[N:N] maximum path cardinality pair. Beginning at a step **980** the system determines whether the maximum path cardinality of the object link attribute under consideration is less than or equal to one. If so, the system adds a foreign key from the referenced parent table to the object link's parent table at a step **982**. A determination is made at a step **983** whether the maximum cardinality of the object link is less than or equal to one. If not, then the system creates a new relational table at a step **984**. A foreign key is imported into the new relational table from both the parent table and the referenced semantic object table at a step **986**. If the answer to step **983** was yes, the system must determine whether the object link is contained in a multivalued group at a step **983***a*. If not, the system then determines whether the relationship specified by the object link is many to many at step **996**. A step **988** determines whether the ID status property of the object link is unique. If so, the foreign key added from the referenced semantic object table is made a candidate key at a step **990**. If the ID status of the object link is not unique, the system determines whether the ID status is equal to "none" at a step **992**. If the ID status is equal to none, a step **994** adds a surrogate key to a new column of the relational table created at step **984**. In all cases, the system then checks whether the maximum path cardinality for the object link is [N:N] and (1) whether the containers of the object links are both a semantic object or (2) both a group, at a step **996**. If either of these is true, an intersection table is created, and a foreign key from the parent semantic object table and a foreign key from the referenced semantic object table are added to the intersection table at a step **998**. If at step **996** it is determined that the container of one object link is contained in a group and the other is contained in a semantic object or vice versa, no intersection table is created.

As can be seen by FIG. **26**, no relational tables or columns are required to process a subtype attribute. A subtype attribute is always paired with a corresponding parent-type attribute; therefore, the present invention lets the parent-type attributes define a relation between the tables created for the parent relational table and the subtype relational table.

FIG. **27** shows the steps taken by the present invention to transform a parent-type attribute into one or more relational tables. Beginning at a step **1010**, the system finds the semantic object referenced by the parent-type attribute. At a step **1012**, the system determines whether a table exists for the referenced semantic object. If no table exists, then a table is created for the referenced semantic object at a step **1014**. At a step **1016**, a foreign key from the referenced semantic object table is placed in the parent semantic object table. Finally, at a step **1018**, the foreign key is marked as a candidate key.

FIG. **28** shows the steps taken by the present invention to transform a subtype group attribute. Beginning at a step **1028**, the system determines if the minimum count of the subtype group is greater than zero. The system then determines at a step **1030** whether the maximum count for the group is less than the number of member attributes contained in the group. If the answers to steps **1028** and **1030** are no, nothing is done. However, if the answer is yes in at least one step, a step **1032** determines whether the maximum cardinality of the group is less than or equal to one. If the maximum cardinality of the group is greater than one, a new relational table is created at a step **1034**, and a foreign key from the parent table is added to the new relational table at a step **1036**. Finally, a column is added to the table and a binary data type is defined that is used to indicate the attributes within the group that are selected.

FIG. **29** shows the steps taken by the present invention in order to define the primary keys for a table and update any keys that may have been ambiguous when the relational tables were created. The system begins at a step **1050**, where for each table created, the primary keys for the table are defined according. At a step **1054**, the system determines whether there are any candidate keys identified. If there are candidate keys, the user selects the primary key to be either a surrogate key or a data key in a step **1056**. In a step **1058**, the system determines whether the user has selected a data key. If a data key has been selected, the user is then prompted to select a key from the list of possible key candidates at a step **1057**. If the user selects a surrogate key, a new column is added to the table and the surrogate key is inserted at a step **1060**. In a step **1061**, all columns in the relational table that have been selected as a key are marked as primary keys. Any remaining candidate keys not selected as primary keys are then defined as unique indices at a step **1062**. Once all primary keys have been defined for all tables, any foreign keys that were dependent upon a candidate key being selected are updated at a step **1064** to reflect which of the candidate keys was selected as a primary key.

As described above, the present invention does not create the tables in the database program. The present invention either places function calls with the appropriate arguments to routines provided by the database program to create the relational tables or creates a text file with the appropriate commands that are read by the database program to create the tables.

For example, if the present invention is to create tables for the Access® database program produced by the Microsoft Corporation, the present invention calls predefined routines to create the tables. To create a relational table in Access® the following function calls are:

```
Start table ("Table Name")
        add field ("Colname", data type)
        •
        •
        •
        create primary key ("Colname, Colmane . . .")
        create index ("Colname, Colname . . .", unique)
End table
Open table
        create foreign key ("reference table name, "Colname,
        Colname . . .", table name)
Close table
```

This code creates a table, adds numerous columns of a particular data type, sets the primary key and creates an index. Additionally, the routine "create foreign key" defines a foreign key for the table, after the referenced tables are defined. As will be appreciated, given the table definitions defined in the figures described above, it is well within the skill of an ordinary programmer to determine what commands must be called or commands created to make a database program create the corresponding relational tables.

While the preferred embodiment of the invention has been illustrated and described, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A programmed computer system for creating a relational database schema from a semantic object model, comprising:

a central processing unit;

a memory containing a sequence of program steps that when executed by the central processing unit: