

SemanticSQL: A Semantic Wrapper for Relational Databases

enterprise data modeling • relational SQL access • complex queries made simple • empowerment of the user • preserve existing investment in your legacy database applications • nothing at risk other than a minor one-time reverse engineering effort • physical database and transactions stay under the old system • decision support interface to a single legacy database • or a unified virtual front to several dissimilar databases

Naphtali David Rische

Director
High-performance Database Research Center
School of Computer Science
Florida International University
University Park, ECS243, Miami, FL 33199
(305)348-2025, Fax (305)348-1707
rishe@fiu.edu
<http://HPDRC.fiu.edu>

President
NOA, Inc.
A spin-off venture
4201 Collins Ave, Suite 2203
Miami Beach, FL 33140
(305)672-6471, Fax (305)673-4975
ndr@acm.org
<http://SemanticSQL.com>

The semantic wrapper provides an easier access to a legacy relational database, in parallel to continued access via existing legacy application software. Our system presents a semantic view over a relational schema. Our patented technology allows the use of standard SQL to access databases at semantic level, via simple and concise SQL queries. As SQL middleware, our system enables the user of third-party GUIs to formulate queries to existing databases with greater ease, yet without switching from their existing GUI.

1. FEATURES

- exceptional usability and flexibility
- shorter application design and programming cycle
- giving the user control via an intuitive structure of information
- empowerment of the end-user to pose complex ad hoc decision support queries
- directly supports conceptual data model of the enterprise

- Internet-integrated

Just as the Relational Database model provided a higher level of abstraction than did the common database structures of twenty years ago, the Semantic Wrapper provides the next step. Also, the Semantic Wrapper is an object-oriented database interface (with many additional features) and therefore is ideally suited to this acclaimed new paradigm. Current object-oriented database systems (OODB), while praised by computer scientists, have yet to significantly penetrate the commercial market. This is due to the lack of intuitive front-end tools and lack of compatibility with existing relational databases (RDB). Our Semantic Wrapper includes an intuitive GUI front end, is accessible over the Internet, and is compatible with industry-standard relational database query language SQL. It subsumes the Object-relational database technology. In fact, our Semantic Wrapper also supports an effectively simplified version of SQL, via the definition of virtual tables. These virtual tables provide a relational view of grouped attributes as single tables. With all relevant attributes in a single virtual table, the SQL complications of multiple table references and keys are eliminated, thereby making SQL ergonomic. Typical SQL programs (as well as other programs) are an order of magnitude shorter with the Semantic Wrapper than in RDBs, resulting in drastically reduced development and maintenance time and costs, as well as increased reliability. The Object-Relational systems, like Oracle-8, offer some semantic capability and an ergonomic improvement of SQL, "OSQL" by adding new syntactic constructs. SemanticSQL offers even greater ergonomics of SQL and that without changing SQL syntax.

Among the advantages of the Semantic Wrapper are:

- *Semantic view mirrors real world.* This simplifies database design; substantially enhances the client's understanding of their database and lets him be in control; allows to capture business rules.
- *Complex relations made simple.* For example, "many-to-many relations" are represented in a natural way, while in relational databases these have to be modeled by additional tables.
- *Queries made simple and very short.* Queries can be up to ten times shorter (and so easier to pose) than in relational databases. For example, the user need not bother about "joins" — cross-references between relational tables, many-to-many relations, inheritance.
- *Shorter application programs.* User programs for a semantic view are substantially shorter than for a relational one, achieving major improvements in the application software development cycle, maintenance, and reliability.
- *SQL.* We have adapted SQL, the standard relational database language, to semantic databases. Programs in SQL via a semantic view tend to be an order of magnitude simpler and shorter than if they were written directly for a relational database.
- *Interoperability.* Our ODBC driver for the Semantic Wrapper allows SQL querying of a semantic database and interoperability with relational database tools, e.g. end-user systems like MS Access Query-By-Example or Crystal Reports. In these tools the number of user keystrokes required is proportional to the size of the generated SQL program. So again, savings are realized and simplicity is attained through the use of the

semantic view. An Embedded SQL interface for C and C++ is also provided.

- *Internet and intranet.* Database operations can be performed via web browsers.
- *Reverse engineering.* Our technology includes a tool that aids in reconstruction of a conceptual/semantic schema and documentation of a legacy relational database.

