ICES CM 2006/M:13

An object-based approach to integration of software to support management and reporting of marine ecosystem survey data.

Marek Ostrowski

An object-based software to support distributed databases and data reporting from combined fisheries and environmental surveys is presented. The key abstraction of this system is expressed as a generic station (data) object. The station object consists of a nested data structure, to store a master record and a matrix of data cycles, and a behavior to capture this data structure from a variety of input data formats. The station objects are not a part of the main system, but are implemented as separate runtime plug-ins or user-defined with XML scripts. This makes this system easily adaptable to particular data collection needs of a given survey.

Once the data are stored in the database, they are accessible through a set of generic data protocols. Through these protocols, implemented with the COM technology, the data are easily integrated with the user-end software on Windows, including GIS and numerical computing environments.

Keywords: Marine Data Management, Object-based technology

Contact author:
Marek Ostrowski: Institute of Marine Research, P.O. Box 1870
5817 Bergen, Norway
tel: + 47 55 23 86 23
e-mail: mareko@imr.no

1. Introduction

Relational database management systems (RDMBSs) have seen the widespread acceptance in management of marine survey data. However, the RDMBS model best support traditional business applications, such as order processing, banking, billing and online reservations. It can be a costly and slow solution, if applied to complex science-derived data structures, such as satellite imagery or high-resolution marine datasets, because of the fragmentation of 'real world' entities into many relations as required to meet the physical data representations in the relational database (Connolloy et al., 1999). For such data types, data management may by more effective using an object data management system (ODBMS). At the most basic level, an ODBMS may be regarded as an extension to file systems to store manage and retrieve a collection complex data structures in one piece, without a need to remap these data into relations. This offers benefits in terms of increased speed of access and reduced complexity of interfacing to the data. In the research environment, these features are of a particular value for the two groups of users: applications programmers who write applications for in-house use and data analysts who access the banked data through standard statistical/ data analysis/GIS software. As the modern ODBMS architectures fully support network transparency and client/server model in the same level as RDMBS, using these systems does not impair in any way the collaborative use of data by organizations. A comprehensive though somewhat dated review of ODBMS technologies may be found in (Cattell, 1994). To learn the basics of the object-oriented data modeling, the reader is referred to (Booch, 1994).

This report introduces a reference implementation of an object-database, called *QuickSurvey* (QS), which the author had developed incrementally in the years 1999-2003 to support immediate needs of a small survey team responsible, among others, for long-term stewardship of diverse physical and resource data collected during routine marine ecosystem surveys of the Nansen Programme (Sætersdal et al., 1999). The focus was on data types, which due to software/staff limitations were not used beyond the immediate survey reporting objectives and were only preserved in their originators formats, on tapes and CDROMs. These data included: underway

weather/sea surface log, post-processed ADCP profiles, acoustic abundance from echo integration, thermosalinograph data and chemical sampling. The QS databases were populated from raw files generated automatically by various data logging systems, from spreadsheets containing data entered manually, and sometimes from outputs of various data reduction and quality assurance procedures carried out by means in house developed software and scripts. Some of these files adhered to the international and institutional standards, many others less so. The use of QS to bank these data was experimental. Many datasets from the described period are still available only as offline archives on CDROMs.

The data preserved with QS are much easier to access and use than those preserved in flat files. QS provides data online, have a simple navigational interface to locate surveys and various data types, produces subsets based on time, geographical box and various station's header criteria, does not depend on availability of the originators firmware to access raw data, and provides a live access to the data from within the scripting environments most frequently used in our laboratory (Excel, IDL).

2. Overview of the system

QS is a Windows-based software. Its task is to support management of marine ecosystem survey data. The design of this software had two main goals: (1) to ensure a functional model (software and operational procedures) to support a small research team or institution to manage their data, and (2) provide an opened environment for integration with end-user software, that is commonly used for survey reporting and data-driven retrospective analysis.

