1. THE SEMANTIC DATABASE: FEATURES

Our Semantic Object-Oriented Database Management System (Sem-ODB) is so much easier to use and more efficient than its predecessors that it can truly be considered as the first of the next generation of such information handling systems. This quantum step in usability and capability is a direct result of its unique design, in which all data is regarded as a collection of facts, rather than data residing in artificially organized tables or in other forms. This design, combined with our innovations in database implementation, allows:

- 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
- superior efficiency, high level of optimization (transparent to the user)
- manifold reduction in storage size for many applications, such as Data Warehouses
- directly supports conceptual data model of the enterprise
- Internet-integrated

This high level of abstraction combined with novel processing algorithms enables significant efficiency benefits. Just as the Relational Database model provided a higher level of abstraction than did the common database structures of twenty years ago, the Sem-ODB provides the next step. Also, Sem-ODB is an object-oriented database (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 Database 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 Database 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 in Sem-ODB 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. Sem-ODB offers even greater ergonomics of SQL and that without changing SQL syntax.

There are additional advantages over the Relational and the Object-Relational models. There is no requirement for keys in Sem-ODB, which eliminates the problems of maintaining static keys which are required by other database systems. Sem-ODB handles variable length data, heterogeneous data, multi-valued data, and missing or incomplete data, all without any waste of space. Again, the relational model does not work well for these conditions, which are very common in the real world. Further, the Sem-ODB model can easily add new relations and attributes without reformatting, something that can be prohibitively time-consuming in large RDBs.

The Sem-ODB model also lends itself perfectly to non-conventional data, such as multimedia data, without trying to force it into a grid format. The relational model is notoriously poor in this regard. Up until the present, databases have been almost exclusively business-oriented products. The popularization of multimedia databases with intuitive manipulative and display capabilities has the potential to turn the database into a consumer product and vastly expand its market.

Among the advantages of Sem-ODB 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; avoids restricting and altering the client’s real world as is typically done when the real world is twisted into relational tables.

- **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.

- **Shorter application programs.** User programs for a semantic database are substantially shorter than for a relational one, achieving major improvements in the application
software development cycle, maintenance, and reliability.

- **No restrictions on data.** All data types are of unlimited length, stored compactly. The designer does not have to specify a limit of string length in any field — strings can be from 0 bytes to gigabytes-long. We have developed techniques to compactly represent numbers of unlimited size and precision. In this varying-length representation, the number "100" takes only one byte, the number "1 trillion" takes only two bytes, the number $\pi$ (3.14...) truncated after 10,000 digits takes thousands of bytes.

- **Very efficient full indexing.** Full indexing — indexing on every attribute and relationship — is required in many applications (e.g. Data Warehouses) where the patterns of ad-hoc queries cannot be predicted in advance. Further, our algorithm guarantees optimality of the basic queries defined in our Semantic Algebra; this includes optimality of range queries.

- **Flexible classification of objects.** Objects can belong to several different categories at the same time, and can move between categories instantly. Full inheritance of properties from super-categories to subcategories is provided.

- **Lazy queries.** We have developed an original technique of lazy queries that allows disk accesses to retrieve facts to be delayed until they are actually needed (if they are needed at all). It also allows efficient query optimization, including lazy query intersection and subtraction.

- **Compaction of sparse data.** NULL values take no space at all. (Sparse tables in relational databases may waste space and processing time.)

- **No keys are needed.** Keys in a relational database are a substantial burden on the database designer and database user and restrict flexibility of data.

- **Automatic consistancy of database.** For example, "referential integrity constraints" are enforced automatically by the semantic database.

- **Better concurrency control.** We have developed a semantic optimistic concurrency control algorithm supporting maximal theoretical granularity without the overhead that such precision would normally require. Further, the algorithm offers maximal safety.

- **Multi-processor parallelism.** A semantic database can be highly parallel. An efficient load balancing algorithm has been designed that will allow arbitrary chunks of data to be stored on different servers, optimizing the server performance.

- **SQL.** We have adapted SQL, the standard relational database language, to semantic databases. Programs in SQL for Sem-ODB tend to be an order of magnitude simpler and shorter than for RDB.