2. PRINCIPLES OF SEMANTIC SQL

Structured Query Language (SQL) is the standard language used to write queries for relational databases. We propose a method, called Semantic SQL (or Sem-SQL), to interpret SQL with respect to semantic and object database schemas. The syntax of Sem-SQL is exactly the same as the syntax of Open Database Connectivity (ODBC) 3.0 standard SQL, but it is interpreted differently. We are re-interpreting SQL in order to further the following goals. SQL is a uniform interface provided by almost every database system; it is, perhaps, the most popular database language and it is known by millions of users. The availability of the Sem-SQL interface will significantly enhance the accessibility of semantic and object-oriented databases. Sem-SQL also affords us the possibility of supporting ODBC, which is a standard database-access interface.

Sem-SQL and standard SQL are alike in syntax. However, from the users' point of view, using Sem-SQL will be different from but easier than using standard SQL. Sem-SQL queries refer to a virtual relational schema. This virtual schema consists of inferred tables, which are defined as a spanning tree of all the relations reachable from a given semantic category. (The central notion of semantic models is the concept of object, which is any real world entity that we wish to store information about in the database. The objects are categorized into classes according to their common properties. These classes are called categories.) Users query the database as if there were a universal table for each class with all the information derivable from it. However, a virtual table is never physically generated. Therefore, Sem-SQL is able to relieve users of explicitly expressing joins; conventional relational SQL requires them to do so.

However, updates against a derived user view, and in particular against the virtual tables, are inherently ambiguous. Therefore, disambiguating semantics are provided in the data manipulation language part of Sem-SQL in terms of the underlying semantic database. SQL insert, delete, and update statements can then be applied to virtual tables, preserving the intuitive meaning of these operations. Sem-SQL enables users to manipulate data in a more intuitive way than the standard SQL does, so it turns out to be simpler and more user-friendly.

3. SEM-SQL AS AN INTERFACE TO EXISTING DATABASE SYSTEMS

Relational databases are in common use today. We provide their end-users with a user-friendly query interface because most users have not been sufficiently trained to use standard SQL. Sem-SQL is more intuitive than the regular use of SQL against a relational database. Our solution is to present a semantic view of the relational schema. Users compose their queries in Sem-SQL based on this semantic view. Queries in Sem-SQL are then translated into relational SQL queries that are semantically equivalent. The basic idea of the query

transformation is to restore the semantic query, which is usually formulated in terms of the virtual tables, by adding the join conditions or sub-queries explicitly in the WHERE clause. This is achieved by referring to the mapping information between the semantic view and the relational schema. The basic components in relational schemas are tables, attributes, and foreign key links. Tables and attributes can be mapped to categories and attributes respectively in the semantic model, while foreign key links can be represented by abstract binary relations in semantic model. Our Semantic Wrapper technology uses a knowledge base to store this mapping information. Sometimes such basic information may not be enough to complete the transformation. Therefore, the system includes a set of inference rules to derive new knowledge that is essential during the query transformation. These techniques can also be applied when integrating relational databases and semantic databases together in a heterogeneous multi-database environment where there are a number of autonomous semantic or relational databases.

The semantic wrapper of a relational database first imports the relational schemas of databases and automatically converts them to semantic schemas. This conversion process maps every table to be a category and every functional dependency to be a relation in the semantic schema. This automatically generated schema does not contain semantically rich information such as inheritance, meaningful relation names, etc. Relational schemas are unable to represent such complex semantics so they cannot be automatically generated from the schema information. Our technology includes a tool, called the Knowledge Base Tool (KDB Tool), which is capable of customizing the generated schemas (enriching them with semantic information) with the interaction of the Database Administrator (DBA). The acquired knowledge and mapping between semantic and relational schemas are stored in the knowledge base. The wrapper provides not only a semantically rich schema for the relational database but also an easy-to-use query language, Semantic SQL, for querying the generated semantic schema. A query translator module transforms Semantic SQL queries posed on the semantic schema into semantically equivalent SQL statements on the relational schema. It uses the mapping information generated in the knowledge base along with the semantic and relational schemas. The query translation process uses temporary views to generate the appropriate projections of the virtual tables. Next, it proceeds to apply outer joins between these temporary views to provide query results. An important point to note is that the query translation process often generates substantially larger relational SQL statements for corresponding Semantic SQL statements. Though our translation algorithm does not provide optimal-size translated queries for every possible Semantic SQL query, this illustrates the ease of using Semantic SQL queries to generate complex queries. Since the translation process is automated, users are required only to specify the simpler Semantic SQL statements.

