TapStep3 > MaxValue THEN
    TS.TapStep3 = (MinValue + MaxValue) / 2;
ENDIF;

IF MinValue = MaxValue THEN
    FIELD(TS.TapStep3).MinValue = MinValue;
    FIELD(TS.TapStep3).MaxValue = MaxValue + 1;
    FIELD(TS.TapStep3).UpdateBias = FB_Visible;
ELSE
    FIELD(TS.TapStep3).MinValue = MinValue;
    FIELD(TS.TapStep3).MaxValue = MaxValue;
    FIELD(TS.TapStep3).UpdateBias = FB_Changeable;
ENDIF;

END;
```

2143652

162

ON Click Apply_Button =
BEGIN

```
TapSetting.Winding1 = TS.Winding1;  
TapSetting.Winding1Name = FIELD(TS.Winding1).CurEnumText;  
TapSetting.TapStep1 = TS.TapStep1;  
TapSetting.Winding2 = TS.Winding2;  
TapSetting.Winding2Name = FIELD(TS.Winding2).CurEnumText;  
TapSetting.TapStep2 = TS.TapStep2;  
TapSetting.Winding3 = TS.Winding3;  
TapSetting.Winding3Name = FIELD(TS.Winding3).CurEnumText;  
TapSetting.TapStep3 = TS.TapStep3;  
TapSetting.Source = TS.Source;  
TapSetting.R1_1_2 = TS.R1_1_2;  
TapSetting.Z1_1_2 = TS.Z1_1_2;  
TapSetting.R0_1_2 = TS.R0_1_2;  
TapSetting.Z0_1_2 = TS.Z0_1_2;  
TapSetting.R1_2_3 = TS.R1_2_3;  
TapSetting.Z1_2_3 = TS.Z1_2_3;  
TapSetting.R0_2_3 = TS.R0_2_3;  
TapSetting.Z0_2_3 = TS.Z0_2_3;  
TapSetting.R1_3_1 = TS.R1_3_1;  
TapSetting.Z1_3_1 = TS.Z1_3_1;  
TapSetting.R0_3_1 = TS.R0_3_1;  
TapSetting.Z0_3_1 = TS.Z0_3_1;  
TapSetting.R1_1 = TS.R1_1;  
TapSetting.Z1_1 = TS.Z1_1;  
TapSetting.X1_1 = TS.X1_1;  
TapSetting.R0_1 = TS.R0_1;  
TapSetting.Z0_1 = TS.Z0_1;  
TapSetting.X0_1 = TS.X0_1;  
TapSetting.R1_2 = TS.R1_2;  
TapSetting.Z1_2 = TS.Z1_2;  
TapSetting.X1_2 = TS.X1_2;  
TapSetting.R0_2 = TS.R0_2;  
TapSetting.Z0_2 = TS.Z0_2;  
TapSetting.X0_2 = TS.X0_2;  
TapSetting.R1_3 = TS.R1_3;  
TapSetting.Z1_3 = TS.Z1_3;  
TapSetting.X1_3 = TS.X1_3;  
TapSetting.R0_3 = TS.R0_3;  
TapSetting.Z0_3 = TS.Z0_3;  
TapSetting.X0_3 = TS.X0_3;  
RETURN;
```

END;

ON Click Cancel_Button =
BEGIN

RETURN;

END;

2143652

163

```

ON ChildSetValue TS =
  BEGIN
    IF TS.Z1_1_2 IS NULL THEN
      RESUME NEXT;
    ENDIF;
  /*
  **      Check that the windings are set up OK
  */
  IF TS.Winding1 = 0 OR TS.Winding2 = 0 THEN
    INSERT INTO GrassCatcher (Object, Problem)
      VALUES (:TS.Object, 'Transformer_011');
    RESUME NEXT;
  ENDIF;

  IF TS.Winding1 = TS.Winding2 OR
    TS.Winding1 = TS.Winding3 OR
    TS.Winding2 = TS.Winding3 THEN
    INSERT INTO GrassCatcher (Object, Problem)
      VALUES (:TS.Object, 'Transformer_012');
    RESUME NEXT;
  ENDIF;
  /*
  **      Figure out the MVA correction
  */
  i = 1;
  NameplateMVA = 0;
  WHILE i <= W.LastRow() DO
    IF W[i].RatedMVA > NameplateMVA THEN
      NameplateMVA = W[i].RatedMVA;
    ENDIF;
    i = i + 1;
  ENDWHILE;

  IF NameplateMVA = 0 THEN
    INSERT INTO GrassCatcher (Object, Problem)
      VALUES (:TS.Object, 'Transformer_013');
    RESUME NEXT;
  ENDIF;
  /*
  **      Do a sanity check on the measurements
  */
  IF TS.Z1_1_2 < 3.0 OR TS.Z1_1_2 > 15.0 OR
    TS.Z1_2_3 < 3.0 OR TS.Z1_2_3 > 15.0 OR
    TS.Z1_3_1 < 3.0 OR TS.Z1_3_1 > 15.0 THEN
    INSERT INTO GrassCatcher (Object, Problem)
      VALUES (:TS.Object, 'Transformer_014');
  ENDIF;
  /*
  **      Figure out the voltage correction for each winding
  */
  KVCORRECTION = (TS.TapStep1 - W[TS.Winding1].NeutralStep) *
    IFNULL(W[TS.Winding1].StepSize, 0.0);
  NameplateKV = W[TS.Winding1].NeutralKV + KVCORRECTION;
  CF1 = ((NameplateKV / W[TS.Winding1].NominalKV) ** 2) *
    (SP.MVABase / NameplateMVA);

  KVCORRECTION = (TS.TapStep2 - W[TS.Winding2].NeutralStep) *
    IFNULL(W[TS.Winding2].StepSize, 0.0);
  NameplateKV = W[TS.Winding2].NeutralKV + KVCORRECTION;
  CF2 = ((NameplateKV / W[TS.Winding2].NominalKV) ** 2) *
    (SP.MVABase / NameplateMVA);
  /*

```

2143652

164

```

**          Calculate the per-winding impedances etc
*/
IF TS.Winding3 = 0 AND
  TS.Z1_1_2 IS NOT NULL THEN

  TS.Z1_1 = 0.5 * TS.Z1_1_2 * CF1;
  TS.Z1_2 = 0.5 * TS.Z1_1_2 * CF2;
  TS.R1_1 = IFNULL(0.5 * TS.R1_1_2, 0.0) * CF1;
  TS.R1_2 = IFNULL(0.5 * TS.R1_1_2, 0.0) * CF2;

  IF TS.Z0_1_2 IS NOT NULL THEN
    TS.Z0_1 = 0.5 * TS.Z0_1_2 * CF1;
    TS.Z0_2 = 0.5 * TS.Z0_1_2 * CF2;
    TS.R0_1 = IFNULL(0.5 * TS.R0_1_2 * CF1, 0.0);
    TS.R0_2 = IFNULL(0.5 * TS.R0_1_2 * CF2, 0.0);
  ENDIF;

  TS.X1_1 = ((TS.Z1_1 ** 2) - (TS.R1_1 ** 2)) ** 0.5;
  TS.X1_2 = ((TS.Z1_2 ** 2) - (TS.R1_2 ** 2)) ** 0.5;

  TS.X0_1 = ((TS.Z0_1 ** 2) - (TS.R0_1 ** 2)) ** 0.5;
  TS.X0_2 = ((TS.Z0_2 ** 2) - (TS.R0_2 ** 2)) ** 0.5;

ELSEIF TS.Z1_1_2 IS NOT NULL AND
  TS.Z1_3_1 IS NOT NULL AND
  TS.Z1_2_3 IS NOT NULL THEN

  KVCORRECTION = (TS.TapStep3 - W[TS.Winding3].NeutralStep) *
    IFNULL(W[TS.Winding3].StepSize, 0.0);
  NameplateKV = W[TS.Winding3].NeutralKV + KVCORRECTION;
  CF3 = ((NameplateKV / W[TS.Winding3].NominalKV) ** 2) *
    (SP.MVABase / NameplateMVA);

  TS.Z1_1 = 0.5 * (TS.Z1_1_2 + TS.Z1_3_1 - TS.Z1_2_3) * CF1;
  TS.Z1_2 = 0.5 * (TS.Z1_2_3 + TS.Z1_1_2 - TS.Z1_3_1) * CF2;
  TS.Z1_3 = 0.5 * (TS.Z1_3_1 + TS.Z1_2_3 - TS.Z1_1_2) * CF3;
  TS.R1_1 = IFNULL(0.5 * (TS.R1_1_2 + TS.R1_3_1 - TS.R1_2_3) * CF1, 0.0);
  TS.R1_2 = IFNULL(0.5 * (TS.R1_2_3 + TS.R1_1_2 - TS.R1_3_1) * CF2, 0.0);
  TS.R1_3 = IFNULL(0.5 * (TS.R1_3_1 + TS.R1_2_3 - TS.R1_1_2) * CF3, 0.0);

  IF TS.Z0_1_2 IS NOT NULL OR
    TS.Z0_3_1 IS NOT NULL OR
    TS.Z0_2_3 IS NOT NULL THEN
    TS.Z0_1 = 0.5 * (TS.Z0_1_2 + TS.Z0_3_1 - TS.Z0_2_3) * CF1;
    TS.Z0_2 = 0.5 * (TS.Z0_2_3 + TS.Z0_1_2 - TS.Z0_3_1) * CF2;
    TS.Z0_3 = 0.5 * (TS.Z0_3_1 + TS.Z0_2_3 - TS.Z0_1_2) * CF3;
    TS.R0_1 = IFNULL(0.5 * (TS.R0_1_2 + TS.R0_3_1 - TS.R0_2_3) * CF1, 0.0);
    TS.R0_2 = IFNULL(0.5 * (TS.R0_2_3 + TS.R0_1_2 - TS.R0_3_1) * CF2, 0.0);
    TS.R0_3 = IFNULL(0.5 * (TS.R0_3_1 + TS.R0_2_3 - TS.R0_1_2) * CF3, 0.0);
  ENDIF;

  TS.X1_1 = ((TS.Z1_1 ** 2) - (TS.R1_1 ** 2)) ** 0.5;
  TS.X1_2 = ((TS.Z1_2 ** 2) - (TS.R1_2 ** 2)) ** 0.5;
  TS.X1_3 = ((TS.Z1_3 ** 2) - (TS.R1_3 ** 2)) ** 0.5;

  TS.X0_1 = ((TS.Z0_1 ** 2) - (TS.R0_1 ** 2)) ** 0.5;
  TS.X0_2 = ((TS.Z0_2 ** 2) - (TS.R0_2 ** 2)) ** 0.5;
  TS.X0_3 = ((TS.Z0_3 ** 2) - (TS.R0_3 ** 2)) ** 0.5;

/*
**          Give the reactance the sign of the impedance
*/
IF TS.Z1_1 < 0 THEN
  TS.X1_1 = - TS.X1_1;
ENDIF;

IF TS.Z1_2 < 0 THEN

```

2143652

165

```
        TS.X1_2 = - TS.X1_2;  
    ENDIF;  
  
    IF TS.Z1_3 < 0 THEN  
        TS.X1_3 = - TS.X1_3;  
    ENDIF;  
  
    IF TS.Z0_1 < 0 THEN  
        TS.X0_1 = - TS.X0_1;  
    ENDIF;  
  
    IF TS.Z0_2 < 0 THEN  
        TS.X0_2 = - TS.X0_2;  
    ENDIF;  
  
    IF TS.Z0_3 < 0 THEN  
        TS.X0_3 = - TS.X0_3;  
    ENDIF;  
  
ENDIF;  
  
END;
```

2143652

Connectivity Editor

File

ALLE

Transformer

1

Connected to

Terminal	Grouping	Type	Name
1	ALLE-ALLE_41	Jumpor	1
	ALLE-ALLE_51	Jumpor	1
2	ALLE	JTSTAT	TRAN_1

Terminal

Connected to

Grouping	Type	Name

Connect Other to Here

Disconnect

2143652

167

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