- **Interoperability.** Our ODBC driver for the Sem-ODB Engine is fully operational, allowing SQL querying of a semantic database and interoperability with relational database tools, e.g. end-user systems like MS Access Query-By-Example. 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 Sem-ODB model. An Embedded SQL interface for C and C++ has also been developed and is fully operational.
- **No tuning required.** Sem-ODB requires virtually no manual tuning of DBMS to an application (for performance optimization). Market leading DBMS’s require substantial tuning by database experts and re-tuning when database parameters change.

- **Benchmarks.** Our benchmarks have demonstrated, for certain types of business applications, a 40-fold query speed improvement and 3-fold storage size reduction vs market-leading DBMS (even when the latter are configured with full indexes, are thoroughly tuned to the application, and allowed to gather statistics to improve their performance). Such savings typically take place when the ad-hoc query load requires full indexing or when the structure of information has complex semantics or when there are many null values or when there are great variances in sizes of data items.

- **Internet and intranet.** Database operations can be performed via web browsers.

- **Small memory footprint.** The 2MB memory footprint of Sem-ODB makes it very suitable for embedded applications.
2. SEMANTIC, RELATIONAL, AND OBJECT-ORIENTED DATABASES: BASIC THEORY

The central notion of semantic database 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, called categories, need not be disjoint — that is, one object may belong to several of them. Further, an arbitrary structure of subcategories and supercategories can be defined. The representation of the objects in the computer is invisible to the user, who perceives the objects as real-world entities, whether tangible, such as persons or cars, or intangible, such as observations, meetings, or desires. The database is perceived by its user as a set of facts about objects. These facts are of three types: facts stating that an object belongs to a category; facts stating that there is a relationship between objects; and facts relating objects to data, such as numbers, texts, dates, images, tabulated or analytical functions, etc. The relationships can be of arbitrary kinds; for example, stating that there is a many-to-many relation address between the category of persons and texts means that one person may have an address, several addresses, or no address at all.

The relational database model, proposed by E. Codd in 1971, has become the state of the art of commercial database management. This model has, in a mathematically elegant way, presented the database user with an abstraction of data, isolating the user from the physical representation of data in computer storage. The data is presented to the user as a collection of tables. Each table is a set of rows. There are two types of tables: tables representing the application’s objects (object tables) and tables representing relationships between objects (relationship tables). The object tables roughly correspond to categories in the semantic models. Each object table consists of one or more columns (jointly called the key) that identify the objects, and several other columns that display data about the objects such as numbers or character strings. Every object must have a unique key value, such as a social security number for a person, or a street name plus a house number for a home. The key must be known at all times and may never change (or the database will be corrupted). Along with the key, an object table’s row contains data about the object, such as the person’s address, and some relationships to other objects. The latter relationships are represented by the keys of the related objects — for example, the social security number of the person’s spouse. This does not work for many-to-many relations, for which the other type of tables must be used: the relationships tables. As far as the system is concerned, the sets of objects of different tables are disjoint (although, the user can de-facto link between rows of different tables by using identical key values, but this causes immense problems in updating and querying the database). The relationships tables consist of rows cross-referencing the keys of related objects.

The mathematical abstraction of the relational model has allowed the introduction of powerful and easy-to-use user languages for retrieval and updates of databases. It has also allowed the recent development of efficient implementations. The latter were facilitated by the invisibility to the user of the computer processing, which permits optimization without affecting the user. The semantic models offer a higher degree of abstraction. This results in much more concise user programs, as well as speedier processing due to optimization and other factors. Beyond that, the semantic models offer a plethora of other features.
The relational databases have provided a good service in many conventional database applications. However, in situations where the structure of information is complex, or where greater flexibility is required (objects with unknown identifiers, or objects moving from one category to another, etc.), or where non-conventional data is involved (long texts, images, etc.), other approaches need to be considered: semantic and/or object-oriented databases.

Akin to semantic models are object-oriented database models. They offer, to various degrees, many of the features of the semantic models, in the sense of abstracting information, and, in addition, formalize some behavioral properties of the data. We have incorporated the latter behavioral properties in our semantic database system, thus unifying the semantic and object-oriented approaches to databases.

Bibliography:

3. SEMANTIC SQL AND Sem-ODB vs. RDB USE COMPARISON

This section serves several objectives:

- to present an example of a semantic database schema
- to compare it to a relational schema
- to introduce Semantic SQL
- to compare program sizes in SQL between semantic and relational databases.