4. EXAMPLES OF SEMANTIC SQL AND COMPARISON TO RELATIONAL SQL

This section contains: the semantic schema of a Hydrology application; a normalized relational schema of the same application (a real schema, not our virtual schema); several SQL statements written for the semantic schema and (for comparison) for the relational schema.

The Hydrology schema of this example is actually a small one-page subschema of the 100-page schema of the database that we have developed for the Everglades National Park.

4.1. Hydrology Application, Semantic Schema

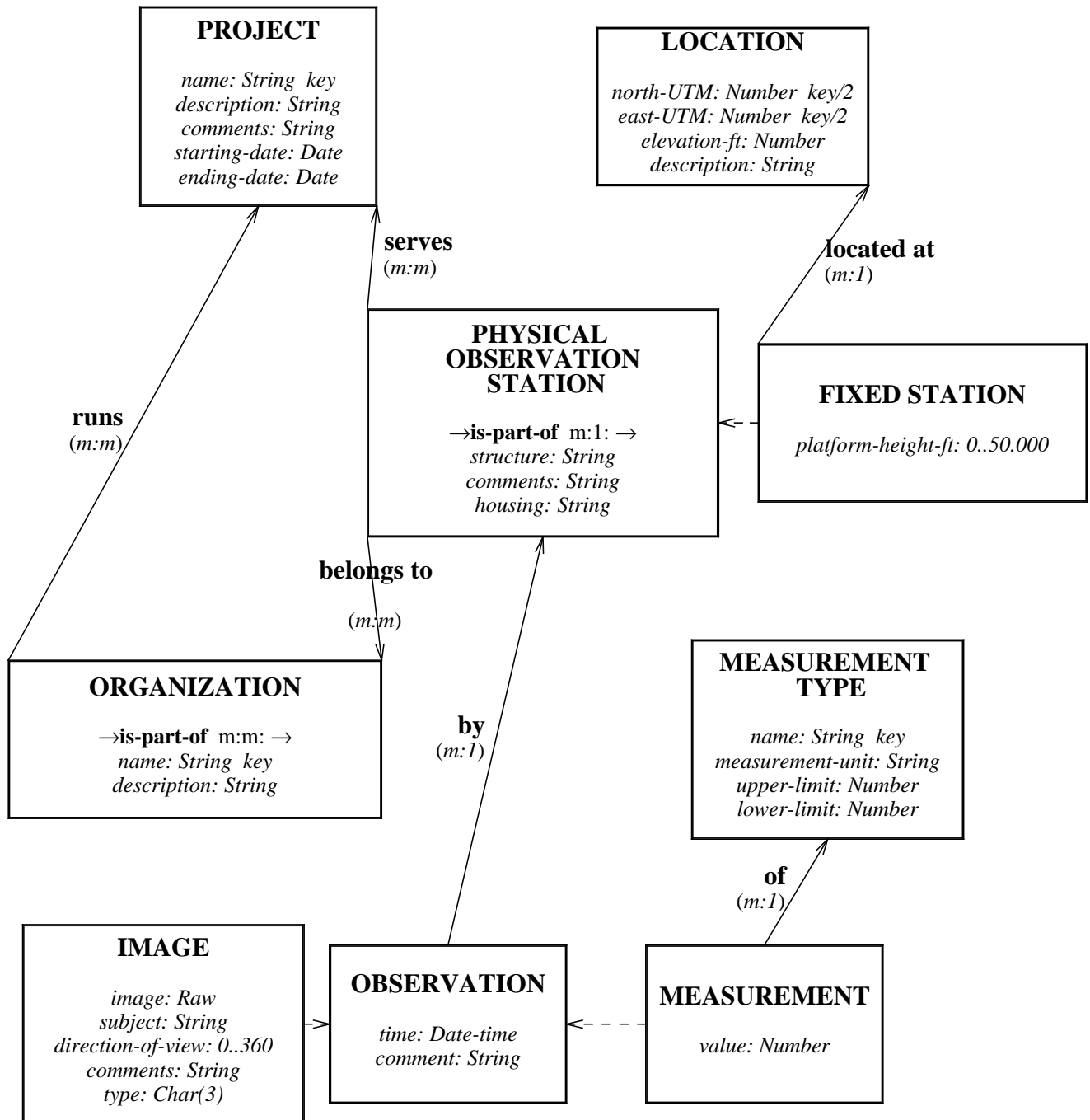


Figure 4-1. Semantic sub-schema for physical observations.