2.1 The functional description of the system.

The functional diagram of the system is depicted in Figure 1. The system is used in the following three domains: on research vessel, in an associated land-based laboratory and is amenable for use from the Web. The aim of the ship-based

subsystem is to capture new data to retrieve the already banked data to support preparation of survey reports. The source data are generated by variety of sampling and postpocessing operations aboard the vessel: automated log from underway instruments, vertical casts and sampling at stations, or outputs from data reduction and quality control procedures. Node **A** symbolizes the sources of onboard survey data. Node **B** denotes the archive of flat files generated by these sources during a survey. If QS is used, most of the data in **B** is subsequently uploaded to the onboard database (Node **C**).

The data stored in the onboard database that are retrieved for survey reporting purposes are denoted by Node **D**. After the survey completes, the updated database is brought to the land-based laboratory (Node **E**), where it is deployed on a designated server (Node **F** or **G**).

The survey-collected databases can be maintained on a single central server or they can be distributed among several computers in the laboratory. The later choice is represented in Figure 1, where Nodes **F** and **G** host different databases, perhaps maintained by different research groups. The QS system serves small organizations. Fellow researches, rather than database professionals, are responsible for hosting databases related to their work. They may choose to keep the data on their own workstations and be responsible their administration. This is the situation symbolized by Nodes **F** and **G**.

Nevertheless, the data from all servers in the laboratory are available to all users. Nodes **J** and **I** depict client machines, which access data from the databases located on various servers. To assure the laboratory wide data integration, an XML registry of serves and databases placed on the lab's web server (Node **H**). This registry is used by QS on the client machines to locate servers on the local net.

The QS system envisions a distribution of data to web based clients (Node **K**). For security reasons a direct access to the database servers is not permitted. The data are sent by means of compressed HDF files (NCSA, 2005). The QS software on the web-

based client machine accesses the sent data in the same way, as if they were located in an online database.

2.2 The system's architecture

The QS system is characterized by four distinct components: the Physical Data Store (PDS), which handles low-level disk and networking operations; Object Abstraction Layer (OAL), which defines a high-level interface used for scripting and writing applications, Application Layer (AL) containing the user-level applications integrated with the base system, and Pool of Station Schemas (PSS), which contain modules implementing schemas of concrete ocean data types (Figure 2). PSS is linked to OAL dynamically, so new data schemas can be added once the system has been deployed.

2.2.1 The Object abstraction layer (OAL)

The central element of this architecture is the Object Abstraction Layer. Implemented with the Microsoft's Component Object Model (Orfali et al., 1996), it defines a high level object interface through which the client software communicates with the database. A great care has been taken to design that interface in such a way that it hides from its user idiosyncrasies of the low-level database handling technologies. Its syntax is concise. Using few lines of scripting code, the users gains access do distributed data archives. For instance, the following Visual Basic script snippet is used to accesses the first ADCP profile from a dataset that has been collected between 21º and 23ºN during survey in 2003, stored in a database named "HYDRO":

Listing 1: Code example showing an access ADCP data from a remote database.

```
Set stations = DataBank.Repository("HYDRO").Datasets("ADCP_2003411")
stations.Open, "LATITUDE > 21.0 and LATITUDE < 23.0"
Set station = stations.CreateStation
stations.read station
dataCycles = station.DataBook.Sheets("BT_VELOCITY")
```

The key abstraction of the QS system is a generic data object, called *Station*. This Station object consists of a data structure to store a master record and a matrix of data

cycles, and a behavior to capture this structure from a variety of input data formats. The schema of the Station object is shown in Figure 3. This schema is very generic to accommodate many types of marine data: vertical profiles, sections of underway recordings or time-series of moored instruments. It has a nested structure: the sampling event level data are stored in the *MasterRecord* object while data cycles are placed into one or more *DataSheets*. The concrete implementation of these structure are not a part of the core system, but are dynamically linked to it by means of the runtime plug-ins (Figure 2). The dynamic linking of schemas makes this system easy adaptable to new types of ocean data categories. All that is required is to devise a new plug-in for a new data category stemming from a code template provided with a new system and the placement of that module in the PSS pool.

Another way to extend QS for a new ocean data category is providing its schema in XML. This is a much easier method to handle by non-programmers, it but does not include behaviors, which tell the system how to read the data from external files. However, often the source data exists or can be exported to an Excel spreadsheet or other software that supports scripting. In such cases, the data can be uploaded to the database using a user script within such a program. Listing 2 demonstrates an XML schema for a nutrient dataset, which was entered manually to a spreadsheet. Figure 4 demonstrates a view of same schema, compiled and merged with the dictionary of existing Station schemas.