3.1. Semantic SQL Principles

We have adapted SQL, the standard relational database language, to semantic databases. The original purpose of this adaptation was to be compatible with, and be able to communicate with, relational tools. Interestingly, it turned out that the size of a typical SQL program for a semantic database is many times smaller than for an equivalent relational database. While we have previously demonstrated substantial program-size advantage for other languages, we had not anticipated an even greater advantage with SQL — a specialized language for relational databases.

Our ODBC driver for the Sem-ODB Engine is fully operational, allowing SQL querying of a semantic database and interoperability with relational database tools, e.g. end-user systems like MS Access Query-By-Example. 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 by use of the Sem-ODB model.

Our application of SQL to semantic databases allows utilization of full semantics of data, applies to scientific and spatial data, properly treats missing values, and produces queries which are typically an order of magnitude shorter than if written in SQL for an equivalent normalized relational database — see examples in a later section.

We have developed several Semantic SQL related tools: ODBC SQL server — allowing interoperability with third party tools; embedded SQL — allowing programming of complex applications; plain SQL server — allowing submission of ad-hoc and pre-programmed SQL queries on Windows NT and Unix machines; an internet/intranet WWW server — allowing submission of SQL queries via standard internet browsers; internet SQL query generator — helping the user to define a query without writing SQL statements; graphical view selector — allowing the user to select a small view of the database against which to pose queries; and others.

3.2. SQL Interpretation

We use the same syntax as the standard ODBC SQL (with null values). However, our SQL queries refer to a virtual schema. This virtual schema consists of an inferred table $T$ defined for each category $C$ as a spanning tree of all the relations reachable from $C$. This virtual table $T$ is never physically generated. The table $T$ contains every attribute reachable from category $C$.

Example: consider the semantic schema of Figure 3-1.
Figure 3-1. A semantic schema for a university application.

The following are some of the attributes of the virtual table STUDENT:
<table>
<thead>
<tr>
<th>full attribute name</th>
<th>abbreviation</th>
<th>type</th>
<th>sample value</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td>-</td>
<td>surrogate</td>
<td>123235</td>
</tr>
<tr>
<td>last_name</td>
<td>-</td>
<td>string</td>
<td>Smith</td>
</tr>
<tr>
<td>birth-year</td>
<td>-</td>
<td>integer</td>
<td>1970</td>
</tr>
<tr>
<td>the_student___the_offer___the_quarter__year</td>
<td>year</td>
<td>integer</td>
<td>1999</td>
</tr>
<tr>
<td>the_student___the_offer___the_quarter__season</td>
<td>season</td>
<td>string</td>
<td>Spring</td>
</tr>
<tr>
<td>the_student___final_grade</td>
<td>final_grade</td>
<td>integer</td>
<td>75</td>
</tr>
<tr>
<td>major</td>
<td>-</td>
<td>surrogate</td>
<td>CS</td>
</tr>
<tr>
<td>minor</td>
<td>-</td>
<td>surrogate</td>
<td>ECE</td>
</tr>
<tr>
<td>major__name</td>
<td>-</td>
<td>string</td>
<td>CompSci</td>
</tr>
<tr>
<td>minor__name</td>
<td>-</td>
<td>string</td>
<td>Electrical</td>
</tr>
</tbody>
</table>

3.3. **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.

3.3.1. **Hydrology application, semantic schema**
**Figure 3-2.** 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.
3.3.2. Relational schema of the hydrology application