```

/*
**      Application:      PowerSystemModel
**      Frame:           ConnectivityEditor
*/
INITIALIZE
(
  T_Row = INTEGER;           /* Row number */
  C_Row = INTEGER;           /* Row number */
  ChoiceItem = INTEGER;      /* Choice item in list */
  FrameName = VARCHAR(20) NOT NULL; /* Editor frame name */
  i = INTEGER NOT NULL;      /* Scratch integer */
  dummy = INTEGER;           /* Scratch integer */
)=
BEGIN
  CurFrame.SendUserEvent (EventName = 'Close');
END;
```

2143652

168

```

ON ChildProperties D1.T[*].C =
  BEGIN
    CALLPROC ObjectEditor (OS = D1.T[[]].C[[]]);
  END;

ON ChildDetails D1.T[*].C =
  BEGIN
    CurFrame.SendUserEvent(EventName = 'LoadObject_D1',
                           MessageInteger = D1.T[[]].C[[]].Object);
  END;

ON UserEvent 'LoadObject_D1' =
  BEGIN
    CurFrame.WindowVisibility = WV_Visible;
/*
*   Get a description of the object
*/
    REPEATED
    SELECT :D1.O.Object = O.Object,
           :D1.O.BaseType = O.BaseType,
           :D1.O.Type = O.Type,
           :D1.O.PrimaryGrouping = O.PrimaryGrouping,
           :D1.O.Name = O.Name,
           :D1.O.TypeName = O.TypeName,
           :D1.O.PrimaryGroupingName = O.PrimaryGroupingName,
           :dummy = :i
    FROM   Object O
    WHERE  :CurFrame.MessageInteger = O.Object;
/*
*   Get the terminals for the object
*/
    D1.T.Clear();
    T_Row = 1;

    REPEATED
    SELECT :D1.T[T_Row].Object = T.Object,
           :D1.T[T_Row].Terminal = T.Terminal,
           :D1.T[T_Row].Node = T.Node,
           :D1.T[T_Row].KVLevel = T.KVLevel,
           :dummy = :i
    FROM   Terminal T
    WHERE  :D1.O.Object = T.Object
    BEGIN
      FIELD(D1.T[T_Row].Terminal).IsReverse = FALSE;
/*
*   Load up all the things connected to this terminal
*/
      D1.T[T_Row].C.Clear();
      C_Row = 1;

      REPEATED
      SELECT :D1.T[T_Row].C[C_Row].Object = O.Object,
             :D1.T[T_Row].C[C_Row].BaseType = O.BaseType,
             :D1.T[T_Row].C[C_Row].Type = O.Type,
             :D1.T[T_Row].C[C_Row].PrimaryGrouping = O.PrimaryGrouping,
             :D1.T[T_Row].C[C_Row].TypeName = O.TypeName,
             :D1.T[T_Row].C[C_Row].Name = O.Name,
             :D1.T[T_Row].C[C_Row].PrimaryGroupingName = O.PrimaryGroupingName,
             :dummy = :i
      FROM   Terminal T,
             Object O
      WHERE  :D1.T[T_Row].Node != :Node$Floating AND
             :D1.T[T_Row].Node = T.Node AND
             :D1.T[T_Row].Object != T.Object AND
             T.Object = O.Object

```

2143652

169

```
BEGIN
    C_Row = C_Row + 1;
END;

T_Row = T_Row + 1;
END;

ROLLBACK;
END;
```

2143652

170

```

ON ChildProperties D2.T[*].C =
  BEGIN
    CALLPROC ObjectEditor (OS = D2.T[[]].C[[]]);
  END;

ON ChildDetails D2.T[*].C =
  BEGIN
    CurFrame.SendUserEvent(EventName = 'LoadObject_D2',
                           MessageInteger = D2.T[[]].C[[]].Object);
  END;

ON UserEvent 'LoadObject_D2' =
  BEGIN
    CurFrame.WindowVisibility = WV_Visible;
  /*
  *   Get a description of the object
  */
    REPEATED
    SELECT :D2.O.Object = O.Object,
           :D2.O.BaseType = O.BaseType,
           :D2.O.Type = O.Type,
           :D2.O.PrimaryGrouping = O.PrimaryGrouping,
           :D2.O.Name = O.Name,
           :D2.O.TypeName = O.TypeName,
           :D2.O.PrimaryGroupingName = O.PrimaryGroupingName,
           :dummy = :i
    FROM   Object O
    WHERE  :CurFrame.MessageInteger = O.Object;
  /*
  *   Get the terminals for the object
  */
    D2.T.Clear();
    T_Row = 1;

    REPEATED
    SELECT :D2.T[T_Row].Object = T.Object,
           :D2.T[T_Row].Terminal = T.Terminal,
           :D2.T[T_Row].Node = T.Node,
           :D2.T[T_Row].KVLevel = T.KVLevel,
           :dummy = :i
    FROM   Terminal T
    WHERE  :D2.O.Object = T.Object
    BEGIN
      FIELD(D2.T[T_Row].Terminal).IsReverse = FALSE;
    /*
    *   Load up all the things connected to this terminal
    */
      D2.T[T_Row].C.Clear();
      C_Row = 1;

      REPEATED
      SELECT :D2.T[T_Row].C[C_Row].Object = O.Object,
             :D2.T[T_Row].C[C_Row].BaseType = O.BaseType,
             :D2.T[T_Row].C[C_Row].Type = O.Type,
             :D2.T[T_Row].C[C_Row].PrimaryGrouping = O.PrimaryGrouping,
             :D2.T[T_Row].C[C_Row].TypeName = O.TypeName,
             :D2.T[T_Row].C[C_Row].Name = O.Name,
             :D2.T[T_Row].C[C_Row].PrimaryGroupingName = O.PrimaryGroupingName,
             :dummy = :i
      FROM   Terminal T,
             Object O
      WHERE  :D2.T[T_Row].Node != :Node$Floating AND
             :D2.T[T_Row].Node = T.Node AND
             :D2.T[T_Row].Object != T.Object AND

```

2143652

```
171
      T.Object = O.Object
BEGIN
  C_Row = C_Row + 1;
END;

  T_Row = T_Row + 1;
END;

ROLLBACK;
END;
```

2143652

172

ON Click D1_Connect =

BEGIN

```

IF FIELD(D1.T).CurrentRow = 0 THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:D1.O.Object, 'Object_008');
    ROLLBACK;
    RESUME;
ENDIF;

```

```
IF FIELD(D2.T).CurrentRow = 0 THEN
    INSERT INTO GrassCatcher (Object, Problem)
        VALUES (:D1.O.Object, 'Object_009');
    ROLLBACK;
    RESUME;
ENDIF;
```

/ ★
 ★
 ★
 ★
 ★
 ★
 ★ /

If both the terminals are already connected to the same node, then assume that the user wishes to break them away from that node. If the user is trying to connect to a disconnected terminal, assume that they mean it. Create a new node, and assign the terminal "to connect to" to the new node.

```

IF D1.T[[]].Node = D2.T[[]].Node OR
   D1.T[[]].Node = Node$Floating THEN
    UPDATE    LastKey
    SET       Value = Value + 1
    WHERE     LastKey = 'Node';

    SELECT    :D1.T[[]].Node = Value
    FROM      LastKey
    WHERE     LastKey = 'Node';

    UPDATE    Terminal
    SET       Node = :D1.T[[]].Node
    WHERE     Object = :D1.T[[]].Object AND
              Terminal = :D1.T[[]].Terminal;
ENDIF;

```

/ ★
 ★
 ★
 ★ /

Connect the terminals by copying the "to connect to" terminal's node to the "to be connected" terminal's node.

```

REPEATED
UPDATE   Terminal
SET      Node = :D1.T[[]].Node
WHERE    Object = :D2.T[[]].Object AND
         Terminal = :D2.T[[]].Terminal;

```

```
COMMIT;
CurFrame.SendUserEvent (EventName = 'LoadObject_D1',
                          MessageInteger = D1.O.Object);
CurFrame.SendUserEvent (EventName = 'LoadObject_D2',
                          MessageInteger = D2.O.Object);
```

END;

2143652

174

ON ChildEntry D1.T =

```
BEGIN
  i = 1;
  WHILE i <= D1.T.LastRow() DO
    FIELD(D1.T[i].Terminal).IsReverse = FALSE;
    i = i + 1;
  ENDWHILE;
  FIELD(D1.T[].Terminal).IsReverse = TRUE;
END;
```

ON ChildEntry D2.T =

```
BEGIN
  i = 1;
  WHILE i <= D2.T.LastRow() DO
    FIELD(D2.T[i].Terminal).IsReverse = FALSE;
    i = i + 1;
  ENDWHILE;
  FIELD(D2.T[].Terminal).IsReverse = TRUE;
END;
```

2112652

175

ON Click D1_Disconnect =

```
BEGIN
    i = FIELD(D1.T).CurRow;

    IF i = 0 THEN
        INSERT INTO GrassCatcher (Object, Problem)
            VALUES (:D1.O.Object, 'Object_010');
        ROLLBACK;
        RESUME;
    ENDIF;

    UPDATE Terminal T
    SET Node = :Node$Floating
    WHERE :D1.T[].Object = T.Object AND
        :D1.T[].Terminal = T.Terminal;

    COMMIT;
    CurFrame.SendUserEvent (EventName = 'LoadObject_D1',
        MessageInteger = D1.O.Object);
    CurFrame.SendUserEvent (EventName = 'LoadObject_D2',
        MessageInteger = D2.O.Object);
END;
```

ON Click D2_Disconnect =

```
BEGIN
    i = FIELD(D2.T).CurRow;

    IF i = 0 THEN
        INSERT INTO GrassCatcher (Object, Problem)
            VALUES (:D2.O.Object, 'Object_010');
        ROLLBACK;
        RESUME;
    ENDIF;

    UPDATE Terminal T
    SET Node = :Node$Floating
    WHERE :D2.T{}.Object = T.Object AND
        :D2.T{}.Terminal = T.Terminal;

    COMMIT;
    CurFrame.SendUserEvent (EventName = 'LoadObject_D1',
        MessageInteger = D1.O.Object);
    CurFrame.SendUserEvent (EventName = 'LoadObject_D2',
        MessageInteger = D2.O.Object);
END;
```

•

2143652

176

/*****

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

*****/

```
/*
**      Application:      PowerSystemModel
**      Frame:           Menu.File.CloseButton
*/
ON UserEvent 'Close',
ON Click Menu.File.CloseButton =

    BEGIN
        CurFrame.WindowVisibility = WV_Invisible;
    END;
```

2143652

File

Find Object

Type Selection

Template

TowerType

Transformer

TransformerBank

☒

☐

☒

Group:

begins with

Name:

contains

COTU

Look Up

5 Found Objects

Group	Type	Name
ALLE	Transformer	COTU
ALLE	Transformer	COTU
LOHA	Transformer	COTU
MACR	Transformer	COTU
SUNB	Transformer	COTU

Apply

Cancel

2143652

178

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

*****/
/*
 *      Application:      PowerSystemModel
 *      Frame:           FindObject
 *
 *      This frame can be called to find an object. The types of objects
 *      that will be presented may be restricted by passing a supertype
 *      to this frame - only the subtypes of this type will be shown for
 *      selection.
 *
 *      The caller frame can also pass its frame identifier, the name of
 *      a user event and (optionally) an integer. If these are passed,
 *      then the selections made by the user are communicated to the caller
 *      by sending user events with the given name and integer and a message
 *      object of type ObjectSummaryClass. If the caller frame does not
 *      pass the frame identifier and event name, this frame returns with
 *      the user selection in an ObjectSummaryClass.
 */
INITIALIZE
(
  SuperType = INTEGER NOT NULL;           /* Passed super type for menu */
  CallerFrame = FrameExec;                /* Passed calling frame */
  CallerEvent = VARCHAR(32);              /* Passed return event to caller */
  CallerInteger = INTEGER;                /* Passed info for caller */
  i = INTEGER NOT NULL;                   /* Scratch integer */
  Type = TypeClass;                       /* Scratch type */
  FO_Title = VARCHAR(32) NOT NULL;        /* Title for FO table field */
  F_GroupingName_Pattern = VARCHAR(22);   /* Pattern for filter name */
  F_ObjectName_Pattern = VARCHAR(22);     /* Pattern for filter name */
) =
BEGIN
/*
 *      Set up a list of all the types for the user to use as a filter
 */
FIELD(F_Type).ValueList.ChoiceItems.Clear();
i = 1;

IF SuperType = 0 THEN
  F_Type = Type$Object;
ELSE
  F_Type = SuperType;
ENDIF;

REPEATED
SELECT  :Type.Type = T.Type,
        :Type.Name = T.Name
FROM    Type T,
        ExtendedType ET