QS maintains access to the data at four hierarchical levels. The users access these levels through navigation, similarly to accessing local disk files. The Station object is at the bottom of this hierarchy. On the second level, all Stations of the same data category are grouped into a collection termed a *Station-set*. On the third level, all station-sets for all data types located in a database are contained in a *Repository*. Finally, at the top level there is a *DataBank*, which consists all *Repositories* registered for use on a local network. This four level hierarchy is fully reflected in the scripting mode (Listing 1) and in the QS user interface (Figure 6).

2.2.2 The Physical Data Store (PDS)

At the physical level QS uses two independent software engines: The FirebirdSQL RDMBS (Firebird, 2006) and Hierarchical Data Format library version 5 (HDF5), (NCSA, 2005).

Despite of the RDMBS engine used, the QS system does not use relational model. The basic building blocks of that model: the record and table are utilized to hold the principal QS objects the Station and Station-set. The arrays, which Station object uses to store data cycles, are mapped into Binary Large Objects (BLOBS) held together with non-array master record fields in the same physical record of the relational database.

The mapping of the QS data structures to the HDF5 format is direct. The HDF stored data can be recovered using standard tools on the NCSA website. However, no data selection capability exists for data objects held in these files. Due to a better performance and compactness, this form of storage is best suited to create temporary data subsets downloaded from the main database for an offline data analysis. These files are also used for transfer of data subsets across the web.

2.2.3 The Application Layer

The application layer consists of the following modules: Database administration software, scripting interfaces to standard computing environments and ActiveX controls for building the database centric applications. The database administration software is distributed with the base system. It includes tools for definition of new Station objects with XML, for creation and maintenance of station-sets and for generic data access. These programs have basic functionality. Currently, this module consists of three such programs depicted in Figures 5 and 6.

The scripting interface is the essential feature of the QS system. The users use QS from their preferred software interactively or by writing short programs. Many industry-standard programs support COM scripting. Excel is one such platform. Other examples include: ArcGIS (Razavi, 2006), and Statistica (StatSoft, 2006). An interface to access QS-stored data using Interactive Data Language (IDL), (ITT, 2006) is provided with the basic distribution of the QS system.

QS supports ActiveX Controls for building data-centric applications. These are visual components that can be inserted in other applications to extend their functionality (Microsoft, 2006). The QS ActiveX provides components for database navigation and access. These have been applied to derive the database administration tools. A complete CTD database and quality control system for use in the Nansen Programme (Ostrowski, 2006) has been implemented with en early version of QS.

3. Conclusions

The paper touched on the issue of integration of various pieces of software to support management and reporting of marine data. A custom developed piece of software called *QuickSurvey* (QS) was presented. This software addressed one fragment of activities connected to data management and reporting: that of preserving survey data of diverse categories, storing them under the hood of a single database management system, and distributing online to the users from the parent laboratory and on the Web.

The application of the object-based database concept proved to be advantageous to the task at hand. A generic station (data) object had a potential to accommodate a broad spectrum of survey data categories such as vertical profiles, underway recording or time-series. The concrete implementations of that object were derived in a computer code or using XML. The changes to the object schemas did not affect the data model of the underlying database engine; and hence QS could be used without modifications to the core system for banking and online distribution of very diverse ocean data types, not envisioned during the system's design phase.

The application of the Common Object Model technology opened QS for interoperation with the industry standard tools used for survey reporting, such as GIS and numerical computing environments. It is suggested that once QS is used according presented functional model, the users of those programs will be able to gain access to life data for very diverse data categories from the distributed data archives maintained in their organization.

REFERENCES

Anonymous, 2006. Report on The Joint Research Between Indonesia and Norway on the Earthquakes and Tsunami Impacts in Aceh and west Sumatra., LIPI, Jakarta.

Booch, G., 1994. Object-Oriented Analysis and Design. The Benjamin/Cummings Series in Object-Oriented Software Engineering. The Benjamin/Cummings Publishing Company, Inc., 589 pp.

Cattell, R. G., 1994. Object Data Management : Object-Oriented and extended Relational Database Systems. Addison-Wesley, 389 pp.

Connolloy, T. M., Begg, C. E., Strachan, A., 1999. Database Systems. A Practical Approach to Design, Implementation, and Management. Addison-Wesley, 1094 pp.

Firebird Project, (2006, August 15), Firebird - Relational Database for the New Millennium, [WWW Document], See http://www.firebirdsql.org/

ITT Industries Inc., (2006, August 15), IDL The Data Visualization and Analysis Platform, [WWW Document], See http://www.ittvis.com/idl/

Microsoft Corporation, (2006, August 15), Description of ActiveX Technologies, [WWW Document], See http://support.microsoft.com/kb/154544/EN-US/

The National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, (2005, October 31), HDF5 - A New Generation of HDF, [WWW Document], See http://hdf.ncsa.uiuc.edu/HDF5/doc/

Orfali, R., Harkey, D., Edwards, J., 1996. The Essential Distributed Objects Survival Guide. John Willey & Sons, Inc, 604 pp.

Institute of Marine Research, Status of Hydrographic data with the Nansen Programme, [WWW Document], See http://mareko.net/SM/QC.pdf

Razavi, A. H., 2006. ArcGIS Developer's Guide For VBA. Onward Press, 188 pp.

Sætersdal, G., Bianchi, G., Strømme, T., 1999. The Dr. Fridtjof Nansen Programme 1975-1993. FAO Fisheries Technical Paper 391. FAO, Rome, pp. 434.

StatSoft, (2006, August 15), Statistica Visual Basic, [WWW Document], See http://www.statsoft.com/uniquefeatures/visualbasic.html

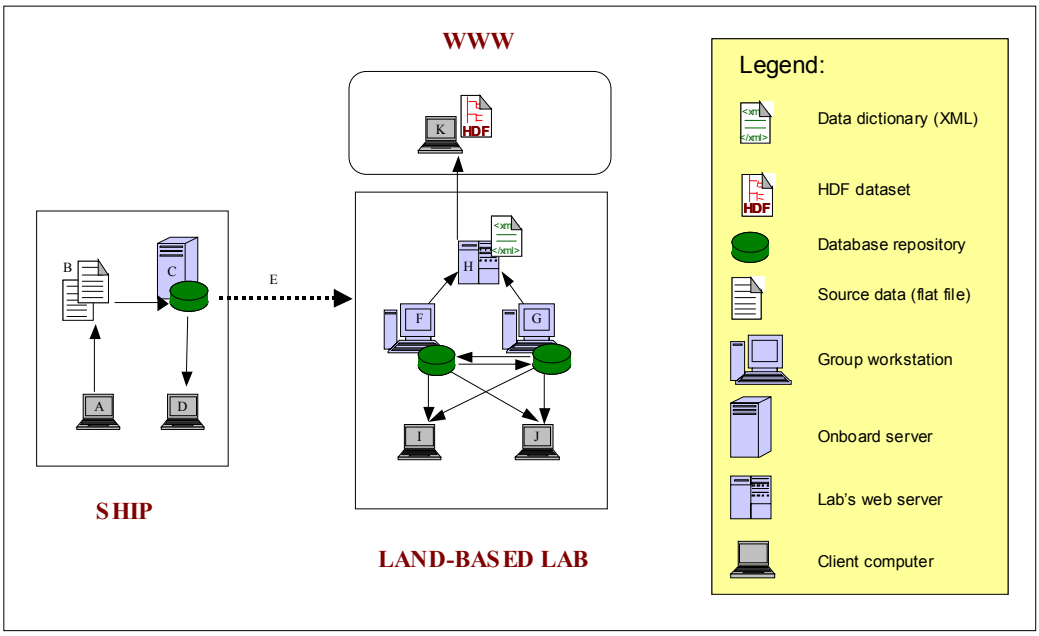

Figure 1. Domains of operations the *QuickSurvey* system.

