Formulating queries to access multiple databases can be a formidable task especially when many terms from various databases and complex constraints are involved. To specify a multidatabase query, the user usually has to search through documents for exact database terms and learn the multidatabase language. This report presents QUICK (QUery Interface to CPL-Kleisli), a graphical user interface to multiple databases. CPL (Collection Programming Language) is a high-level multidatabase language built on top of an open query system Kleisli. QUICK allows users to handle overwhelming information from different data sources in an intuitive and uniform manner. The query specification is reduced to specifying user’s terms in his/her own world, selecting paths and specifying constraints in a graph. QUICK is able to automatically generate a CPL query that corresponds to the user’s intent. Additional graphical functions are provided for the user to fine-tune the query generated.

A multidatabase system is a distributed system that acts as a front end to many autonomous DBMSs and a global layer above the autonomous DBMSs through a global schema or a multidatabase language. The global user can access information from multiple sources in the multidatabase system in a single straightforward request. However, the multitude of information available in multidatabase systems often impedes the user from quickly formulating a query. One reason is that the user often has to search through numerous manuals or documents for exact database terms in order to precisely specify a query. For example, it is difficult to be sure that employee identification number is termed “emp_id” and not “emp-id” or “employee_id” in the multidatabase. Although this problem also exists for single database systems, the magnitude of information in multidatabase environments makes it a more immediate problem. For instance, the schema documentation of GDB (Pearson, 1991) (Genome Data Base), a collection of databases providing human genome information, is well over 300 pages. In addition, multidatabase users are usually occasional users, in the sense that they use their home databases most of the time and access multidatabases only occasionally. As such, it is unreasonable to require multidatabase users to provide exact database terms. Some form of help should be given to reduce the user’s effort in query specification.

Traditional textual query formulation requires syntactic and semantic knowledge of the language. A large number of graphical user interfaces exists for single database systems which make query specification more user-friendly. However, the issue of graphical user interfaces is not well-addressed in multidatabase systems, see Section RELATED WORK. In this paper, we present a prototype system QUICK (QUery Interface to CPL-Kleisli) to address this issue. The CPL (Wong, 1995) (Collection Programming Language) is a high level multidatabase language, built on top of an open query system called Kleisli (Wong, 1995), which can handle nested relations and structured files. QUICK is a graphical query interface which translates graphical specifications into CPL. The purpose of QUICK is to minimize the effort of end users in formulating queries for multidatabase systems. QUICK allows fast query formulation even with sporadic users having neither sufficient knowledge of query languages, nor extensive prior knowledge of database struct-
QUICK is written in Tcl 7.4/Tk 4.0 and it can be executed in any Unix environment with X-Windows system and Tcl 7.4/Tk 4.0 installed. To run a query, CPLTCL, a variant of CPL for interfacing with TCL, is required to be installed.

Figure 1 shows the overall architecture of the system. QUICK is running on the Engine module CPLTCL that executes the query sent by QUICK. To replace CPLTCL by another multidatabase language, only the Query Composer and Meta Dictionary need to be replaced. The Thesaurus Dictionary provides a synonym mapping between user terms and database terms. The Meta Dictionary contains the schema information about views and frequently used predicates between views in the form of graphs. We consider general predicates that are not necessarily join. The Meta Dictionary can only be modified by the DBA. The editing of graphs during a particular user session have no effect on the Meta Dictionary. Instead, the editing result is saved onto a separate user file kept in the Data Store. The user can retrieve this file in a later session. Within the QUICK, there are five main modules. The Thesaurus is responsible for extracting corresponding database terms for the user specified terms. The View Definition is responsible for extracting the subgraph that contains the database terms returned by the Thesaurus. The Graphical Editor is the core module which supports essential graphical functions. The IO module is responsible for accessing session files in Data Store. Finally, the Query Composer generates a well-formed CPL query from a graphical specification.

There are three layers in the use of this system — the Thesaurus layer, Graph layer, and CPL layer. A user can enter the system at any of these layers. An expert user may like to enter the system at the lowest CPL layer by directly formulating a query in CPL but in the comfort of the graphical environment. A naive user may like to enter from the Thesaurus layer or Graph layer. The Thesaurus layer is good for users with minimal knowledge of databases and when only user terms are known. To map user terms to corresponding database terms, an interactive confirmation by the user may be needed and certain context information such as description of database terms and containing views and databases will be available to help with the confirmation. Based on confirmed database terms, the View Definition module retrieves the relevant portion view and schema definitions from the Meta Dictionary and presents it to the user in the form of a subgraph. Then the user proceeds to formulate queries using graphical interface functions provided by QUICK. The user can also choose to skip the Thesaurus layer and work with the entire graph or retrieve a subgraph by manually removing the irrelevant nodes — the Graph Layer. This is suitable for users with some knowledge of the underlying databases.

Though the use of QUICK in this paper is based on the Genome databases, QUICK is a generic interface for general multidatabase applications. For a new application running on CPL, only new Meta Dictionary and the Thesaurus Dictionary need to be created (by the DBA); for a new application running on a multidatabase language other than CPL, the Query Composer also needs to be substituted. The rest of this paper is organized as follows. In the next section, we review related interface work on multidatabase systems. In the section BIOLOGICAL DATA SOURCES, we describe the example biological databases used in this paper. In the section AN APPLICATION WITH QUICK, we show how a multidatabase query can be formulated with our prototype system QUICK using some genome databases. The conclusion is given in the last section.

Related Work

Most multidatabase research projects have emphasized on schema conflict resolution, query optimization, query processing and concurrency control. Query formulation for multidatabase systems has been predominantly textual. For