```

2143652

```

                                179
WHERE      :F_Type = ET.SuperType AND
            ET.Type = T.Type AND
            T.BaseType != 0      /* Ignore "for classification only" */
ORDER BY Name
BEGIN
    FIELD(F_Type).ValueList.ChoiceItems[i].EnumText = Type.Name;
    FIELD(F_Type).ValueList.ChoiceItems[i].EnumValue = Type.Type;
    i = i + 1;
END;

ROLLBACK;

/*
 *   Save the original title of the table field
 */
FO_Title = FIELD(FO).Title;
END;
```

2143652

180

```

ON Click LookUp =
/*
*
*/
BEGIN
    FIELD(FO).Title = FO_Title;
    FO.Clear();
    i = 1;
/*
*
*/
    Set up the pattern for the GroupingName match based on the selection.
/*
*
*/
    IF F_GroupWild = 0 THEN
        F_GroupingName_Pattern = F_GroupingName;
    ELSEIF F_GroupWild = 1 THEN
        F_GroupingName_Pattern = '%' + F_GroupingName + '%';
    ELSEIF F_GroupWild = 2 THEN
        F_GroupingName_Pattern = F_GroupingName + '%';
    ELSEIF F_GroupWild = 3 THEN
        F_GroupingName_Pattern = '%' + F_GroupingName;
    ENDIF;
/*
*
*/
    Set up the pattern for the ObjectName match based on the selection.
/*
*
*/
    IF F_ObjectWild = 0 THEN
        F_ObjectName_Pattern = F_ObjectName;
    ELSEIF F_ObjectWild = 1 THEN
        F_ObjectName_Pattern = '%' + F_ObjectName + '%';
    ELSEIF F_ObjectWild = 2 THEN
        F_ObjectName_Pattern = F_ObjectName + '%';
    ELSEIF F_ObjectWild = 3 THEN
        F_ObjectName_Pattern = '%' + F_ObjectName;
    ENDIF;

    REPEATED
    SELECT :FO[i].Object = O.Object,
           :FO[i].BaseType = O.BaseType,
           :FO[i].Type = O.Type,
           :FO[i].PrimaryGrouping = O.PrimaryGrouping,
           :FO[i].TypeName = O.TypeName,
           :FO[i].Name = O.Name,
           :FO[i].PrimaryGroupingName = O.PrimaryGroupingName
    FROM   Object O
    WHERE  O.PrimaryGroupingName LIKE :F_GroupingName_Pattern AND
           O.Name LIKE :F_ObjectName_Pattern AND
           O.Type IN
           (
               SELECT ET.Type
               FROM   ExtendedType ET
               WHERE  :F_Type = ET.SuperType
           )

    ORDER BY
           PrimaryGroupingName,
           TypeName,
           Name

    BEGIN
        i = i + 1;
    END;

    FIELD(FO).Title = VARCHAR(i - 1) + ' ' + FO_Title;
    ROLLBACK;
END;

```

2143652

181

```
ON Click Apply_Button =
/*
 *      Pass the selected object (if any) back to the caller
 */
BEGIN
    IF FIELD(FO).ActiveRow != 0 THEN
        IF CallerEvent = '' THEN
            RETURN FO[];
        ELSE
            CallerFrame.SendUserEvent (EventName = CallerEvent
                                     ,MessageObject = FO[].Duplicate()
                                     ,MessageInteger = CallerInteger
                                     );
        ENDIF;
    ENDIF;
END;

ON Click Cancel_Button =
/*
 *      Done for now
 */
BEGIN
    RETURN NULL;
END;

ON ChildProperties FO =
/*
 *      Edit the selected object
 */
BEGIN
    IF FIELD(FO).ActiveRow != 0 THEN
        CALLPROC ObjectEditor (OS = FO[]);
    ENDIF;
END;
```


2143652

183

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

```

/*
 *      Application:  PowerSystemModel
 *      Frame:       GrassCatcher
 */
INITIALIZE
(
  i = INTEGER NOT NULL;
  SGC = GrassCatcherClass;
) =
BEGIN
  i = 1;

  SELECT :GC[i].Object = GC.Object,
         :GC[i].Problem = GC.Problem,
         :GC[i].Text = GC.Text,
         :GC[i].O.Object = O.Object,
         :GC[i].O.BaseType = O.BaseType,
         :GC[i].O.Type = O.Type,
         :GC[i].O.PrimaryGrouping = O.PrimaryGrouping,
         :GC[i].O.TypeName = O.TypeName,
         :GC[i].O.Name = O.Name,
         :GC[i].O.PrimaryGroupingName = O.PrimaryGroupingName,
         :GC[i].P.Problem = P.Problem,
         :GC[i].P.Severity = P.Severity,
         :GC[i].P.Description = P.Description
  FROM   GrassCatcher GC,
         Object O,
         Problem P
  WHERE  GC.Object = O.Object AND
         GC.Problem = P.Problem
  ORDER BY
         PrimaryGroupingName,
         TypeName,
         Name
  BEGIN
    i = i + 1;
  END;

  ROLLBACK;
END;

```

2143652

184

```

ON DeleteRow GC =
/*
*      Wipe out all matching entries
*/
BEGIN
  IF FIELD(GC).ActiveRow != 0 THEN
    SGC = GC[].Duplicate();

    DELETE FROM GrassCatcher GC
    WHERE   :SGC.Object = GC.Object AND
            :SGC.Problem = GC.Problem;

    i = GC.LastRow();

    WHILE i > 0 DO

      IF GC[i].Object = SGC.Object AND
         GC[i].Problem = SGC.Problem THEN
        GC.RemoveRow (RowNumber = i);
      ENDIF;

      i = i - 1;
    ENDWHILE;

    COMMIT;
  ENDIF;

END;

ON ClearTable GC =
/*
*      Nuke the whole schmeer; throw in a quick MODIFY to reclaim space
*/
BEGIN
  DELETE FROM GrassCatcher;
  MODIFY GrassCatcher TO Btree
    ON Object, Problem;

  GC.Clear();

  COMMIT;
END;

ON ChildProperties GC =
/*
*      Edit the selected object
*/
BEGIN
  IF FIELD(GC).ActiveRow != 0 THEN
    CALLPROC ObjectEditor (OS = GC[].O);
  ENDIF;
END;

```

2143652

Grouping Selection

File

Group hierarchy

Type	Name
Object	Object
Specification	Specification

Group selection

Type	Name
ConductorType	Dove
ConductorType	Flicker
ConductorType	Grosbeak
ConductorType	Hawk
ConductorType	Hen
ConductorType	Ibis
ConductorType	Linnet
ConductorType	Merlin
ConductorType	Osprey
ConductorType	Parakeet

Members

Network

2143652

186

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

```

/*
**      Application:  PowerSystemModel
**      Frame:       GroupingSelection
*/
INITIALIZE
(
  i = INTEGER;                               /* Scratch integer */
  MenuEntry = ChoiceItem;                     /* Scratch menu entry */
  FrameName = VARCHAR(20) NOT NULL;           /* Editor frame name */
  ObjectSummary = ObjectSummaryClass;         /* Object summary */
) =
BEGIN
/*
*      Set up the top level "grouping" and load up its members
*/
  GH.Clear();

  GH[1].Object = 0;
  GH[1].BaseType = 0;
  GH[1].Type = 0;
  GH[1].PrimaryGrouping = 0;
  GH[1].TypeName = 'Object';
  GH[1].Name = 'Object';
  GH[1].PrimaryGroupingName = 'Object';

  ObjectSummary = GH[1];
  CurFrame.SendUserEvent (EventName = 'GSLoadData');
END;

ON UserEvent 'GSLoadData' =
  BEGIN
/*
*      Get the members of the selected group object
*/
    GS.Clear ();
    i = 1;

    REPEATED
    SELECT :GS[i].Object = O.Object,
           :GS[i].BaseType = O.BaseType,
           :GS[i].Type = O.Type,
           :GS[i].PrimaryGrouping = O.PrimaryGrouping,
           :GS[i].TypeName = O.TypeName,
           :GS[i].Name = O.Name,
           :GS[i].PrimaryGroupingName = O.PrimaryGroupingName
    FROM   Object O,

```

2143552

187

```
      Grouping G
WHERE   :ObjectSummary.Object = G.Grouping AND
        :Type$Member_of = G.Relationship AND
        G.Member = O.Object
ORDER BY
        TypeName,
        Name
BEGIN
    i = i + 1;
END;

ROLLBACK;
END;
```

2143652

188

```

ON ChildEntry GH =
    BEGIN
        IF FIELD(GH).CurRow != 0 THEN
            ObjectSummary = GH[];
/*
*       Remove the entries lower down the hierarchy
*/
            i = GH.LastRow();

            WHILE i > FIELD(GH).CurRow DO
                GH.RemoveRow(RowNumber = i);
                i = i - 1;
            ENDWHILE;

            CurFrame.SendUserEvent (EventName = 'GSLoadData');
        ENDIF;
    END;

ON ChildDetails GS,
ON Click Members =
    BEGIN
        IF FIELD(GS).ActiveRow != 0 THEN
            ObjectSummary = GS[];
/*
*       Add the selected row to the end of the heirarchy
*/
            i = GH.LastRow();
            GH[i + 1] = GS[].Duplicate();
            CurFrame.SendUserEvent (EventName = 'GSLoadData');
        ENDIF;
    END;

ON ChildProperties GS =
    BEGIN
        IF FIELD(GS).CurRow != 0 THEN
            CALLPROC ObjectEditor (OS = GS[]);
        ENDIF;
    END;

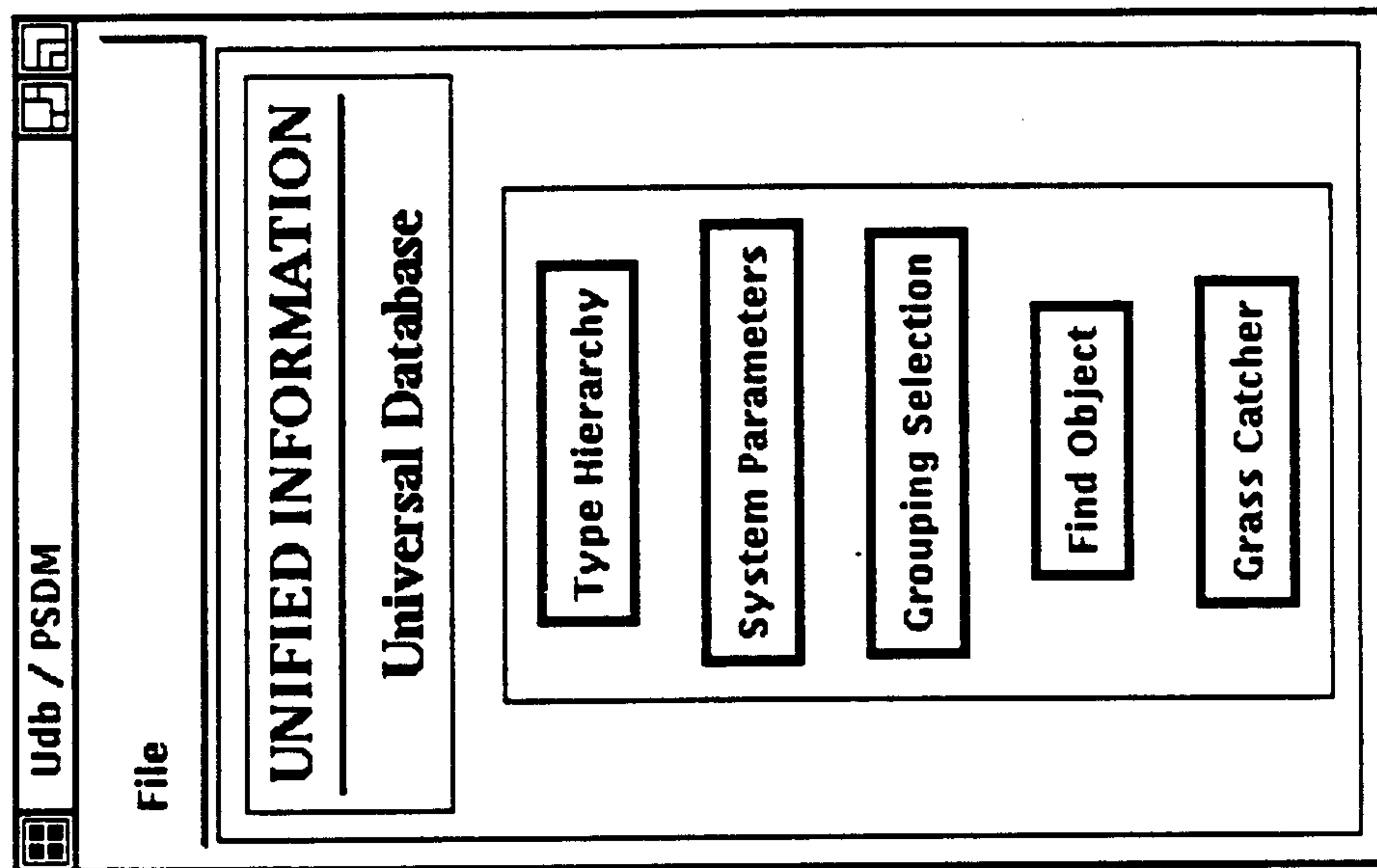
ON Click Network =
    BEGIN
        IF FIELD(GS).CurRow != 0 THEN
            GlobalFrame[Frame$ConnectivityEditor].Frame.SendUserEvent
                (EventName = 'LoadObject_D1', MessageInteger = GS[].Object);
        ENDIF;
    END;

ON Details Network =
    BEGIN
        IF FIELD(GS).CurRow != 0 THEN
            GlobalFrame[Frame$ConnectivityEditor].Frame.SendUserEvent
                (EventName = 'LoadObject_D2', MessageInteger = GS[].Object);
        ENDIF;
    END;