<table>
<thead>
<tr>
<th>PHYSICAL-OBSERVATION-STATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical-observation-station-id-key:Integer 1:1; comments:String; housing:String; structure:String; is-part-of--physical-observation-station-id:Integer;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>north-UTM-in-key:Number; east-UTM-in-key:Number; elevation-ft:Number; description:String;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-key:String 1:1; description:String;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-key:String 1:1; description:String; comments:String; starting-date:Date; ending-date:Date;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEASUREMENT-TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>name-key:String 1:1; measurement-unit:String; upper-limit:Number; lower-limit:Number;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FIXED-STATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical-observation-station-id-key:Integer 1:1; platform-height-ft:0..50.000; located-at--north-UTM:Number; located-at--east-UTM:Number;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MEASUREMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>observation-id-key:Integer 1:1; comment:String; time:Date-time; value:Number; of--name:String; by--physical-observation-station-id:Integer;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IMAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>observation-id-key:Integer 1:1; comment:String; time:Date-time; image:Raw; subject:String; direction-of-view:0..360; comments:String; type:Char(3); by--physical-observation-station-id:Integer;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PHYSICAL-OBSERVATION-STATION--BELONGS-TO--ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical-observation-station-id-in-key:Integer; organization--name-in-key:String;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORGANIZATION--RUNS--PROJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>organization--name-in-key:String; project--name-in-key:String;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PHYSICAL-OBSERVATION-STATION--SERVES--PROJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical-observation-station-id-in-key:Integer; project--name-in-key:String;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORGANIZATION--IS-PART-OF--ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>organization--name-in-key:String; organization-2--name-in-key:String;</td>
</tr>
</tbody>
</table>

Figure 3-3. 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.)
3.3.3. 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:*

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

*SQL statement based on relational schema:*

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

4.1. Early Adoptions

- **Everglades.** A very large semantic schema (over 2000 relations and attributes) has been developed for environmental research activities at the South Florida (Everglades) Research Center of the National Park Service.

- **NASA RAC.** In October 1996, Florida International University, through our Database Research Center, was named the fifth NASA Regional Applications Center (RAC) for storage, processing, and distribution of environmental data (satellite and other) to serve local and global communities. Per NASA’s instructions, this RAC uses our semantic database technology. In the next step, NASA wishes to operate Sem-ODB at the primary RAC located at the NASA Goddard Space Flight Center. In the third step, NASA wishes to install Sem-ODB at the other RACs throughout the United States.

- **Spatial Data Store.** At present the FIU RAC stores semantic databases containing significant quantities of spatial data in the following areas: Ikonos satellite 1 meter data, Ocean Temperature; Ozone Layer Thickness; Reflectivity; SeaWiFS; LANDSAT, USGS Areal Photography, and GOES Weather (we own a ground station that receives data directly from the GOES satellite and loads it into our database in real time). Under an agreement with USGS, we are loading into the database 13 terabytes of aerial photography.

- **TerraFly.** An application allowing virtual flight through remote sensing data, intended for mass market, has been developed as a CD-ROM embedding Sem-ODB and spatial data and as a web-based system www.TerraFly.com.

4.2. Current Investment

The present development is funded at about $17.5 million, much of it by the U.S. Government, including: NASA ($5.5M) and the National Science Foundation ($4.5M).

This project currently involves about 100 professionals. Of them, 27 are full-time employees, including 13 Ph.D.’s. The rest are half-time employees who are students (working towards Ph.D., M.S., or B.S. degrees).

4.3. The High Performance Database Research Center

Sem-ODB is a flagship project of the High Performance Database Research Center (HPDRC), affiliated with the School of Computer Science at Florida International University in Miami. The HPDRC is sponsored by federal and state agencies, as well as industry. The staff consists of some of the top computer science talent worldwide with national prize winners from China and Russia, and numerous Ph.D.s, Ph.D. candidates, and other post-graduates.

NASA has been sufficiently impressed by HPDRC technology and capability that they are considering Sem-ODB as one of their standards. NASA has also cited the HPDRC as "the
best one" among the 10 database research efforts they are funding. This compares the HPDRC very favorably against well-known entrenched interests working on "cutting edge" extensions to relational databases.

The HPDRC is led by Professor Naphtali Rishe. Rishe’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. Rishe’s Semantic Modeling theory was published as a book by McGraw-Hill in 1992. Rishe is the editor of three books and author of 23 papers in journals (including IEEE KDE, DKE, Information Systems, Fundamenta Informaticae), 7 chapters in books and serials (including 3 in Springer Verlag’s LNCS), over 50 papers published in proceedings (including ACM SIGMOD, PDIS, IEEE DE, ACM SIGIR, SEKE, ARITH, FODO). Dr. Rishe has been awarded millions of dollars in research grants by government and industry. His research is currently sponsored at about $17M by NASA, NSF, and other agencies. Dr. Rishe 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 Rishe 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). Rishe is the founder and director of the High Performance Database Research Center at FIU. Dr. Rishe chaired the program and steering committees of the PARBASE conference and is on the steering committee of the PDIS conference series.