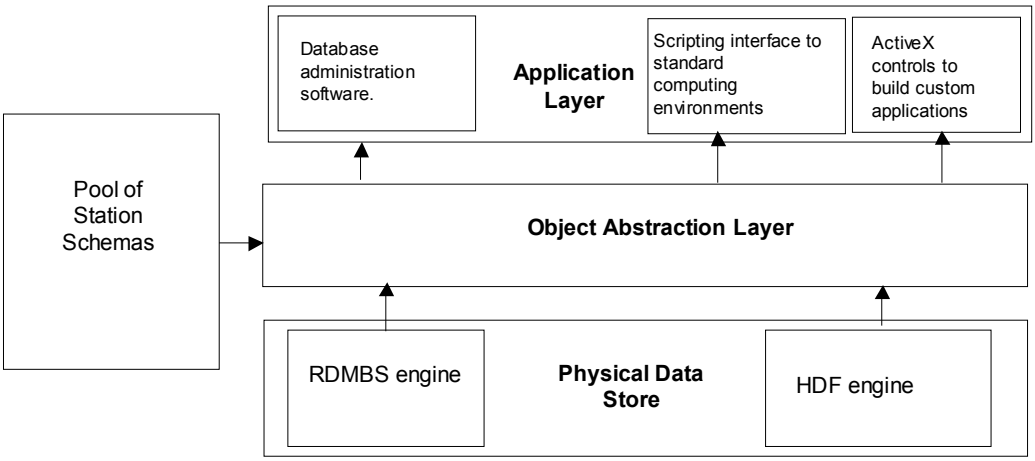

Figure 2. The architecture of the QuickSurvey object database.

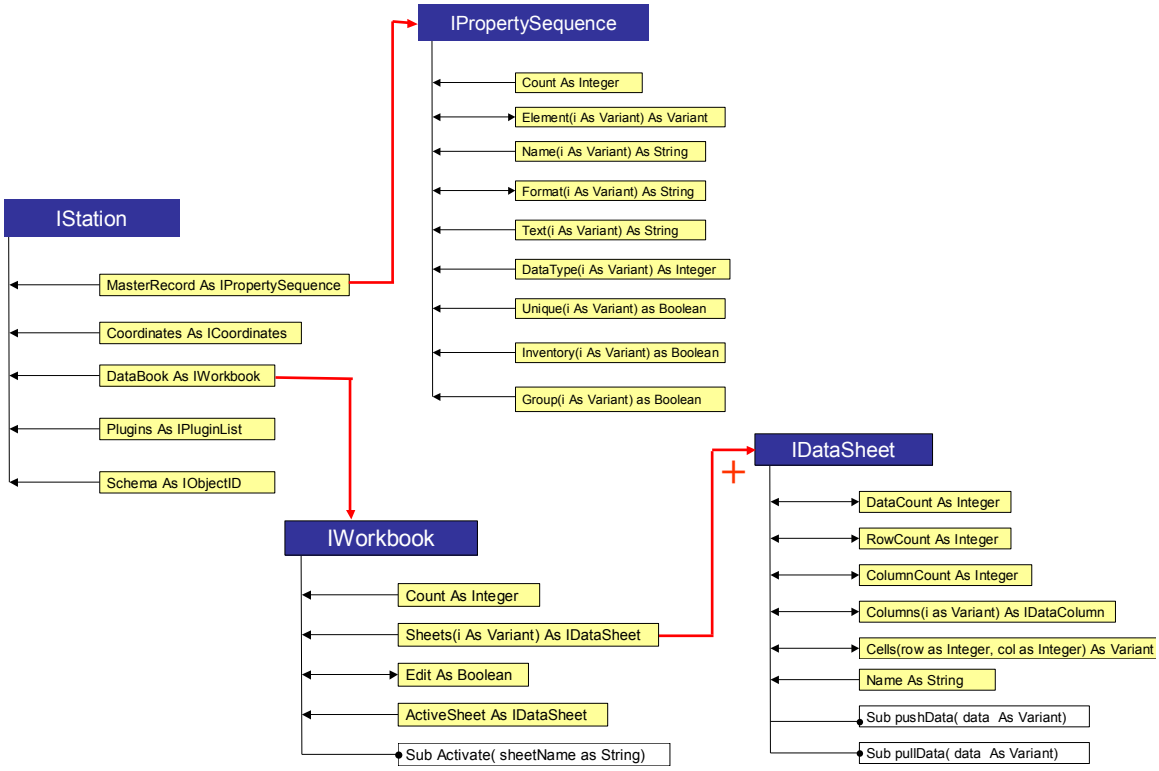

Figure 3. The *QuickSurvey* Generic Station Object expressed by set of COM interfaces. The red arrows point from the owner to subordinate objects. The "+" sign denotes a one-to-many ownership. The black arrows denote object's properties: the double and left pointed arrows denote the read/write and read-only properties, respectively.