```

2143652

189



2143652

190

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

```

/*
**      Application:  PowerSystemModel
**      Frame:       PSM
*/
INITIALIZE
(
)=
BEGIN
/*
**      Set up the global variables from the database
*/
SELECT      :SP.Frequency = SP.Frequency,
             :SP.MVABase = SP.MVABase,
             :SP.KVReference = SP.KVReference,
             :SP.LengthRatio = SP.LengthRatio,
             :SP.GroundResistivity = SP.GroundResistivity,
             :SP.Temperature = SP.Temperature
FROM        SystemParameters SP;

CurFrame.SendUserEvent (EventName = 'StartGlobalFrames');
ROLLBACK;
END;

ON UserEvent 'StartGlobalFrames' =
BEGIN
/*
**      Start up all the global frames that the editors can use. Do this
**      in subscript order so that the array extends correctly.
*/
GlobalFrame[Frame$ConnectivityEditor].Frame = OPENFRAME ConnectivityEditor;
END;

ON ChildClick B =
BEGIN
CALLFRAME :CurFrame.TriggerField.Name;
END;

```

2143652

Set Grouping

File Grouping Load Rating Location

230

Line

ALBU-ALBR_T1

Groups of which this object is a member

Primary	Relationship	Type	Name	Reference
☒	Member of	Division	230	
☐	Load modeled by	Line	ALBU-ALBR_T1	
☐				
☐				

Apply

Cancel

2143652

192

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
**      Application:  PowerSystemModel
**      Frame:       SetGrouping
*/
INITIALIZE
(
  Object = ObjectClass;                      /* Object being edited */
  OG = ARRAY OF GroupingClass;                /* Passed groupings */
  Relationship = INTEGER NOT NULL;             /* Relationship being added */
  i = INTEGER NOT NULL;                       /* Scratch integer */
  CList = ChoiceList;                         /* Scratch choice list entry */
  Type = TypeClass;                           /* Scratch type */
  OS = ObjectSummaryClass;                    /* Scratch Object summary */
) =
BEGIN
/*
*      Show the object being worked on
*/
  LO.Object = Object.Object;
  LO.BaseType = Object.BaseType;
  LO.Type = Object.Type;
  LO.PrimaryGrouping = Object.PrimaryGrouping;
  LO.TypeName = Object.TypeName;
  LO.Name = Object.Name;
  LO.PrimaryGroupingName = Object.PrimaryGroupingName;
/*
*      Load up the relationship (interaction) list
*/
  CALLPROC LoadTypeList
    (EF = EnumField(FIELD(MG[*].Relationship).ProtoField)
    , StartType = Type$Interaction
    );
/*
*      Get the names for the grouping objects and set the
*      primary grouping marker
*/
  MG = OG.Duplicate();
  i = 1;
  WHILE i <= MG.LastRow() DO
    REPEATED
    SELECT :MG[i].GO.Object = GO.Object,
           :MG[i].GO.BaseType = GO.BaseType,
           :MG[i].GO.Type = GO.Type,
           :MG[i].GO.PrimaryGrouping = GO.PrimaryGrouping,
           :MG[i].GO.TypeName = GO.TypeName,
           :MG[i].GO.Name = GO.Name,
           :MG[i].GO.PrimaryGroupingName = GO.PrimaryGroupingName

```

2143652

193

```
FROM      Object GO
WHERE     :MG[i].Grouping = GO.Object;

IF MG[i].Grouping = Object.PrimaryGrouping THEN
    MG[i].Primary = 1;
ELSE
    MG[i].Primary = 0;
ENDIF;

FIELD(MG[i].Primary).HasDataChanged = FALSE;
i = i + 1;
ENDWHILE;

ROLLBACK;
END;
```

2143652

194

```

On Click Menu.Grouping.Member_of =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$Object
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Member_of
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Grouping.Made_by =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$Manufacturer
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Made_by
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Grouping.Shown_on =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$Drawing
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Shown_on
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Load.Load_modeled_by =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$Consumer
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Load_modeled_by
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Load.Load_family =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$LoadFamily
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Load_modeled_by
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Load.Self =
  BEGIN
    CurFrame.SendUserEvent (EventName = 'AddSelection'
      ,MessageObject = LO
      ,MessageInteger = Type$Load_modeled_by
      );
    END;

On Click Menu.Rating.Rated_as =
  BEGIN

```

2143652

195

```

OPENFRAME FindObject
  (SuperType = LO.BaseType
  ,CallerFrame = CurFrame
  ,CallerEvent = 'AddSelection'
  ,CallerInteger = Type$Rated_as
  ) WITH
  WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
END;

On Click Menu.Rating.Rating_family =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$RatingFamily
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Rated_as
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Rating.Self =
  BEGIN
    CurFrame.SendUserEvent (EventName = 'AddSelection'
                           ,MessageObject = LO
                           ,MessageInteger = Type$Rated_as
                           );
    END;

On Click Menu.Location.Located_with =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$Object
      ,CallerFrame = CurFrame
      ,CallerEvent = 'AddSelection'
      ,CallerInteger = Type$Located_with
      ) WITH
      WindowTitle = MenuButton(CurFrame.OriginatorField).TextLabel;
    END;

On Click Menu.Location.Self =
  BEGIN
    CurFrame.SendUserEvent (EventName = 'AddSelection'
                           ,MessageObject = LO
                           ,MessageInteger = Type$Located_with
                           );
    END;

```

2143652

196

```
ON UserEvent 'AddSelection' =
/*
 *      Add the selection to the members' group list
 */
BEGIN
  OS = ObjectSummaryClass(CurFrame.MessageObject);
  Relationship = CurFrame.MessageInteger;
  i = 1;

  WHILE i <= MG.LastRow() DO
    IF MG[i].Grouping = OS.Object AND
       MG[i].Relationship = Relationship THEN
      INSERT INTO GrassCatcher (Object, Problem)
        VALUES (OS.Object, 'Object_011');
      ROLLBACK;
      RESUME;
    ENDIF;

    i = i + 1;
  ENDWHILE;

  MG[i].Primary = 0;
  FIELD(MG[i].Primary).HasDataChanged = FALSE;
  MG[i].GO = OS.Duplicate();
  MG[i].Grouping = OS.Object;
  MG[i].GroupingType = OS.Type;
  MG[i].Relationship = Relationship;
  MG[i].Reference = '';
END;
```

2143652

197

```

ON ChildSetValue MG[*].Primary =
/*
*   If this is being set, ensure it is the only one; set the active
*   row to the one being selected.
*/
BEGIN
    i = 1;
    WHILE i <= MG.LastRow() DO
        IF FIELD(MG[i].Primary).HasDataChanged = TRUE THEN
            FIELD(MG).ActiveRow = i;
        ELSE
            MG[i].Primary = 0;
        ENDIF;

        FIELD(MG[i].Primary).HasDataChanged = FALSE;
        i = i + 1;
    ENDWHILE;

END;

ON ChildProperties MG =
/*
*   Edit the selected object
*/
BEGIN
    IF FIELD(MG).ActiveRow != 0 THEN
        CALLPROC ObjectEditor (OS = MG[.GO]);
    ENDIF;
END;

ON ChildDetails MG =
/*
*   Show the details of the association
*/
BEGIN
    IF FIELD(MG).ActiveRow != 0 THEN
        IF MG[.Relationship] = Type$Member_of THEN
            /* No details to show */
        ELSEIF MG[.Relationship] = Type$Made_by THEN
            /* No details to show */
        ELSEIF MG[.Relationship] = Type$Shown_on THEN
            /* No details to show */
        ELSEIF MG[.Relationship] = Type$Load_modeled_by THEN
            CALLFRAME SetLoadCurve (Object = MG[.Grouping]);
        ELSEIF MG[.Relationship] = Type$Rated_as THEN
            CALLFRAME SetRating (Object = MG[.Grouping]);
        ELSEIF MG[.Relationship] = Type$Located_with THEN
            CALLFRAME SetLocation (Object = MG[.Grouping]);
        ENDIF;
    ENDIF;
END;

```

2143652

198

```
ON Click Apply_Button =
/*
 *   Ensure that the objects' primary group is set correctly and
 *   pass back the modified list
 */
BEGIN
  Object.PrimaryGrouping = 0;
  Object.PrimaryGroupingName = '';
  i = 1;
  OG.Clear();

  WHILE i <= MG.LastRow() DO
    IF MG[i].Primary = 1 THEN
      Object.PrimaryGrouping = MG[i].Grouping;
      Object.PrimaryGroupingName = MG[i].GO.Name;
    ENDIF;

    OG[i] = MG[i].Duplicate();
    i = i + 1;
  ENDWHILE;

  RETURN;
END;

ON Click Cancel_Button =
BEGIN
  RETURN;
END;
```


2143652

200

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

```

/*
 *      Application:  PowerSystemModel
 *      Frame:       SetKVLevel
 */
INITIALIZE
(
  kVLevel = FLOAT NOT NULL;          /* Passed kVLevel */
  Voltage = ARRAY OF FloatObject;    /* Defined voltages */
  i = INTEGER NOT NULL;              /* Scratch integer */
)=
BEGIN
/*
 *      Set up a list of all the kVLevels
 */
FIELD(F_kVLevel).ValueList.ChoiceItems.Clear();
i = 1;

SELECT  :Voltage[i].Value = kV.kVLevel
FROM    kVLevel kV
ORDER BY Value
BEGIN
  FIELD(F_kVLevel).ValueList.ChoiceItems[i].EnumText =
    VARCHAR (Voltage[i].Value);
  FIELD(F_kVLevel).ValueList.ChoiceItems[i].EnumValue = i;

  IF Voltage[i].Value = kVLevel THEN
    F_kVLevel = i;
  ENDIF;

  i = i + 1;
END;

ROLLBACK;
END;

```

2143652

201

```
ON Click Apply_Button =
/*
 *      Pass the selected voltage level back to the caller
 */
BEGIN
    RETURN Voltage[F_kVLevel].Value;
END;