Boxes are categories of objects (dashes connect sub- to super-categories), solid arrows are semantic relationships (many-to-many relationships are marked *m:m*). Keys are optional, changeable, combinable identifiers. Numbers are optionally of unlimited size and precision.

Strings and raw attributes are optionally of unlimited length.

4.2. Relational Schema of the Hydrology Application

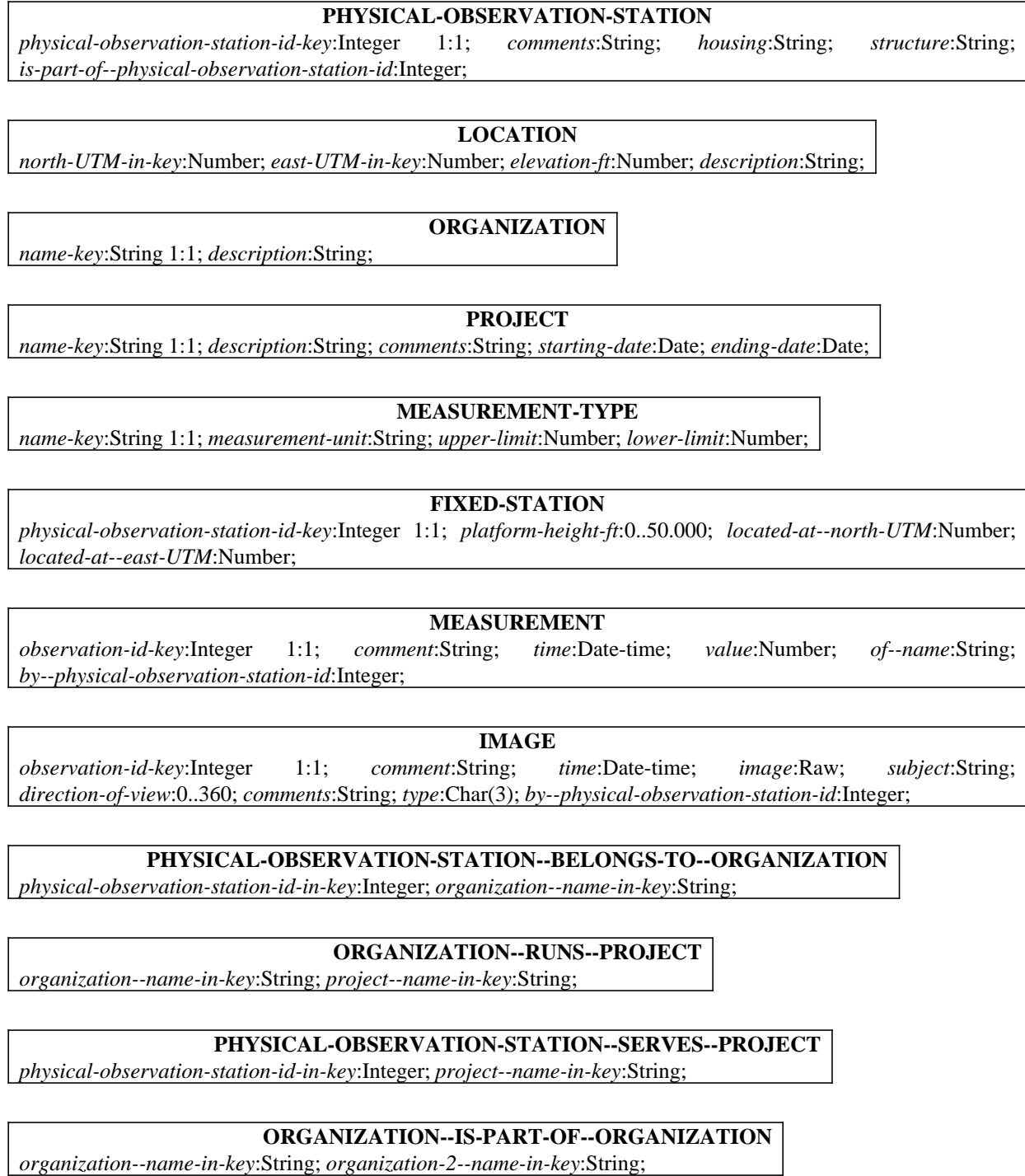


Figure 4-2. Relational sub-schema for physical observations.

