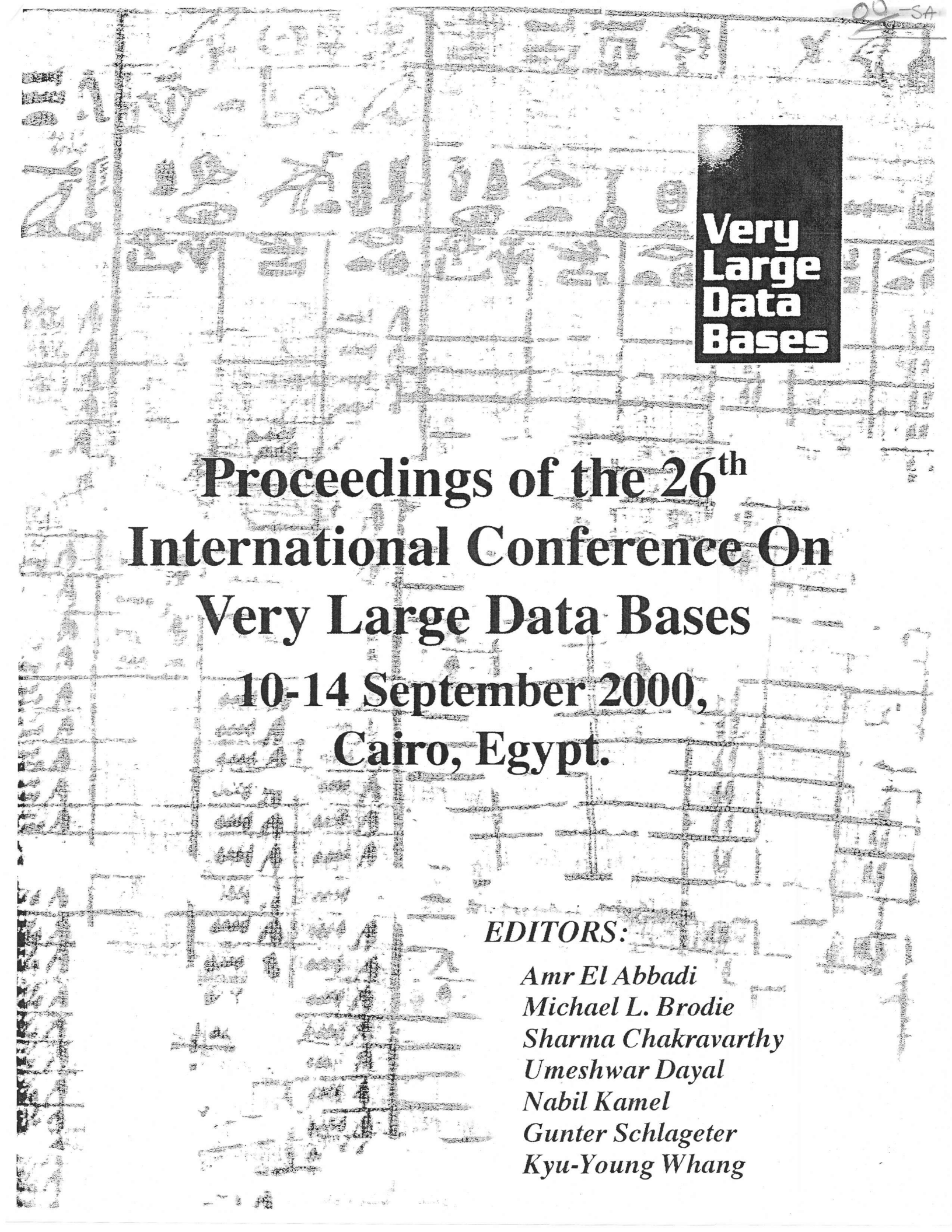


Proceedings of the 26th International Conference On Very Large Data Bases

10-14 September 2000, Cairo, Egypt.

EDITORS:

Amr El Abbadi
Michael L. Brodie
Sharma Chakravarthy
Umeshwar Dayal
Nabil Kamel
Gunter Schlageter
Kyu-Young Whang

SemanticAccess: Semantic Interface for Querying Databases

Naphtali Rishe
Shu-Ching Chen
Alexander Vaschillo

Jun Yuan
Xiaoling Lu
Artyom Shaposhnikov

Rukshan Athauda
Xiaobin Ma
Dmitry Vasilevsky

High-performance Database Research Center
School of Computer Science
Florida International University
University Park, Miami, FL 33199
USA
{rishen, yuanj, rathau01, chens, lu01, ma01, avasch01, shaposhn, dvasil01}@cs.fiu.edu

Abstract

Semantic Binary Object-oriented Data Model (Sem-ODM) provides an expressive data model (similar to Object-oriented Data Models) with a well-known declarative query facility - SQL (similar to relational databases). Advantages of using Sem-ODM include (i.) friendlier and more intelligent generic user interfaces; (ii.) comprehensive enforcement of integrity constraints; (iii.) greater flexibility; (iv.) substantially shorter application programs; and (v.) easier query facility. SemanticAccess is a set of tools developed to provide a semantic interface to Semantic Binary Object-oriented Databases (Sem-ODB) as well as relational databases. This presentation focuses on the system architecture of SemanticAccess including Semantic Binary Object-oriented Data Model, Semantic SQL query language, Semantic Binary Database and a wrapper developed for relational databases.

Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

1. Purpose

Semantic Binary Object-Oriented Data Model (Sem - ODM) [4] combines the advantages of relational and object-oriented data models. Sem-ODM provides expressive data modeling capabilities, similar to object-oriented data model, but also has the simplicity of constructs similar to relational data model (which provides only one construct, namely *table*). Sem-ODM consists of *category*, which may be inherited and *relation*, which is a relationship between categories. Detailed discussion on Sem-ODM can be found in [4]. One of the major advantages contributing to relational databases' success is the standard query language, SQL, which is declarative in nature. Object-Oriented Database (OODB) query languages are usually correlated with an Object-Oriented Programming Language (OOPL) [1] and/or are procedural in nature [3]. We have adapted SQL (SQL-92) for Sem-ODM (called Semantic SQL), thus providing a well-known declarative query language for Sem-ODM. We, at HPDRC [8], have developed a fully functional Semantic Binary Object-Oriented Database System (Sem-ODB). Due to the above-mentioned features, Sem-ODB has many advantages including friendlier and more intelligent generic user interfaces, comprehensive enforcement of integrity constraints, greater flexibility, substantially shorter application programs and easier query facility. We have been able to successfully deploy Sem-ODB for non-traditional applications such as Geographic Information System (GIS) [2] at the NASA Regional Application Center at Florida International

University. In order to propagate the advantages of Sem-ODM and its query facilities to access various databases, we have developed SemanticAccess – a set of tools to access both relational and semantic databases, which is the focus of our discussion in this paper.

2. System Architecture

SemanticAccess is a set of tools developed for accessing semantic and relational databases using Semantic Binary Object-Oriented Data Model and Semantic SQL query language. It consists of three major components: Query Coordinator, Relational Site and Semantic Site. Figure 1 depicts the overall architecture of the system.

- Query Coordinator: This component is responsible for collecting schemas from different databases and dispatching the users' queries to the appropriate sites. It contains a catalog of schemas stored in a Sem-ODB. This component uses CORBA based architecture for communication and query distribution to other components.
- Relational Site: This component (SemWrap [7]) wraps relational databases to provide a Sem-ODB interface. It contains a knowledge base and a reverse-engineering tool (KDBTool) for schema translation and storage. The relational schema is loaded into the knowledge base and a corresponding semantic schema is generated. This conversion process is a bottom-up methodology similar to the reverse order of conversion described in [5]. The DBA can create complex semantic schema with the use of KDBTool and Knowledge Base thereafter. We used Sem-ODB as the storage medium of the knowledge base in the relational site. Translator module implements a query translation algorithm from Semantic SQL to relational SQL. Currently, this module is capable of wrapping any commercial relational database system, which has an appropriate ODBC driver.
- Semantic Site: This module implements the Semantic Database Engine (Sem-ODB [6]) and Semantic SQL interpreter. Sem-ODB engine is a multi-platform fully functional client-server database system (platforms include Solaris, HPUX, Linux, and various versions of Windows). Clients running on any platform can interact with one or more database servers running on the same or different platforms. Moreover, database files are fully compatible across platforms at binary level. Multiple clients can access server through network protocols such as TCP/IP or NETBIOS while some other clients can run locally as threads within the server process. While the database is suitable for large applications storing terabytes of data, it is also appropriate for small embedded applications because the database engine has very low memory requirements. Its footprint in main memory is about 2 Megabytes including code and auxiliary structures, plus the amount allocated for cache which could be specified by database administrator. A 2 Megabytes cache is enough for a wide class of embedded applications, which means that the database can efficiently run in 4 Megabytes of total memory. The size of the database server executable is about 1Mb.