ON Click Cancel_Button =
/*
 *      Pass the original voltage level back to the caller
 */
BEGIN
    RETURN kVLevel;
END;
```

2143652

Load model

File

Load curves

Season	Day type	Temperature	Interpolate
Template	Template	48.0	Linear

Load values

Time	Load value
09:00	0.890
09:15	0.910
09:30	0.950
09:45	0.960
10:00	0.950
10:15	0.930

2143652

203

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

*****
/*
 *      Application:  PowerSystemModel
 *      Frame:       SetLoadCurve
 */
INITIALIZE
(
  Object = INTEGER NOT NULL;          /* Passed object */
  i = INTEGER NOT NULL;              /* Scratch integer */
  Season = ObjectSummaryClass;        /* Scratch season */
  DayType = ObjectSummaryClass;       /* Scratch day type */
  CList = ChoiceList;                /* Reference to choice list */
) =
BEGIN
/*
 *      Set up the Season list
 */
CList = EnumField(FIELD(LC[*].Season).ProtoField).ValueList;
CList.ChoiceItems.Clear();
CList.ChoiceItems[1].EnumValue = 0;
CList.ChoiceItems[1].EnumText = '';
i = 2;

REPEATED
SELECT  :Season.Object = S.Object,
        :Season.Name = S.Name
FROM    Object S
WHERE   :Type$Season = S.Type
ORDER BY Name
BEGIN
  CList.ChoiceItems[i].EnumValue = Season.Object;
  CList.ChoiceItems[i].EnumText = Season.Name;
  i = i + 1;
END;

EnumField(FIELD(LC[*].Season).ProtoField).UpdChoiceList();

/*
 *      Set up the DayType list
 */
CList = EnumField(FIELD(LC[*].DayType).ProtoField).ValueList;
CList.ChoiceItems.Clear();
CList.ChoiceItems[1].EnumValue = 0;
CList.ChoiceItems[1].EnumText = '';
i = 2;

REPEATED
SELECT  :DayType.Object = DT.Object,
        :DayType.Name = DT.Name

```

2143652

204

```

FROM      Object DT
WHERE      :Type$DayType = DT.Type
ORDER BY Name
BEGIN
    CList.ChoiceItems[i].EnumValue = DayType.Object;
    CList.ChoiceItems[i].EnumText = DayType.Name;
    i = i + 1;
END;

EnumField(FIELD(LC[*].DayType).ProtoField).UpdChoiceList();
/*
 * Load up the load curves from the passed object
 */
i = 1;

REPEATED
SELECT :LC[i].Object = LC.Object,
       :LC[i].DayType = LC.DayType,
       :LC[i].Season = LC.Season,
       :LC[i].Temperature = LC.Temperature,
       :LC[i].LoadCurve = LC.LoadCurve,
       :LC[i].Source = LC.Source,
       :LC[i].Interpolate = LC.Interpolate
FROM    LoadCurve LC
WHERE   :Object = LC.Object
BEGIN
    i = i + 1;
END;

ROLLBACK;
END;

ON ChildEntry LC =
/*
 * Display the list of values for the load curve
 */
BEGIN
    i = 1;
    REPEATED
    SELECT :LV[i].LoadCurve = LV.LoadCurve,
           :LV[i].Time = LV.Time,
           :LV[i].LoadValue = LV.LoadValue
    FROM    LoadValue LV
    WHERE   :LC[].LoadCurve = LV.LoadCurve
    BEGIN
        i = i + 1;
    END;

    ROLLBACK;
END;

ON ChildProperties LV =
/*
 * Draw a graph of the load values
 */
BEGIN
    CALLFRAME LoadCurveGraph (LV = LV);
END;

```

2143652

205

```

INITIALIZE
(
    LV = ARRAY OF LoadValueClass;

    Xmax = INTEGER NOT NULL;
    Ymax = INTEGER NOT NULL;
    Xseg = FLOAT NOT NULL;
    Yseg = FLOAT NOT NULL;
    NextSegment = SegmentShape;

    LastX = INTEGER NOT NULL;
    LastY = INTEGER NOT NULL;
    NewX = INTEGER NOT NULL;
    NewY = INTEGER NOT NULL;

    Time = INTEGER NOT NULL;
    i = INTEGER NOT NULL;
) =
BEGIN
    Xmax = FIELD(load_subform).Width;
    Ymax = FIELD(load_subform).Height;

    Xseg = Xmax / 1440.0;          /* Minutes per day */
    Yseg = Ymax / 2.0;           /* Load value range */

    /* calculate the data points and line segments */

    i=1;

    WHILE i <= LV.LastRow() DO
        Time = DATE_PART ('hours', LV[i].Time) * 60;
        Time = Time + DATE_PART ('minutes', LV[i].Time);
        NewX = Xseg * Time;
        NewY = Ymax - (Yseg * LV[i].LoadValue);

        IF i > 1 THEN
            NextSegment = SegmentShape.Create();
            NextSegment.SetEndpoints(
                Point1X = lastx,
                Point1Y = lasty,
                Point2X = newx,
                Point2Y = newy);
            NextSegment.LineWidth = LW_VERYTHIN;
            NextSegment.LineColor = CC_LIGHT_BLUE;
            NextSegment.ParentFIELD = FIELD(load_subform);
        ENDIF;

        LastX = NewX;
        LastY = NewY;
        i = i + 1;
    ENDWHILE;

END;

```

2143652

Set Location

Co-located Objects

Group	Type	Name

Contact: Joseph P. Blow

Address: 1234 Ridgemoor Drive

Comment: Heavily wooded area, low lighting (carry your flashlight).

Phone: 672-8640

Combination: 696

Anywhere, USA 98012

Apply

Cancel

2143652

207

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

*****
/*
 *      Application:  PowerSystemModel
 *      Frame:       SetLocation
 */
INITIALIZE
(
  Object = INTEGER;                                /* Passed object */
  i = INTEGER NOT NULL;                             /* Scratch integer */
) =
BEGIN
/*
 *      Build the list of co-located objects
 */
CO.Clear();
i = 1;

REPEATED
SELECT :CO[i].Object = CO.Object,
       :CO[i].BaseType = CO.BaseType,
       :CO[i].Type = CO.Type,
       :CO[i].PrimaryGrouping = CO.PrimaryGrouping,
       :CO[i].Name = CO.Name,
       :CO[i].TypeName = CO.TypeName,
       :CO[i].PrimaryGroupingName = CO.PrimaryGroupingName
FROM   Object CO,
       Grouping G
WHERE  :Object = G.Grouping AND
       :Type$Located_with = G.Relationship AND
       G.Member = CO.Object

BEGIN
  i = i + 1;
END;

/*
 *      Set up the location description. Ensure that if there isn't one
 *      in the database, then the object reference is set to null and the
 *      last used text is left so that it can be edited.
 */
L.Object = Object$Null;

REPEATED
SELECT :L.Object = L.Object,
       :L.Combination = L.Combination,
       :L.Phone = L.Phone,
       :L.Contact = L.Contact,
       :L.L1 = L.L1,
       :L.L2 = L.L2,
       :L.L3 = L.L3,

```

2143652

208

```
      :L.L4 = L.L4,  
      :L.Comment = L.Comment  
FROM   Location L  
WHERE  :Object = L.Object;  
  
ROLLBACK;  
END;
```

2143652

209

```

ON Click Apply_Button =
/*
*      Set up the location, if needed; then update it
*/
BEGIN
    IF L.Object = Object$Null THEN
        INSERT INTO Location
            (Object
            )
        VALUES
            (:Object
            );
    ENDIF;

    UPDATE Location L
    SET
        Combination = :L.Combination,
        Phone = :L.Phone,
        Contact = :L.Contact,
        L1 = :L.L1,
        L2 = :L.L2,
        L3 = :L.L3,
        L4 = :L.L4,
        Comment = :L.Comment
    WHERE :Object = L.Object;

    COMMIT;
    RETURN;
END;

ON Click Cancel_Button =
BEGIN
    RETURN;
END;

ON ChildProperties CO =
/*
*      Edit the selected object
*/
BEGIN
    IF FIELD(CO).CurRow != 0 THEN
        CALLPROC ObjectEditor (OS = CO[]);
    ENDIF;

    END;

```

2143652

Measurement

SUBU

CB

PROV_1

Measurand type	Measurand type	Measurand type	Measurand type
MW	Telemetered	SUBU-1	12.1
MVAR	Telemetered	SUBU-1	12.0
MVA	Calculated		
Status	StateEstimated		

Measurand type: MW

Terminal 1: 1

Terminal 2: 1

Measurand type: Telemetered

Measurand type: SUBU-1

Measurand type: Analog

Measurand type: 12.1

Sensor type: Power

Minimum: -200.000

Maximum: 200.000

Reverse polarity

Accuracy: 5.000

Alarm

History

Trigger

Event log

Input

Output

Apply

Cancel

2143652

211

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
**      Application:  PowerSystemModel
**      Frame:       SetMeasurment
*/
INITIALIZE
(
  Object = ObjectClass;                      /* Object being edited */
  OM = ARRAY OF MeasurementClass;             /* Passed measurements */
  i = INTEGER NOT NULL;                       /* Scratch integer */
  Terminal = INTEGER NOT NULL;                /* Scratch terminal number */
  CList = ChoiceList;                         /* Scratch choice list entry */
) =
BEGIN
  CurFrame.CurMode = FM_Query;
/*
*      Show the object being worked on
*/
  O.Object = Object.Object;
  O.BaseType = Object.BaseType;
  O.Type = Object.Type;
  O.PrimaryGrouping = Object.PrimaryGrouping;
  O.TypeName = Object.TypeName;
  C.Name = Object.Name;
  O.PrimaryGroupingName = Object.PrimaryGroupingName;
/*
*      Load up the type lists
*/
  CALLPROC LoadTypeList
    (EF = EnumField(FIELD(ML[*].MeasurandType).ProtoField)
    , StartType = Type$Measurand
    );
  FIELD(M.MeasurandType).ValueList =
    EnumField(FIELD(ML[*].MeasurandType).ProtoField).ValueList;

  CALLPROC LoadTypeList
    (EF = EnumField(FIELD(ML[*].MeasurerType).ProtoField)
    , StartType = Type$Measurer
    );
  FIELD(M.MeasurerType).ValueList =
    EnumField(FIELD(ML[*].MeasurerType).ProtoField).ValueList;

  CALLPROC LoadTypeList
    (EF = EnumField(FIELD(M.SensorType))
    , StartType = Type$Sensor
    );
/*
*      Load the lists of valid terminals for the object

```

2143652

212

```

/*
CList = FIELD(M.Terminal1).ValueList;
CList.ChoiceItems.Clear();
CList.ChoiceItems[1].EnumValue = 0;
CList.ChoiceItems[1].EnumText = '';
i = 2;

REPEATED
SELECT DISTINCT
    :Terminal = T.Terminal
FROM    Terminal T
WHERE   :Object.Object = T.Object
ORDER BY Terminal
BEGIN
    CList.ChoiceItems[i].EnumValue = Terminal;
    CList.ChoiceItems[i].EnumText = VARCHAR(Terminal);
    i = i + 1;
END;

FIELD(M.Terminal1).UpdChoiceList();
FIELD(M.Terminal2).ValueList = CList;
FIELD(M.Terminal2).UpdChoiceList();
/*
*   Build the list of measurements made on this object.
*/
ML = OM.Duplicate();
i = 1;
WHILE i <= ML.LastRow() DO
    REPEATED
    SELECT :ML[i].MO.Object = MO.Object,
           :ML[i].MO.BaseType = MO.BaseType,
           :ML[i].MO.Type = MO.Type,
           :ML[i].MO.PrimaryGrouping = MO.PrimaryGrouping,
           :ML[i].MO.TypeName = MO.TypeName,
           :ML[i].MO.Name = MO.Name,
           :ML[i].MO.PrimaryGroupingName = MO.PrimaryGroupingName
    FROM    Object MO
    WHERE   :ML[i].Measurer = MO.Object;
    i = i + 1;
ENDWHILE;

ROLLBACK;
END;

ON ChildEntry ML =
/*
*   Edit a measurement
*/
BEGIN
    M = ML[];
    CurFrame.CurMode = FM_Update;
END;

ON ChildProperties ML =
/*
*   Edit the selected object
*/
BEGIN
    IF FIELD(ML).ActiveRow != 0 THEN
        IF ML[].Measurer != 0 THEN
            CALLPROC ObjectEditor (OS = ML[].MO);
        ENDIF;
    ENDIF;
END;