This schema developed for a relational DBMS is functionally equivalent to the previous semantic schema (if we disregard the "flexibility parameters": numbers will have limited size

and precision, keys must always exist and cannot be changed, etc.)

4.3. Program Size Comparisons: SQL

1. List of the time and housing of temperature measurements over 50 degrees

SQL statement based on semantic schema:

```
select housing,time from MEASUREMENT where of__name='Temperature' and value>50
```

SQL statement based on relational schema:

```
select housing, time
from PHYSICAL_OBSERVATION_STATION, MEASUREMENT
where exists
  (select * from MEASUREMENT-TYPE
   where name_key = of__name and name_key = 'Temperature' and
    by_physical_observation_station_id = physical_observation_station_id_key and
    value > 50)
```

2. The descriptions of organizations and locations of their fixed stations

SQL statement based on semantic schema, Alternative 1:

```
select description, belongs_to__located_at__LOCATION from ORGANIZATION
```

SQL statement based on semantic schema, Alternative 2:

```
select description, LOCATION from ORGANIZATION
```

SQL statement based on relational schema:

```
select description, LOCATION.north_UTM_in_key, LOCATION.east_UTM_in_key
from ORGANIZATION, LOCATION
where exists
    (select * from FIXED_STATION
    where exists
        (select *
        from
            PHYSICAL_OBSERVATION_STATION__BELONGS_TO__ORGANIZATION
        where name_key = organization__name_in_key and
            PHYSICAL_OBSERVATION_STATION__BELONGS_TO__ORGANIZATION.
                physical_observation_station_id_in_key =
                FIXED_STATION.physical_observation_station_id_key and
                located_at__north_UTM = north_UTM_in_key and located_at__east_UTM =
                east_UTM_in_key ))
```

3. The observations since January 1, 1993 (including images, measurements and their types) with location of the stations

*SQL statement based on **semantic** schema:*

```
select OBSERVATION__, of__, LOCATION from OBSERVATION where time > '1993/01'
```

*SQL statement based on **relational** schema:*

```
(select MEASUREMENT_TYPE.*, LOCATION.north_UTM_in_key,  
LOCATION.east_UTM_in_key, MEASUREMENT.*, NULL, NULL, NULL, NULL,  
NULL, NULL, NULL, NULL, NULL
```

```
from MEASUREMENT_TYPE, LOCATION, MEASUREMENT
```

```
where time > '1993/01' and exists (select * from FIXED_STATION where  
by__physical_observation_station_id = physical_observation_station_id_key and  
located_at__north_UTM = north_UTM_in_key and located_at__east_UTM =  
east_UTM_in_key and of__name = name_key )) union
```

```
(select MEASUREMENT_TYPE.*, NULL, NULL, MEASUREMENT.*, NULL, NULL,  
NULL, NULL, NULL, NULL, NULL, NULL
```

```
from MEASUREMENT_TYPE, MEASUREMENT
```

```
where time > '1993/01' and not exists (select * from FIXED_STATION where  
by__physical_observation_station_id = physical_observation_station_id_key and  
of__name = name_key )) union
```

```
(select NULL, NULL, NULL, NULL, LOCATION.north_UTM_in_key,  
LOCATION.east_UTM_in_key, NULL, NULL, NULL, NULL, NULL, NULL,  
IMAGE.*
```

```
from LOCATION, IMAGE
```

```
where time > '1993/01' and exists (select * from FIXED_STATION where  
by__physical_observation_station_id = physical_observation_station_id_key and  
located_at__north_UTM = north_UTM_in_key and located_at__east_UTM =  
east_UTM_in_key )) union
```

```
(select NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,  
NULL, IMAGE.*
```

```
from IMAGE
```

```
where time > '1993/01' and not exists (select * from FIXED-STATION where  
by__physical_observation_station_id = physical_observation_station_id_key))
```

