

**DYNAMIC OBJECT PROPERTIES****BACKGROUND**

The present invention relates to business intelligence tools for building applications on a database management system (DBMS).

The advent of powerful, yet economical computers made possible by advances in processor, memory and data storage devices has made computers an integral part of modern companies. An important class of application for these computers includes a DBMS where information is collected and organized according to a data model and searched using queries. The DBMS allows users to perform operations such as locating, adding, deleting and updating records stored in the computer without a detailed knowledge of how the information making up the records actually is stored in the computer.

One powerful type of DBMS is known as a relational DBMS where stored information appears to the user as a set of tables, each of which is termed a relation. In each relation, the information appears to be arranged in rows and columns, with columns of data being related to each other by one or more predetermined functions.

To access particular information in the relational DBMS, a query compiler converts a user request, typically expressed in a query language such as a Structured Query Language (SQL), into a set of operations to be performed on one or more input relations to yield a solution responsive to the user's request. Using the query language provided by the DBMS, the user may develop application programs which facilitate retrieval of the data from the DBMS, processing of the data, and organization of the data into reports.

One issue in developing business intelligence tools is the type of reports that the tool is to generate. Typically, the tool generates certain pre-formatted reports using the query language. Although the query language is easier to use than conventional programming languages such as Basic or C, the generation of each new report still requires a certain programming expertise and can often take a substantial amount of time.

**SUMMARY**

The invention supports objects with dynamic properties which may be edited using a property entry sheet that supports dynamic properties. In one aspect, a computer-implemented property entry sheet for contextually assigning a property of an object associated with an application includes an attribute name section adapted to receive an identification of the property; and a property input section adapted to receive a functional expression for the property identified by the attribute name section, the functional expression being referenceable at run-time as a data value.

Implementations of the property entry sheet may include one or more of the following. The functional expression may be a function, an operator, a database column name, a variable, and/or a constant. The property entry sheet may also include an attribute name section adapted to receive an identification of the property and a property input section adapted to receive a static data value for the property identified by the attribute name section. Further, the object has a plurality of properties and wherein the attribute name section and the property input section of each property form a name-value pair for each property. The functional expression may be parsed to generate a function which is stored as

a run-time value and byte code associated with the function may be generated. The function may also be cloned and stored as a design time value if the function is a constant. Additionally, an error message may be displayed and an existing byte code execution image may be invalidated and new byte code is generated to replace the existing byte code execution image if the expression is invalid.

In another aspect, a method for assigning a property of an object associated with an application includes receiving an expression into a property input section of the property entry sheet, the expression being referenceable at run-time as a data value; parsing the expression; generating from the expression a function; and storing the function as a run-time value.

Implementations of the invention include the following. The method may invalidate the object's byte code execution image. The method may also determine whether a run-time display of the object is automatically updated, and if so, the method may generate and execute the byte code. The method may also change an attribute of the object by determining which object property maps to the changed attribute; creating a constant function representing the changed attribute value; storing the constant function as a run-time property value and a design-time property value. When the object has a byte code execution image, the method may invalidate the byte code execution image and may also determine whether a run-time display of the object is automatically updated, and if so, the byte code is generated and executed. The method may also clone and store the function as a design time value if the function is a constant.

In another aspect, a computer-implemented object is provided with an object state and one or more interfaces providing access to the object state through a plurality of attributes, each of the attributes defined as a functional expression and referenceable at run-time as a data value.

Implementations of the object include the following. The object's functional expression includes a function, an operator, a database column name, a variable, and/or a constant. Moreover, the attribute may be a static data value. Each property may have a name-value pair. The object's functional expression may be parsed to generate a function which is stored as a run-time value. The function may be cloned and stored as a design time value if the function is a constant. Further, in the event that the expression is invalid, an error message is displayed, the existing byte code execution image is invalidated, and new byte code is generated to replace the existing byte code execution image.

Advantages of the invention include one or more of the following. The invention is a visual business intelligence tool for building applications that extend beyond the limitations inherent in conventional forms-based or report-based applications. Specialized programmers are removed from the application development process and users are moved closer to the data so that application development time is reduced. User interfaces can be created quickly and easily for information rich databases and for applications such as data warehousing and decision support.

The invention's hyperlinks provides context and "look-ahead" information to applications. This capability supports several powerful advantages in building data-driven applications. First, users can see through portals into other views of their data without losing the context of where they are. The navigational path taken by a user browsing the application can affect the application itself, thus providing dynamic customization of the application. In addition, context portals simplify the consolidation of diverse data