```

2143652

213

```

ON ChildProperties M.MO =
  BEGIN
    IF M.Measurer != 0 THEN
      CALLPROC ObjectEditor (OS = M.MO);
    ENDIF;
  END;

ON ChildDetails M.MO =
  BEGIN
    OPENFRAME FindObject
      (SuperType = Type$Sensing
      ,CallerFrame = CurFrame
      ,CallerEvent = 'UpdateM.MO'
      ,CallerInteger = 0
      ) WITH
      WindowTitle = 'Find Measurer';
  END;

ON UserEvent 'UpdateM.MO' =
  BEGIN
    M.MO = ObjectSummaryClass(CurFrame.MessageObject).Duplicate();
    M.Measurer = M.MO.Object;
    FIELD(ML).UpdField();
  END;

ON ChildSetValue M =
  BEGIN
    FIELD(ML).UpdField();
  END;

ON Click Apply_Button =
/*
*   Pass back the modified list.
*/
  BEGIN
    i = 1;
    OM.Clear();

    WHILE i <= ML.LastRow() DO
      OM[i] = ML[i].Duplicate();
      i = i + 1;
    ENDWHILE;

    RETURN;
  END;

ON Click Cancel_Button =
  BEGIN
    RETURN;
  END;

```

2143652

214

Set Object

Object selection

- ConductorType Dove
- ConductorType Flicker
- ConductorType Grosbeak
- ConductorType Hawk

Apply Cancel

2143652

215

Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.

```

/*
 *      Application:  PowerSystemModel
 *      Frame:       SetObject
 */
INITIALIZE
(
    Type = INTEGER NOT NULL;           /* Passed Type */
    DefaultObject = INTEGER NOT NULL;  /* Passed default object */
    i = INTEGER NOT NULL;              /* Scratch integer */
    s = VARCHAR(80) NOT NULL;          /* Scratch string */
    OS = ObjectSummaryClass;           /* Scratch Object */
) =
BEGIN
/*
 *      Set up a list of all the Objects for the user to use as a filter
 */
    FIELD(F_Object).ValueList.ChoiceItems.Clear();
    i = 1;

    REPEATED
    SELECT :OS.Object = O.Object,
           :OS.PrimaryGroupingName = O.PrimaryGroupingName,
           :OS.TypeName = O.TypeName,
           :OS.Name = O.Name
    FROM   Object O,
           ExtendedType ET
    WHERE  :Type = ET.SuperType AND
           ET.Type = O.Type
    ORDER BY PrimaryGroupingName, TypeName, Name
    BEGIN
        s = OS.PrimaryGroupingName + ' ' + OS.TypeName + ' ' + OS.Name;
        s = Squeeze (s);
        FIELD(F_Object).ValueList.ChoiceItems[i].EnumText = s;
        FIELD(F_Object).ValueList.ChoiceItems[i].EnumValue = OS.Object;
        i = i + 1;
    END;

    F_Object = DefaultObject;
    ROLLBACK;
END;

```

2143652

216

```
ON Click Apply_Button =
/*
 *      Pass the selected Object back to the caller
 */
BEGIN
    RETURN F_Object;
END;

ON Click Cancel_Button =
BEGIN
    RETURN NULL;
END;

ON Properties F_Object =
/*
 *      Edit the selected object
 */
BEGIN
    IF F_Object != Object$Null THEN
        REPEATED
        SELECT  :OS.Object = O.Object,
                :OS.BaseType = O.BaseType,
                :OS.Type = O.Type,
                :OS.PrimaryGrouping = O.PrimaryGrouping,
                :OS.TypeName = O.TypeName,
                :OS.Name = O.Name,
                :OS.PrimaryGroupingName = O.PrimaryGroupingName
        FROM    Object O
        WHERE   :F_Object = O.Object;

        CALLPROC ObjectEditor (OS = OS);
    ENDIF;
END;
```

2143652

Set Rating

File

Co-rated objects

Group	Type	Name

Type	Season	Temperature	Normal	Short term	Emergency
MVA	Template	48.000	1.000	1.200	1.300
		0.000	0.000	0.000	0.000

2143652

218

```

/*****
Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.
*****/

/*
 *      Application:  PowerSystemModel
 *      Frame:       SetRating
 */
INITIALIZE
(
    Object = INTEGER;                               /* Passed object */
    i = INTEGER NOT NULL;                           /* Scratch integer */
    Season = ObjectSummaryClass;                    /* Scratch season */
    SeasonList = ChoiceList;                        /* Reference to season list */
) =
BEGIN
/*
 *      Set up the Season list
 */
SeasonList = EnumField(FIELD(R[*].Season).ProtoField).ValueList;
SeasonList.ChoiceItems.Clear();
SeasonList.ChoiceItems[1].EnumValue = 0;
SeasonList.ChoiceItems[1].EnumText = '';
i = 2;

SELECT  :Season.Object = S.Object,
        :Season.Name = S.Name
FROM    Object S
WHERE   :Type$Season = S.Type
ORDER BY Name
BEGIN
    SeasonList.ChoiceItems[i].EnumValue = Season.Object;
    SeasonList.ChoiceItems[i].EnumText = Season.Name;
    i = i + 1;
END;

EnumField(FIELD(R[*].Season).ProtoField).UpdChoiceList();

/*
 *      Build the list of co-rated objects
 */
CO.Clear();
i = 1;

REPEATED
SELECT  :CO[i].Object = CO.Object,
        :CO[i].BaseType = CO.BaseType,
        :CO[i].Type = CO.Type,
        :CO[i].PrimaryGrouping = CO.PrimaryGrouping,
        :CO[i].Name = CO.Name,
        :CO[i].TypeName = CO.TypeName,
        :CO[i].PrimaryGroupingName = CO.PrimaryGroupingName

```

2143652

219

```
FROM      Object CO,
          Grouping G
WHERE      :Object = G.Grouping AND
          :Type$Rated_as = G.Relationship AND
          G.Member = CO.Object
BEGIN
    i = i + 1;
END;

/*
*
* Load up the ratings
*/

R.Clear();
i = 1;

REPEATED
SELECT     :R[i].Object = R.Object,
          :R[i].Type = R.Type,
          :R[i].Season = R.Season,
          :R[i].Temperature = R.Temperature,
          :R[i].Normal = R.Normal,
          :R[i].ShortTerm = R.ShortTerm,
          :R[i].Emergency = R.Emergency,
          :R[i].Loadshed = R.Loadshed
FROM       Rating R
WHERE      :Object = R.Object
ORDER BY  Type, Temperature
BEGIN
    i = i + 1;
END;

ROLLBACK;
END;
```

2143652

220

```

ON Click Apply_Button =
/*
 *      Blow the ratings away, and re-load them
 */
BEGIN
    DELETE FROM Rating WHERE :Object = Object;

    i = 1;
    WHILE i <= R.LastRow() DO
        INSERT INTO Rating
            (Object
            ,Type
            ,Season
            ,Temperature
            ,Normal
            ,ShortTerm
            ,Emergency
            ,Loadshed
            )
        VALUES
            (:Object
            ,:R[i].Type
            ,:R[i].Season
            ,:R[i].Temperature
            ,:R[i].Normal
            ,:R[i].ShortTerm
            ,:R[i].Emergency
            ,:R[i].Loadshed
            );
        i = i + 1;
    ENDWHILE;

    COMMIT;
    RETURN;
END;

ON Click Cancel_Button =
BEGIN
    RETURN;
END;

ON ChildProperties CO =
/*
 *      Edit the selected object
 */
BEGIN
    IF FIELD(CO).CurRow != 0 THEN
        CALLPROC ObjectEditor (OS = CO[]);
    ENDIF;

END;

```

2143652

221

The image shows a graphical user interface window titled "Set Type". The window has a standard Mac OS-style title bar with a close button, a zoom button, and a scroll button. Below the title bar is a vertical scroll bar. The main content area of the window is titled "Type Selection" and contains a list of four options: "Organization", "Spares", "Substation", and "System". Each option is preceded by a small square checkbox. The "Organization" checkbox is checked. Above the list is a small rectangular input field. To the right of the list are two buttons: "Apply" and "Cancel".

Type Selection
☒ Organization
☐ Spares
☐ Substation
☐ System

Apply Cancel

2143652

222

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

```

*****
/*
 *      Application:  PowerSystemModel
 *      Frame:       FindType
 */
INITIALIZE
(
  Object = ObjectClass;          /* Passed object */
  i = INTEGER NOT NULL;          /* Scratch integer */
  Type = TypeClass;              /* Scratch type */
)=
BEGIN
/*
 *      Set up a list of all the types for the user to use as a filter
 */
FIELD(F_Type).ValueList.ChoiceItems.Clear();
i = 1;
F_Type = Object.Type;

REPEATED
SELECT  :Type.Type = T.Type,
        :Type.Name = T.Name
FROM    Type T,
        ExtendedType ET
WHERE   :Object.BaseType = ET.SuperType AND
        ET.Type = T.Type
ORDER BY Name
BEGIN
  FIELD(F_Type).ValueList.ChoiceItems[i].EnumText = Type.Name;
  FIELD(F_Type).ValueList.ChoiceItems[i].EnumValue = Type.Type;
  i = i + 1;
END;

ROLLBACK;
END;

```

2143652

223

```
ON Click Apply_Button =
/*
 *      Pass the selected type back to the caller
 */
BEGIN
    Object.Type = F_Type;
    Object.TypeName = FIELD(F_Type).CurEnumText;
    RETURN;
END;

ON Click Cancel_Button =
BEGIN
    RETURN;
END;
```

2143652

System Parameters	
File	
Frequency:	60.00
MVA base:	100.00
Reference kV:	1.00
Long / short length ratio:	1000.00
Ground resistivity:	100.00
Temperature:	C ☐
Voltage levels	⊗
kV level	Δ
138.00	
230.00	
345.00	
500.00	

2143652

225

```

/*****
Copyright 1991, 1992 by Unified Information, Inc.
All Rights Reserved. Unpublished rights reserved under the
copyright laws of the United States.

The software contained on this media is proprietary to and
embodies the confidential technology of Unified Information, Inc.
Possession, use, duplication or dissemination of the software
and media is authorized only pursuant to a valid written license
from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
U.S. Government is subject to restrictions as set forth in
Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
52.227-19, as applicable.
*****/

/*
**      Application:      PowerSystemModel
**      Frame:           SystemParameters
*/
INITIALIZE
(
    i = INTEGER NOT NULL;                /* Scratch integer */
    OriginalKV = ARRAY OF kVLevelClass;  /* Original values in table */
) =

BEGIN
/*
*      Show the global copy of the system parameters
*/
    SPF = SP;
    CurFrame.SendUserEvent(EventName = 'LoadkVLevel');
END;

ON UserEvent 'LoadkVLevel' =
    BEGIN
/*
*      Get the kV levels and make a copy of the original values
*/
        kV.Clear();
        i = 1;

        REPEATED
        SELECT :kV[i].kVLevel = kV.kVLevel
        FROM   kVLevel kV
        ORDER BY kVLevel
        BEGIN
            OriginalkV[i] = kV[i].Duplicate();
            i = i + 1;
        END;

        ROLLBACK;
    END;

ON ClearTable kV =
    BEGIN
        RESUME;
    END;

ON DeleteRow kV=
/*
*      Attempt to delete the kVLevel entry, then re-load the table
*      field so the user can see the current state of affairs.

```

2143652

226

```

*/
BEGIN
    i = FIELD(kV).CurRow;

    DELETE FROM kVLevel kV
    WHERE :OriginalkV[i].kVLevel = kV.kVLevel;

    COMMIT;
    CurFrame.SendUserEvent(EventName = 'LoadkVLevel');
END;

ON InsertRow kV=
/*
*   Add a new row in the same position in the "original" array and
*   set the value to null.
*/
BEGIN
    i = FIELD(kV).CurRow;
    OriginalkV.InsertRow (RowNumber = i);
    OriginalkV[i].kVLevel = NULL;
END;

ON ChildSetValue kV =
/*
*   Update the database immediately so that the user can
*   get any feedback, then re-load the table
*   field so they can see the current state of affairs.
*/
BEGIN
    i = FIELD(kV).CurRow;

    IF OriginalkV[i].kVLevel IS NULL THEN
        INSERT INTO kVLevel
            (kVLevel
            ,Name
            ,Voltage
            )
        VALUES
            (:kV[i].kVLevel           /***** fix up *****/
            ,VARCHAR(:kV[i].kVLevel)  /***** fix up *****/
            ,:kV[i].kVLevel           /***** fix up *****/
            );
    ELSE
        UPDATE kVLevel kV
        SET
            kVLevel = :kV[i].kVLevel, /***** fix up *****/
            Name = VARCHAR(:kV[i].kVLevel), /***** fix up *****/
            Voltage = :kV[i].kVLevel /***** fix up *****/
        WHERE :OriginalkV[i].kVLevel = kV.kVLevel;
    ENDIF;

    COMMIT;
    CurFrame.SendUserEvent(EventName = 'LoadkVLevel');
END;

```

2143652

Type

File

Type Hierarchy

Name

Object

Organization

Type Selection

Name

Bay

CapacitorBank

Circuit

Company

Division

Line

Subtypes

2143652

228

/*****

Copyright 1991, 1992 by Unified Information, Inc.
 All Rights Reserved. Unpublished rights reserved under the
 copyright laws of the United States.

The software contained on this media is proprietary to and
 embodies the confidential technology of Unified Information, Inc.
 Possession, use, duplication or dissemination of the software
 and media is authorized only pursuant to a valid written license
 from Unified Information, Inc.

RESTRICTED RIGHTS LEGEND Use, duplication, or disclosure by the
 U.S. Government is subject to restrictions as set forth in
 Subparagraph (c) (1) (ii) of DFARS 252.227-7013, or in FAR
 52.227-19, as applicable.

*****/

```

/*
**      Application:  PowerSystemModel
**      Frame:       Type
*/
INITIALIZE
(
  i = INTEGER;                      /* Scratch integer */
  Row = INTEGER;                    /* Scratch integer */
  ErrorNumber = INTEGER;            /* Scratch integer */
  MenuEntry = ChoiceItem;          /* Scratch menu entry */
  TypeName = VARCHAR(20);          /* Local string variable */
)=
BEGIN
/*
*      Set up the Type Hierarchy table field.
*/

```

```

TH.Clear();
SELECT  :TH[1].Type = T.Type,
        :TH[1].BaseType = T.BaseType,
        :TH[1].Name = T.Name
FROM    Type T
WHERE   T.Type = :Type$Object;

```

```

/*
*      Set up the Type Selection table field.
*/

```

```

i = 1;
TS.Clear();
SELECT  :TS[i].Type = T.Type,
        :TS[i].BaseType = T.BaseType,
        :TS[i].Name = T.Name
FROM    Type T,
        SuperType ST
WHERE   :TH[1].Type = ST.SuperType AND
        ST.Type = T.Type
ORDER BY Name
BEGIN
  i = i + 1;
END;

```

```

ROLLBACK;
END;

```

2143652

229

```

ON ChildSetValue TS =
  BEGIN
/*
*   Insert a new type and it's supertype if the key is undefined,
*   otherwise update the existing entry.
*/
    IF TS[].Type = 0 THEN
      INSERT INTO Type
        (Type
        , BaseType
        , Name
        )
      VALUES
        (-1
        , :TH[TH.LastRow()].BaseType
        , :TS[].Name
        );
      INQUIRE_SQL( :ErrorNumber = ERRORNO );

      IF ErrorNumber != 0 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
          VALUES (0, :TS[].Name, 'Type_007');
        ROLLBACK;
        RESUME;
      ENDIF;

      SELECT  :TS[].Type = K.Value
      FROM    LastKey K
      WHERE   K.LastKey = 'Type';

      INSERT INTO SuperType
        (SuperType
        , Type
        )
      VALUES
        (:TH[TH.LastRow()].Type
        , :TS[].Type
        );
    ELSE
      UPDATE  Type T
      SET     Name = :TS[].Name
      WHERE   :TS[].Type = T.Type;
      INQUIRE_SQL( :ErrorNumber = ERRORNO );

      IF ErrorNumber != 0 THEN
        INSERT INTO GrassCatcher (Object, Text, Problem)
          VALUES (0, :TS[].Name, 'Type_007');
        ROLLBACK;
        RESUME;
      ENDIF;
    ENDIF;

  COMMIT;
END;

```

2143652

230

```
ON DeleteRow TS =
/*
 * Delete the SuperType which points to this Type. Then delete the Type
 * itself.
 */
BEGIN
    DELETE FROM SuperType ST
    WHERE   :TS[].Type = ST.Type;

    INQUIRE_SQL( :ErrorNumber = ERRORNO );

    IF ErrorNumber = 0 THEN                                /* Query succeeded */
        DELETE FROM Type T
        WHERE   :TS[].Type = T.Type;

        INQUIRE_SQL( :ErrorNumber = ERRORNO );

        IF ErrorNumber = 0 THEN
            TS.RemoveRow( RowNumber = FIELD(TS).CurRow );
        ELSE
            ROLLBACK;
        ENDIF;
    ENDIF;

    COMMIT;
END;

ON ClearTable TS =
BEGIN
    RESUME;
END;
```

2143652

231

```

ON ChildEntry TH =
/*
 * Load the Type Selection table field with the subclasses of the selected
 * type in the Type Hierarchy table field.
 *
 * Never allow the TH table field to get input focus.
 */
BEGIN
    Row = FIELD(TH).CurRow;

    i = 1;
    TS.Clear();
    REPEATED
    SELECT :TS[i].Type = T.Type,
           :TS[i].BaseType = T.BaseType,
           :TS[i].Name = T.Name
    FROM   Type T,
           SuperType ST
    WHERE  :TH[Row].Type = ST.SuperType AND
           ST.Type = T.Type
    ORDER BY Name
    BEGIN
        i = i + 1;
    END;
/*
 * Remove everything below the current row starting at the end.
 */
Row = TH.LastRow();

WHILE Row > FIELD(TH).CurRow DO
    TH.RemoveRow(RowNumber = Row);
    Row = Row - 1;
ENDWHILE;

ROLLBACK;
RESUME;
END;
/* Throw away the rest of the events. */

```

2143652

232

```

ON ChildDetails TS,
ON Click SubClasses_Btn =
/*
* Load the Type Hierarchy table field with the superclass of the selected
* type in the Type Selection table field. Then reload the Type Selection
* table field with the subclasses of that Type Hierarchy table field entry.
*/
BEGIN
    IF FIELD(TS).CurRow = 0 THEN          /* If nothing is selected go away */
        ROLLBACK;
        RESUME;
    ENDIF;

    Row = TH.LastRow() + 1;
    TH[Row] = TS[].Duplicate(); /* Copy the TS entry to TH table field. */

    i = 1;
    TS.Clear();

    REPEATED
    SELECT :TS[i].Type = T.Type,
           :TS[i].BaseType = T.BaseType,
           :TS[i].Name = T.Name
    FROM   Type T,
           SuperType ST
    WHERE  :TH[Row].Type = ST.SuperType AND
           ST.Type = T.Type
    ORDER BY Name
    BEGIN
        i = i + 1;
    END;

    ROLLBACK;
    RESUME;                                /* Throw away the rest of the events. */
END;

```

-233-

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A computer-implemented method for modelling a physical system using an object-oriented information model in which elements of the physical system are modelled by objects, the information model being implemented using a relational database, the method comprising:

(a) storing in the relational database an object table, each entry in the object table containing an object identifier, an object type value, and a plurality of object attributes;

(b) storing in the relational database a plurality of type-specific tables, one for each of a number of defined object types, each entry in said type-specific tables containing an object identifier and a plurality of further object attributes;

(c) storing in the relational database a supertype table defining a hierarchy of object types; and

(d) selectively accessing information stored in the relational database to perform operations involving said objects.

2. A method according to Claim 1, further including storing in the relational database a grouping table, each entry in said grouping table defining a group-member relationship between objects.

3. A method according to Claim 1, further including assigning a unique surrogate key to each object, by which that object is identified in said object table and type-specific tables, and searching the relational database for a particular surrogate key using a balanced binary tree index employing sequential keys.

-234-

4. A method according to Claim 1, further including providing a graphic user interface based on a window environment for creating and editing said objects.

5. A method according to Claim 1, further including providing a second computer that stores in a proprietary database element data associated with a portion of the physical system, and transferring the element data between the second computer and the first-mentioned computer such that a single point of maintenance is provided at the first-mentioned computer.