5. VIRTUAL DATA WAREHOUSES

Data warehousing and On-line Analytical Processing (OLAP) play a more and more important role in the Information Systems community because of the requirements of using a Decision Support System (DSS) to gain competitive advantages for businesses. To facilitate complex analyses and visualization, the data in a data warehouse is typically modeled multi-dimensionally, and such data is often called data cubes in an OLAP data source. Because of the abundant semantics, for example operational semantics, being stored, the semantic object model is undoubtedly more suitable to represent data cubes than the traditional relational model. However, a relational database is also a necessary component in a DSS since much essential data had already been stored in the existing relational databases. Thus, our wrapper of semantic view over a relational schema makes it easier to integrate the OLAP data sources and relational databases as a whole in data warehouse. Wrapping the relational databases semantically would greatly increase the interoperability of relational databases and the OLAP data sources. In our approach, the client will not need to move data from the transactional database into a Data Warehouse — the latter will be a virtual front not affecting the legacy database.

Users of DSS are typically interested in identifying trends rather than looking at individual records in isolation, and therefore decision-support queries are usually more complex than ordinary queries. So, there comes another advantage of using semantic model in DSS because of the intelligence of the SemSQL as depicted before.

With wrapped relational databases as the data repository, the provided ODBC interface of the SemSQL enables other business intelligence tools, such as COGNOS products, to retrieve information directly. End users can benefit from whatever desired tools to develop their DSSs. In short, our wrapper product over relational databases provides developer with a better solution to implement data warehouse and OLAP technology.

6. COST OF OWNERSHIP

In order to use the semantic wrapper, the Client's analysts or consultants need to define a semantic schema for the application and specify translation rules. The effort depends on the degree of complexity of the database and the quality of its existing documentation. If the existing database is well documented and has a conceptual schema then the effort is relatively small. Otherwise, one needs to reverse engineer the existing database schema, at a greater effort, yet much smaller than the effort that went into the original systems analysis of this application. As a byproduct, the reverse engineering effort creates documentation of the existing database, which facilitates its use, improves its reliability, helps in training personnel, and allows the executive level understand their information assets. We provide a tool that automates the reverse engineering and wrapping process to the degree possible, with some manual guidance from the systems analyst. The tool also produces extensive documentation and indices.

7. COMPETITION

There are three principal types of efforts regarding improving query interfaces over the relational databases. They are SQL3 (sometimes called SQL 1999), Object Query Language (OQL), and some graphical (or visual) query languages.

The SQL3 approach, often called object-relational, can be regarded as bringing the best of object-orientation into the relational world, while OQL, on the other hand, aims to bring the best of SQL into the object-oriented world. Both approaches focus on enhancing the expressiveness of the current SQL by changing the data models. When users need to develop a new application, the new features that come with the object-orientation, for example super/subclasses and inheritance, enable them to do the database modeling in a way closer to the real world. However, it is not very helpful to the existing databases that have been developed under the conventional relational model and therefore have no object-oriented features at all.

Our approach of semantic wrapping of the existing relational database provides a semantic view of the relational schema and then enables user to query the relational database with Semantic SQL. Similar to each of the above approaches, our approach also provides users with a number of object-oriented features, such as super/sub-categories, relationships, and inheritance, which can be utilized by users when formulating their queries. Without changing the basic SQL syntax, we enhance its ease of use, expressiveness, and conciseness. Users can benefit from reach semantics while the existing database remains pure relational without any modifications.

There is another advantage of our approach with respect to the SQL syntax and programming style. Both SQL3 and OQL introduce new syntax as well as a semi-procedural programming paradigm to fulfill the object-oriented requirements. Users who are used to programming with a pure declarative language such as the SQL92 have difficulty or are uncomfortable when being asked to switch from declarative programming to procedural programming. The Semantic SQL, by the contrast, remains to be a pure declarative language, and more importantly, is syntactically identical to ODBC SQL 2.0. Thus, it only requires a minimum of prior training in using Semantic SQL compared with using either SQL3 or OQL.

The ODBC compatible syntax of the Semantic SQL also allows our product to be easily connected to any ODBC compliant tool as a middle-ware. This is what SQL3 or OQL efforts cannot do at present.

Another type of attempt to make the query interface more friendly is to use graphical query languages. Their advantage is query visualization. However, there are downsides with the current graphical query languages. For example, a navigational paradigm with a hypertext language is usually restrictive and inefficient. As a result, it often yields useless information after a user has spent a lot of time in navigating in a perplexed cyberspace. A menu-driven query paradigm as found in some desktop databases such as in dBaseIII+ or a table-like browser Query-By-Example (QBE) of; Microsoft Access, frees users from having to learn the SQL syntax. But their usage for complex queries with joins is too complicated for an end-user. Further, their users also must know the logical structures of relational database such as foreign key links clearly so as to compose queries with explicitly expressing each join correctly. Specification of outer joins (which are the intended semantics of most typical

queries) is next to impossible. However, when graphical tools like this talk ODBC SQL as their intermediate language via SemanticSQL as middleware, in most cases, users can query the whole database as if it were a single virtual table. Users can formulate their queries by simply picking the desired attributes.

Other efforts include G-log, Functional Graphical Language, Visual-Query-Language, Graqla, DUO, and Query-By-Diagram. Most of them utilize formalisms based on instance-level graphs, which may run into a scalability problem since this type of graphs in a large database is potentially very complex. Moreover, similar to the above methods, they are all less intelligent than Semantic SQL.

Several software firms have products facilitating the interoperability of different data models, for example TITANIUM at Micro Database System. Their efforts are enabling access to a single database with different query interfaces based on users' preference. Using these tools, developers first need to convert the existing database into a specific database product, for example the TITANIUM database engine. In some senses, such tools only wrap the particular types of DBMS while SemanticSQL can wrap any kind of relational databases as long as it supports ODBC. There is no data conversion actually happening in our wrapper. That is, we do not change anything in the existing databases, but only enable users to access these databases in a more friendly manner, which is simple, intuitive, and intelligent.

8. VIRTUAL FRONT TO A COLLECTION OF HETEROGENEOUS DATABASES

Many corporations own several databases. For example a car leasing company has a payroll database in ORACLE and a customer database in INFORMIX. Or, after merger between the Bank of America and NationsBank there two differently-structured customer databases.

Our system will have a module of a *heterogeneous distributed database*, i.e. a virtual database encapsulating simultaneously several data sources and presenting them to the user as if they were one database. The data sources under the virtual front can be relational databases, intranet/internet information services, and others.

9. CURRENT INVESTMENT

The present technological development is performed by Florida International University's High Performance Database Research Center under directorship of Dr. Naphtali Rishe. The present development benefits from the current HPDRC resource of \$30 million, most of it provided by the U.S. Government, including: NASA and the National Science Foundation.

10. INTELLECTUAL PROPERTY

The government sponsors have released all intellectual property of this project to FIU, subject to a free limited license to the U.S. Government. FIU, in turn, has released this IP to Dr. Rische subject to revenue sharing.

This intellectual property is comprised of two components, disclosed and trade-secret.

1. Disclosed
 - U.S. Patent (notice of allowance received in May 2004)
 - copyrighted software
 - Dr. Rische's books and papers in scientific journals and conferences
2. The trade-secret component consists of a large body of algorithms and software. All the employees of the High Performance Database Research Center have entered into non-disclosure agreements.

11. Dr. Rische

Rische's methodology for the design of database applications and his work on the Semantic Binary Database Model were published as a book by Prentice-Hall in 1988. Rische's Semantic Modeling theory was published as a book by McGraw-Hill in 1992. Rische is the editor of three books and author of 200 papers. Dr. Rische has been awarded over \$30 million in research grants by government and industry. Dr. Rische also has extensive experience in database applications and database systems in the industry. This included eight years of employment as head of software and database projects (1976-84) and later consulting for companies such as Hewlett-Packard and the telecommunications industry. Since Rische completed his Ph.D. at Tel Aviv University in 1984 he worked as an assistant professor at the University of California, Santa Barbara (1984-1987), and associate professor (1987-1992) and professor (1992-) at Florida International University (FIU). Rische is the founder and director of the High Performance Database Research Center at FIU.

12. QUALITY ASSURANCE

HPDRC has a Quality Assurance Laboratory, where all systems undergo vigorous testing independent of the software programmers, set of regressions are developed, and compliance with software and documentation is reviewed.

13. EVALUATION VERSION

Evaluation version of the system is shipped on a CD-ROM for MS Windows. It creates a sample MS Access database, preset semantic view thereof, and an SQL querying facility thereto at semantic level. A tool is provided to reverse engineer any other MS Access database. Also provided is the middleware functionality where the user can enter vastly simplified queries via the GUI of MS Access QBE — QBE then generates semantic SQL

queries to the middleware, which expands them into more complex SQL queries being sent to the actual database.