  In addition to the SQL-level access provided by Semantic SQL interpreter, the database engine (architecture shown in Figure 2) provides a native C++ and Java API for elementary database access, similar to procedural access in an OODB. This is the API which controls three modules that work closely together. ***Vocab*** controls database schema, ***SetQuery*** provides functionality for cursors, and ***Elementary Queries*** module is the main module which provides the functionality of the elementary database access. It uses several logical data types (and correspondingly named modules) to represent data: ***Fact Data***, ***Record Data***, and ***Index***. Furthermore, data is stored using one of three physical data storage types which are ***B-Tree***, ***Bit-Scale***, and ***Raw Data***. All three go through ***Memory manager*** and ***Cache Manager*** to access disk files through the file system. Cache Manager module includes version control and concurrency control to provide complete transaction isolation and optimistic concurrency. Binary data can be stored using ***Parallel Binary Server*** which can be a part of the system or run as a separate process on a different server. It uses several Disk Servers for physical storage, which can use either an underlying file system or raw disks to store data. This server is capable of storing huge amounts of binary data such as pictures and other multimedia data distributed over a TCP/IP network.

SemanticAccess was implemented in C++. The source code for the database engine itself is about 75,000 lines. In addition to that, the source code of SQL server is about 40,000 lines. KDBTool was implemented in VC++ 6.0 with the use of Microsoft Foundation Classes (MFC) for the implementation of graphical user interfaces to interact with the DBA, while the translator module was implemented in C++. The source code of this module is about 45,000 lines. The Query Coordinator was implemented in C++ as well.

3. Demonstration

We will demonstrate Semantic Database Technology emphasizing its advantages. We focus on Semantic Binary Object-Oriented Data Model, Semantic Binary Object-Oriented Database, Semantic SQL query language and Semantic Wrapper for relational databases. Following are some highlights of our demonstration:

- Sem-ODM and database interoperability: SemanticAccess provides Sem-ODM access to both