Table 1. First three stations from nutrient dataset off Western Sumatra onboard R/V Baruna Jaya VIII in August 2005 (Anonymous, 2006).

Depth	Lat	Long	Time	pH	O2, ml/L	NO3, µg A/L	PO4, µg A/L	SiO3, µg A/L	TOM, % w/w	Date
0	05.18.431	96.40.037	10	8.08	4.18	0.99	0.61	3.77		1/8/2005
25				8.1	4.12	0.54	0.52	4.22		
50										
75				8.13	3.84	2.77	0.43	9.52		
100				8.16	3.02	7.24	0.7	28.19		
200				8.17	1.09	24.37	2	33.13		
300				8.22	0.86	25.9	2.26	36.36		
400				8.28	0.84	27.07	0.83	37.7		
0	05.18.422	96.17.939	15.05	8.02	4.26	1.05	0.22	5.57		1/8/2005
25				8.04	4.17	0.99	0.3	4.94		
50				8.06	4.03	1.01	0.35	4.13		
70				8.09	3.9	2.04	0.26	5.3		
0	05.30.120	96.00.033	19.25	8.01	4.2	0.39	0.22	4.58		1/8/2005
25				8.03	4.18	0.35	0.22	4.85		
50				8.05	4.16	1.63	0.52	3.77		
75				8.09	3.63	2.14	1.26	4.76		
100				8.1	2.68	9.75	2.26	11.04		
200				8.09	0.87	23.77	2.48	26.57		
300				8.1	0.84	25.82	2.48	32.14		
400				8.1	0.82	26.08	3.09	35.46		
500				8.11	0.77	26.93	2.91	41.11		

Listing 2. Station object schema of the nutrient dataset shown in Table 1.

```
<?xml version="1.0" ?>
- <Schema ID="CHEM_BJVIII" VER="1.0">
  - <Header>
      <Description>Nutrients BJVIII</Description>
      <Creator>Marek Ostrowski</Creator>
      <Date.Stamp>20050820</Date.Stamp>
      <Comment>A schema for nutrient data analyzed
          on Baruna Jaya VIII off Sumatra in August 2005.
      </Comment>
    </Header>
  - <!--
    This is the master record
    -->
  - <Master>
    - <Field ID="SURVEY_NO" TYPE="INT">
        <Unique />
        <Summary />
      </Field>
    - <Field ID="DATE_TIME" TYPE="DATE">
        <Inventory />
        <Unique />
        <Summary />
      </Field>
    - <Field ID="SHIP_CODE" TYPE="INT">
```

```
        <Unique />
      </Field>
      <Field ID="COUNTRY_CODE" TYPE="INT" />
    - <Field ID="PROJECT_ID" TYPE="STRING">
        <Summary />
      </Field>
    - <Field ID="LATITUDE" TYPE="LAT">
        <Inventory />
        <Summary />
      </Field>
    - <Field ID="LONGITUDE" TYPE="LON">
        <Inventory />
        <Summary />
      </Field>
    - <Field ID="STATION_NO" TYPE="INT">
        <Inventory />
      </Field>
    - <Field ID="BOTTOM_DEPTH" TYPE="FLOAT">
        <Inventory />
        <Format DEF="%4.0f" />
      </Field>
  </Master>
- <!--
  These are mandatory mappings for a profile
  -->
- <Coordinates>
    <Station REF="STATION_NO" />
    <Latitude REF="LATITUDE" />
    <Longitude REF="LONGITUDE" />
    <Date REF="DATE_TIME" />
    <Depth REF="BOTTOM_DEPTH" />
  </Coordinates>
- <!--
  These are the data cycles to the dababase
  -->
- <Sheets>
  - <Sheet ID="DATA">
      <Column ID="DEPTH" FORMAT="%6.3f" />
      <Column ID="PH" FORMAT="%6.3f" />
      <Column ID="O2" FORMAT="%6.3f" />
      <Column ID="NO3" FORMAT="%6.3f" />
      <Column ID="PO4" FORMAT="%6.3f" />
      <Column ID="SiO3" FORMAT="%6.3f" />
    </Sheet>
  </Sheets>
    - <!--
      These is a read-only version of the data cycles
      in the database. In this simple case these are
       the same as the Sheets
  -->
- <Views>
  - <View ID="DATA">
      <ColumnRef ID="DEPTH" />
      <ColumnRef ID="PH" />
      <ColumnRef ID="O2" />
      <ColumnRef ID="NO3" />
      <ColumnRef ID="PO4" />
      <ColumnRef ID="SiO3" />
    </View>
  </Views>
</Schema>
```