6. A method according to Claim 1, wherein the physical system is an electric utility power system.

**Smart & Biggar
Ottawa, Canada
Patent Agents**

2/5

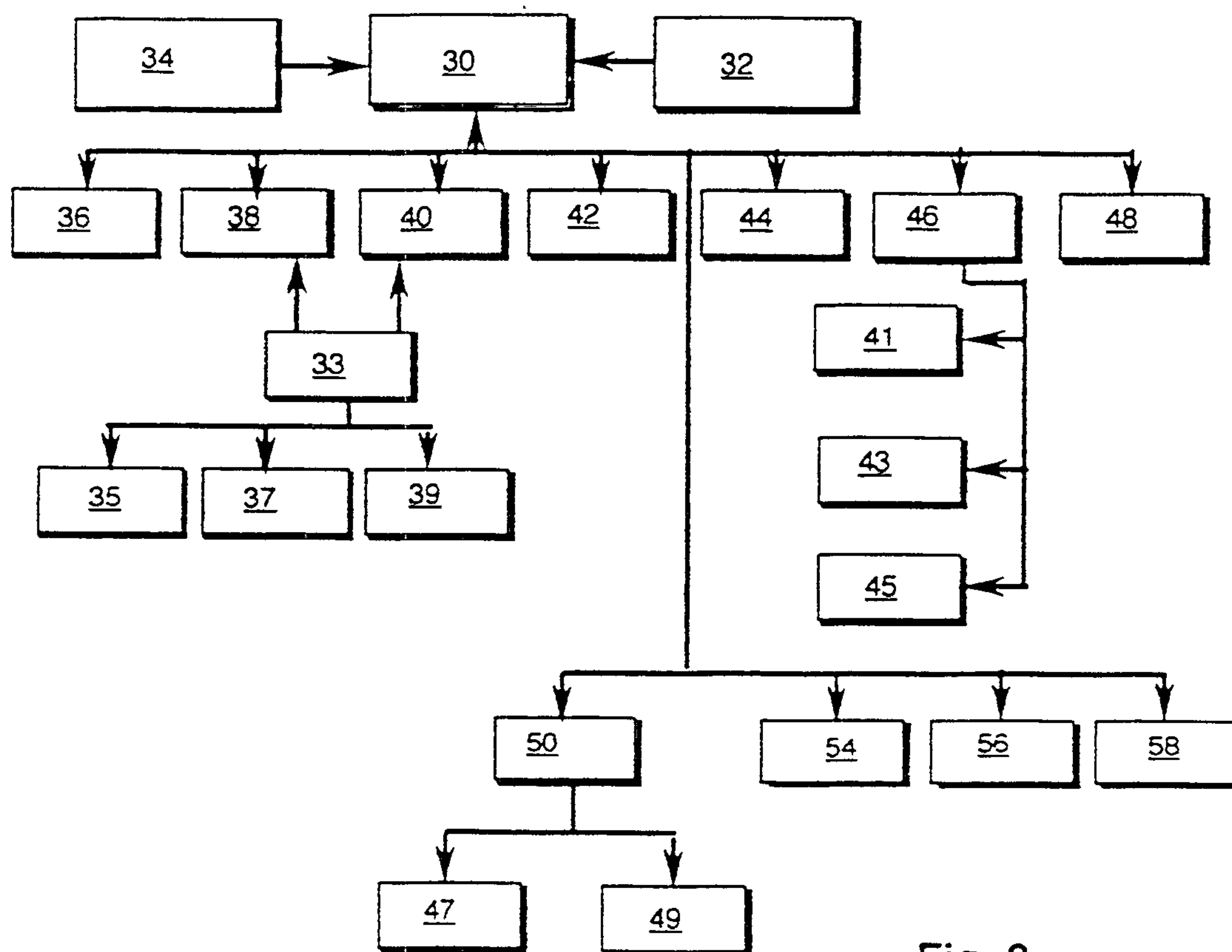
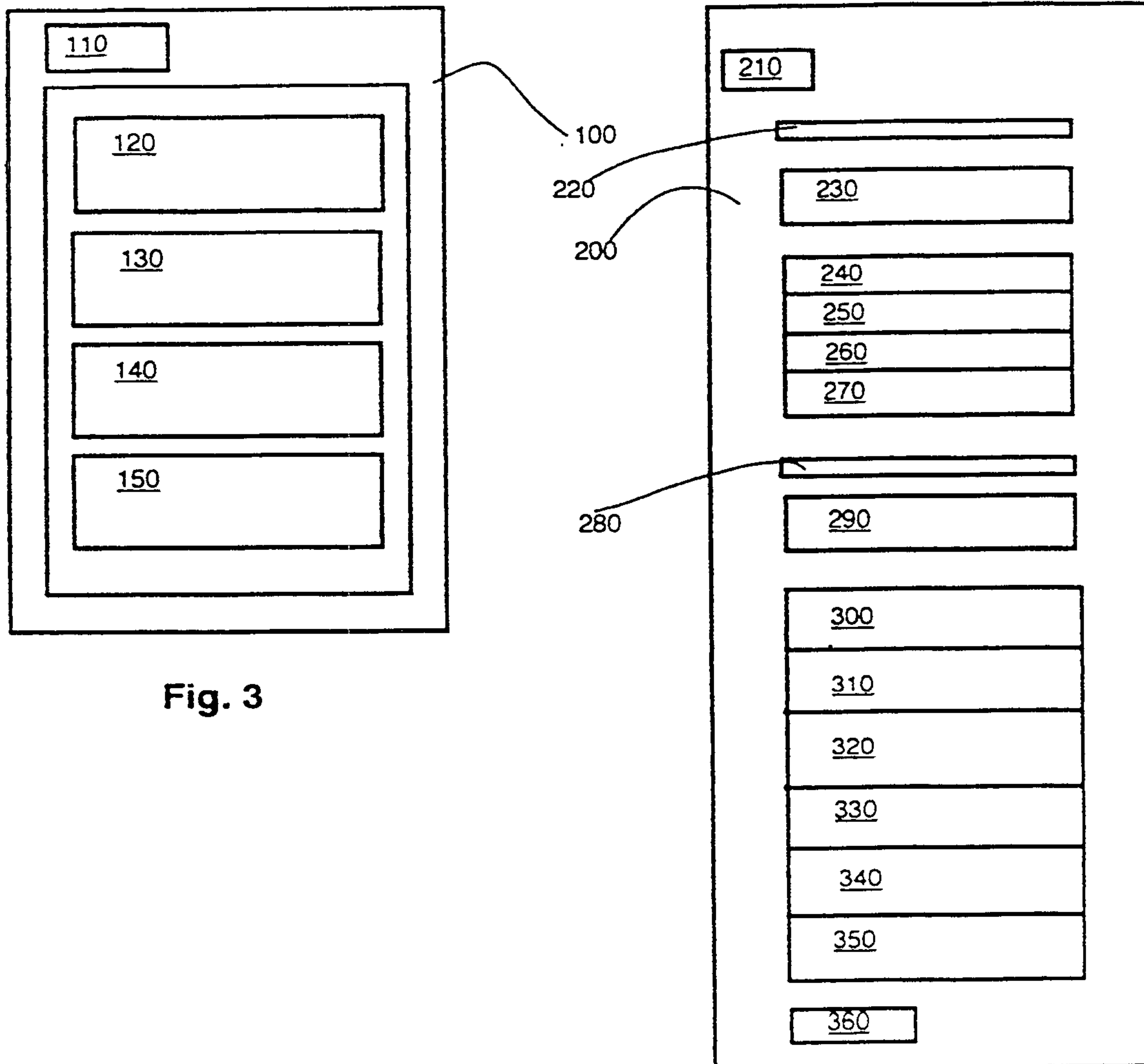


Fig. 2

2143652

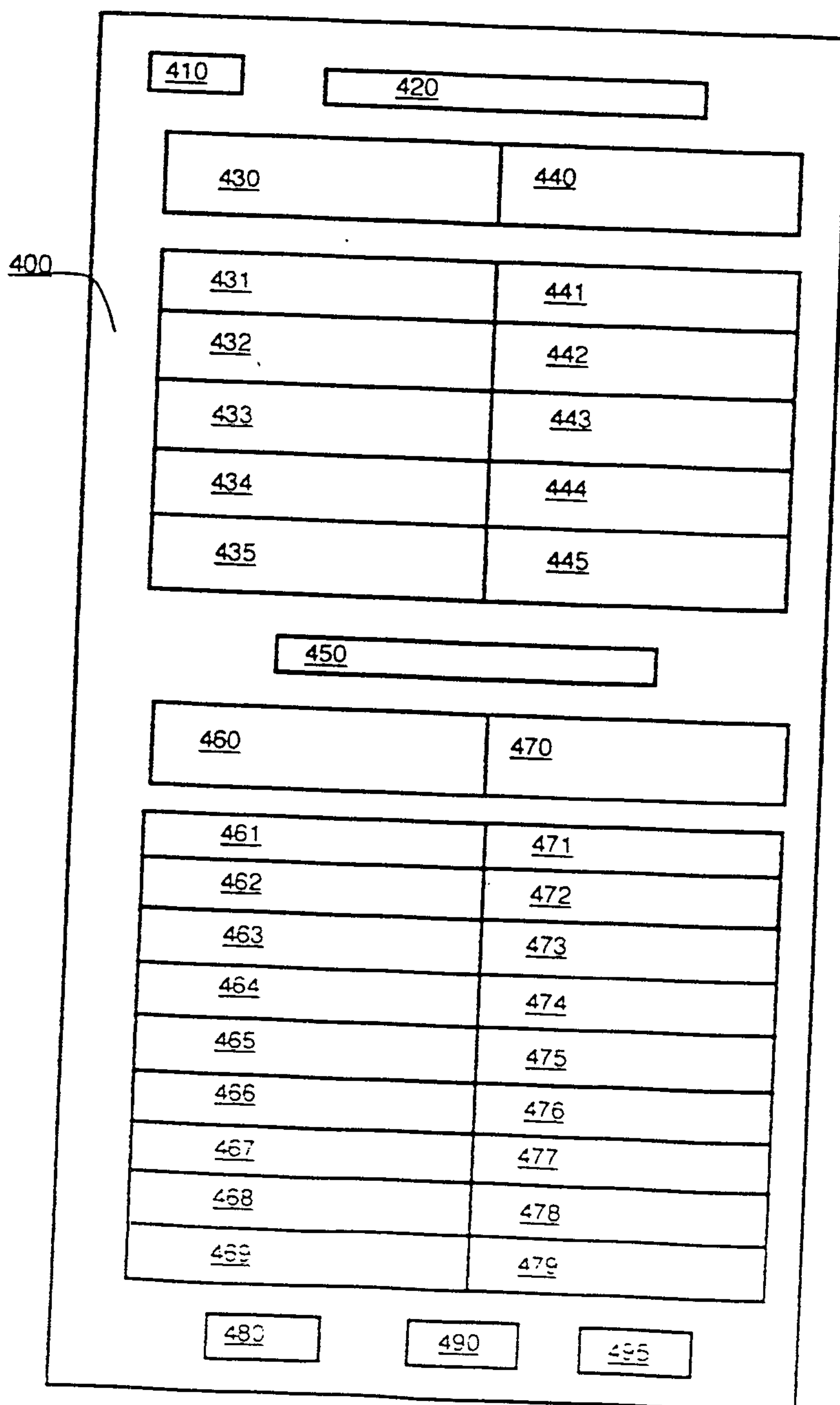
3/5



2143652

4/5

Fig. 5



2143652

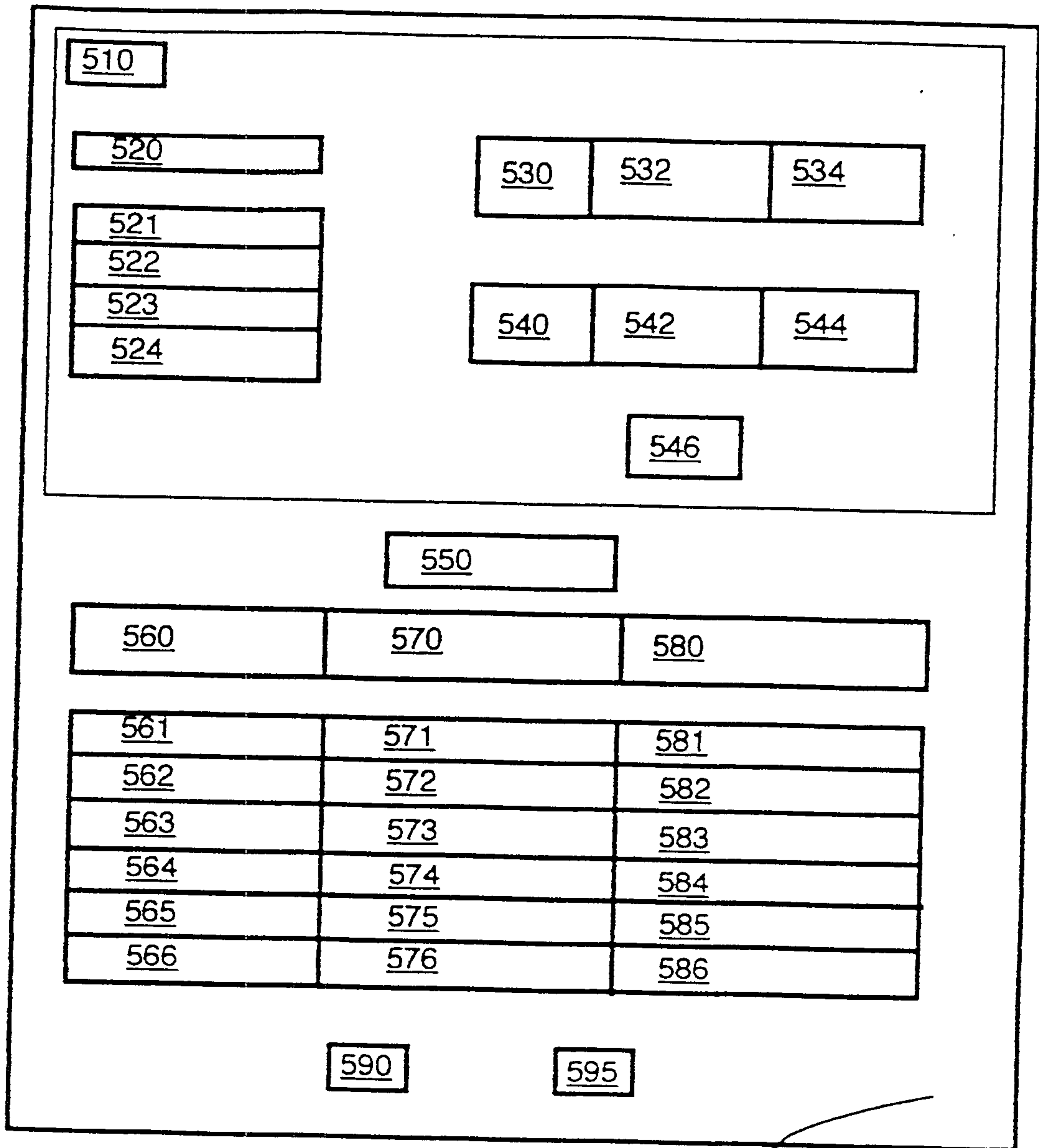


Fig. 6

500