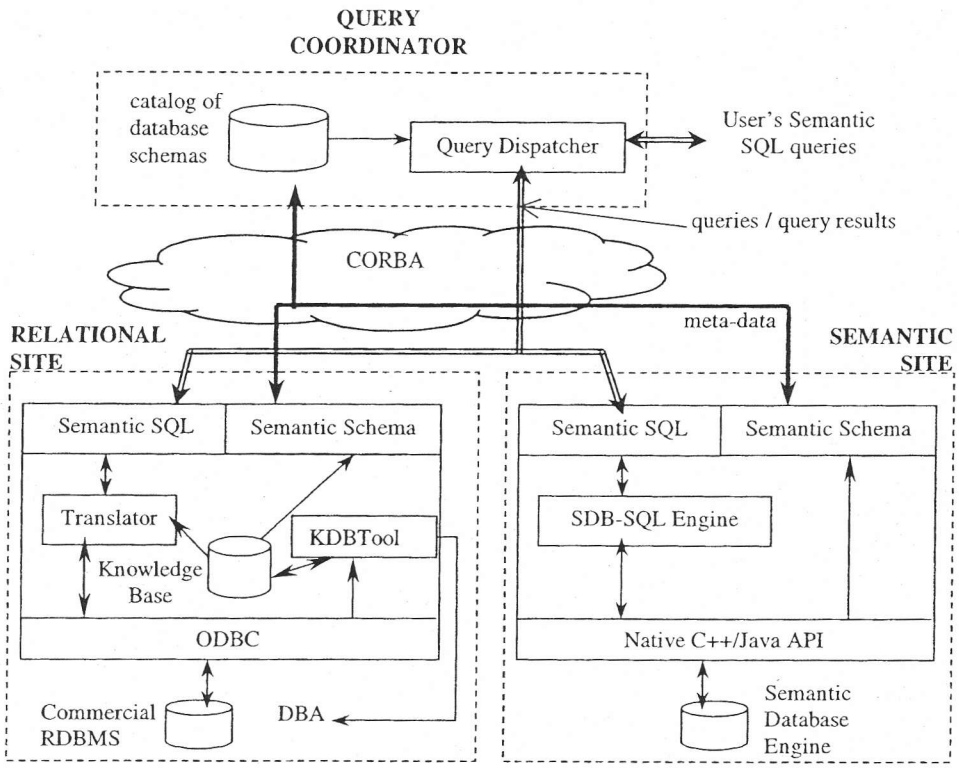


Figure 1. Overall architecture of SemanticAccess

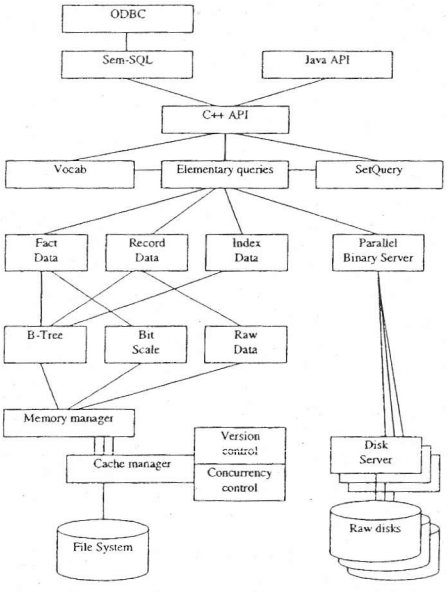


Figure 2. Sem-ODB Engine architecture

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

(a.)

```
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)
```

(b.)

Figure 3. (a.) Semantic SQL query based on a semantic schema (b.) Equivalent relational SQL query for (a.) based on a relational schema

semantic and relational databases. Hence, we can gain the advantages of a more expressive data model than relational data model while providing a well-known query facility, namely SQL.

- Semantic SQL query language: Semantic SQL is SQL language adopted for Sem-ODM. The syntax of Semantic SQL is identical to standard SQL (SQL-92). However, we have extended the semantics of SQL when posing a query to a Sem-ODM schema. The queries are posed on virtual tables, which is determined by examining the query complexity and semantic schema on which the query is posed. Due to the availability of relationships between categories in a semantic schema, explicit joins need not be specified, as in the case of relational schemas. Hence, we have achieved easier and less complex query facilities. A Semantic SQL query based on a semantic schema is shorter and less complex than a relational SQL query posed on an equivalent relational schema. An example is illustrated in Figure 3.
- Sem-ODB and native APIs: We will demonstrate Sem-ODB including its expressive data model capabilities such as inheritance, relations (1:m relations, m:m relations, multi-valued attributes), capability of objects to belong to multiple categories and other features (including native APIs that provides basic database access). Also, Semantic SQL interpreter for Sem-ODB will be demonstrated.
- Semantic Wrapper: We will demonstrate the KDBTool module which is used in creating complex semantic schemas (including inheritance, m:m relations and multi-valued attributes, which is not inherent in the relational schema). The Translator module will be demonstrated with a comparison of Semantic SQL queries and the corresponding translated relational SQL queries. Semantic Wrapper module preserves database autonomy. That is, installation of the wrapper does not effect the existing relational database or its applications. Thus, new applications can be built on top of the wrapper, which provides a more expressive data model and easier query facilities, without affecting the existing applications. Also, the wrapper module is easily installable on any existing commercial relational database with an appropriate ODBC Driver.

Sem-ODB technology is suitable for non-traditional database areas such as GIS, multi-media databases, where relational databases are inadequate. Also, the expressive data model, easier query facility and interoperability architecture of SemanticAccess can be exploited in application areas such as federated/multidatabase environments and data-warehousing environments. We plan to extend the SemanticAccess to access semi-structured and unstructured data sources as well.

Acknowledgements

This research was supported in part by NASA (under grants NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830) and NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409).

4. References

[1] Blakeley J., "OQL[C++]:Extending C++ with an Object Query Capability". *Modern Database Systems: The Object Model, Interoperability, and Beyond*, ACM Press, pp.69-88, 1995.

[2] Chen S.-C., N. Rishe, X. Wang, M. Weiss, "A User-Friendly Multimedia System for Querying and Visualizing of Geographic Data". To appear in *The 4th World Multiconference on Systemics, Cybernetics and Informatics*, July, 2000.

[3] Krieger D., T. Andrews, "C++ Bindings to an Object Database". *Modern Database Systems: The Object Model, Interoperability, and Beyond*, ACM Press, pp. 89-107, 1995.