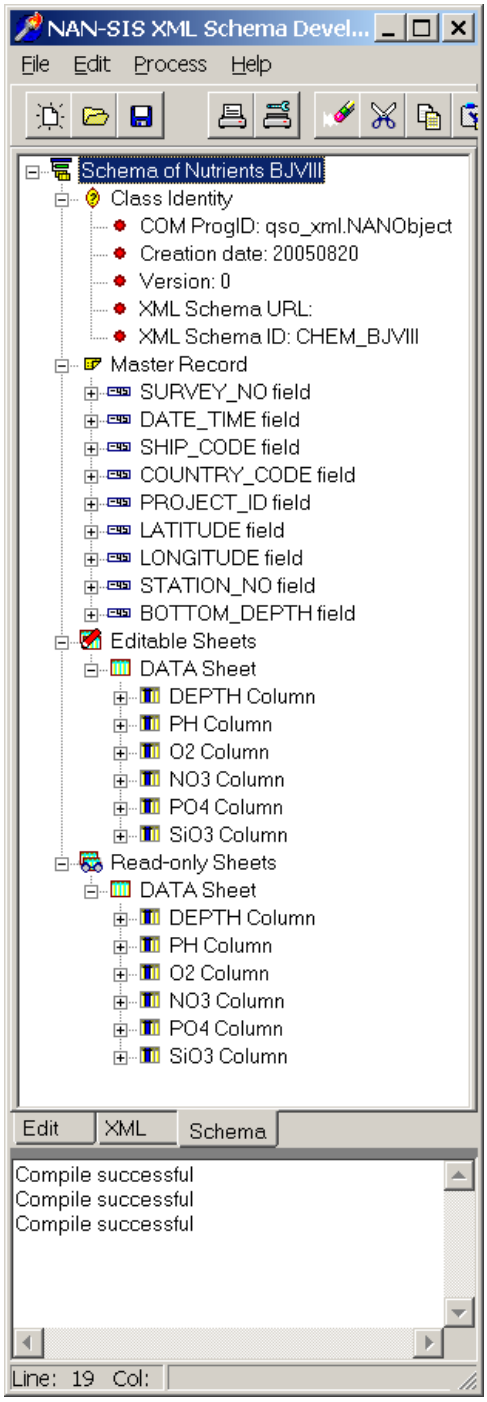

Figure 4. Visual representation the station schema from Listing 1 inside of window of the schema definition program distributed with the QS system..