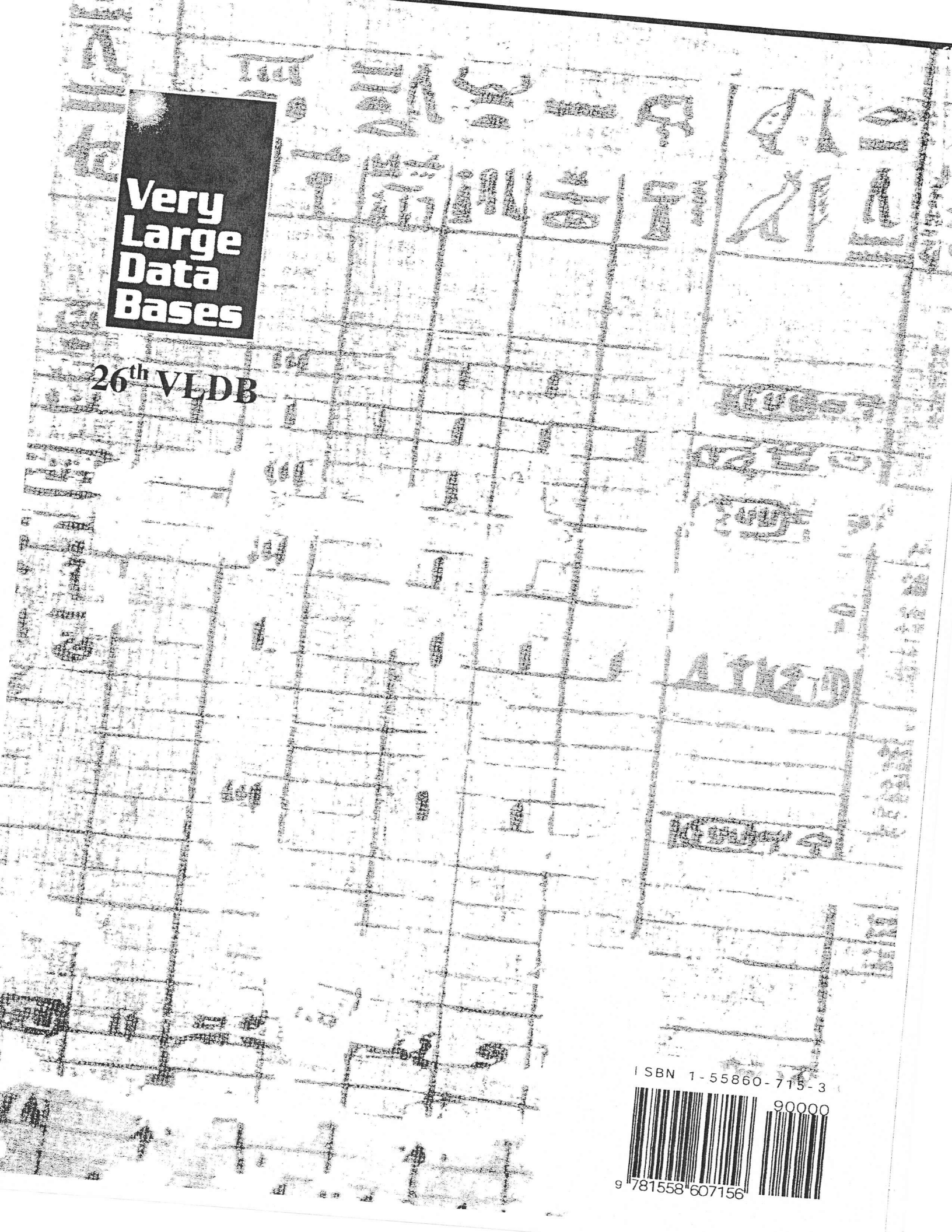
[4] Rishe N., *Database Design: The Semantic Modeling Approach*, McGraw-Hill, 1992.

[5] Rishe N., "A Methodology and Tool for Top-down Relational Database Design". *Data and Knowledge Engineering*, Vol. 10, pp 259-291, 1993.

[6] Rishe N., A. Vaschillo, D. Vasilevsky, A. Shaposhnikov, S.-C. Chen, "A Benchmarking Technique for DBMS`s with Advanced Data Models". To appear in *ACM SIGMOD ADBIS-DASFAA Symposium on Advances in Databases and Information Systems*, September, 2000.

[7] Rishe N., J. Yuan, R. Athauda, X. Lu, X. Ma, "SemWrap: A Semantic Wrapper over Relational Databases, with Substantial Size Reduction of User's SQL Queries". In *Proceedings of the 7th International Conference on Extending Database Technology - Software Demonstrations Track*, pp. 13-14, March, 2000.

[8] Rishe N., W. Sun, D. Barton, Y. Deng, C. Orji, M. Alexopoulos, L. Loureiro, C. Ordonez, M. Sanchez, A. Shaposhnikov, "Florida International University High Performance Database Research Center". *SIGMOD Record*, Vol. 24, No. 3, pp. 71-76, 1995.



ISBN 1-55860-715-3