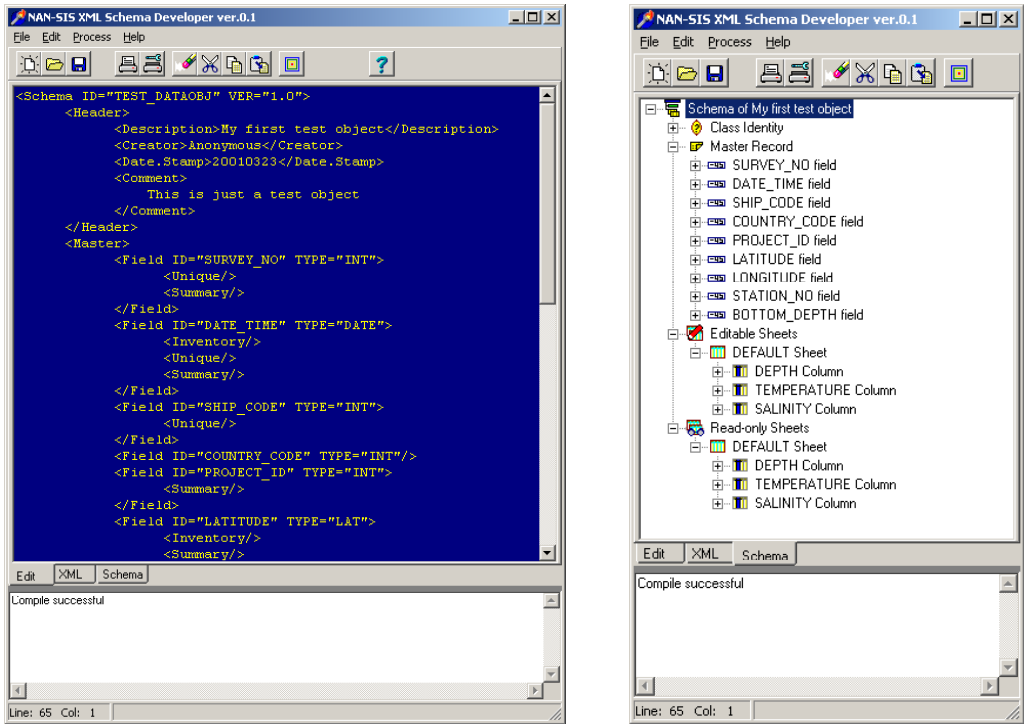

Figure 5. Screen shot from the application to define schemas of the stations objects storable to QS databases. The panel to the left displays the editor used to define an XML schema for a new station object; that to the right shows the resulting schema after compilation and registration with the system.

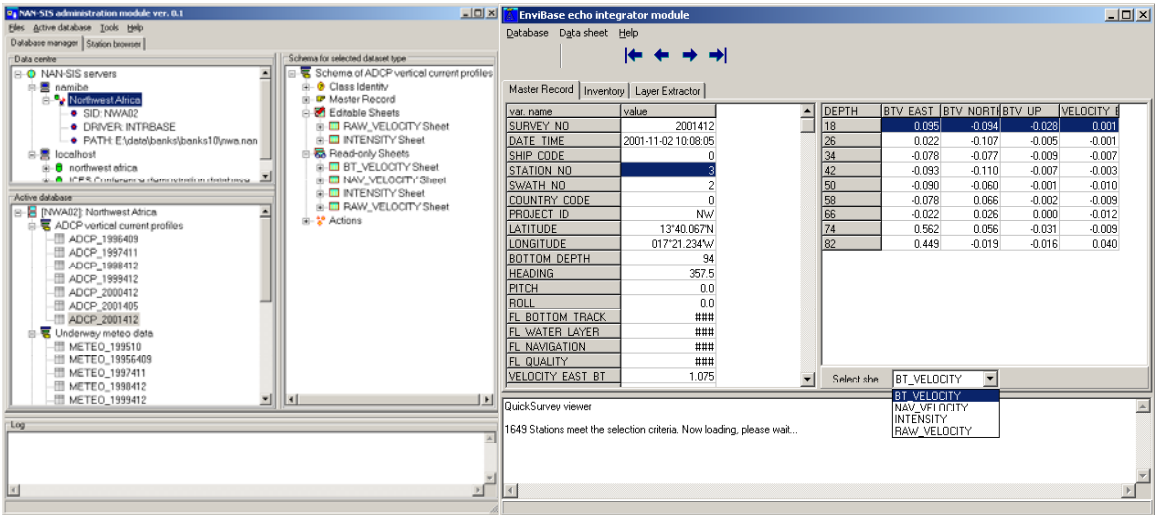

Figure 6. Two principal applications for database administration and data access. **To the left:** the database administrator program showing lists of repositories (top-left) and station-sets for different data types (bottom left). **To the right**: the basic data exploration program displays a view showing the data for a selected station. Other views include inventory of stations for the opened station-set, and a sheet to extract horizontal layers from vertically profiled data.