2001-01-01

# E-Spatial Technology for Spatial Analysis and Decision making in Web-based Land Information Management Systems

James Carswell
*Technological University Dublin*, james.carswell@tudublin.ie



## Recommended Citation

Carswell, J. (2001) E-Spatial technology for spatial analysis and decision making in web-based land information management systems. *Journal of Geographic Information and Decision Analysis,* Volume 5(2); 2001.

GIDA

*Journal of Geographic Information and Decision Analysis*
*2001, Vol. 5, No. 2, pp. 95-114*

# *e*-Spatial™ Technology for Spatial Analysis and Decision Making in Web-Based Land Information Management Systems

**Michela Bertolotto**
Department of Computer Science
University College Dublin
Ireland
michela.bertolotto@ucd.ie

**James D. Carswell**
Digital Media Centre
Dublin Institute of Technology
Ireland
jcarswell@dit.ie

**Liam McGeown**
e-Spatial Solutions Limited
Dublin, Ireland
info@e-spatialsolutions.com

**John McMahon**
e-Spatial Solutions Limited
Dublin, Ireland
info@e-spatialsolutions.com

**ABSTRACT** In this paper we describe e-Spatial™ technology, comprising an innovative software solution for deploying integrated web-based spatial applications within Oracle 9i Database environments. The two main components of the solution are the iSMART™ database development technology and the i-Spatial™ Information Server (iSIS). This technology allows Oracle users to build and deploy both spatially enabled and standard Internet applications without requiring any application-specific GIS source code. It can be deployed on any Oracle supported hardware platform and on any device running a Java Virtual Machine, thus providing full support for wireless, PDAs and other mobile devices. In particular, we describe a land information management system developed using e-Spatial™ technology and its effectiveness as a spatial analysis and decision-making system.

**KEYWORDS:** Oracle Spatial Database, Web-based GIS, Land Information Management Systems, Spatial analysis

**Acknowledgements** The authors wish to thank the Irish Department of Agriculture for their cooperation and input into the preparation of this paper.

## 1. Introduction

Traditional GIS applications (East et al. 2001; URL1; URL2; URL3; URL4) store spatial and non-spatial data attributes separately. Typically spatial properties of objects are stored in files that are managed by a file management system while attribute data is stored in a commercial database (e.g., a relational database). This split design presents the difficulty of maintaining data integrity between spatial and attribute data as the two datatypes are not managed by the same engine. Oracle Spatial provides the information infrastructure that includes a single database system for managing both types of data, together with a data structure that is independent of the particular application.

An important issue in handling spatial data is the preservation of topological consistency: spatial data must be topologically consistent in order to be usable. Oracle Spatial guarantees the

 ISSN 1480-8943

consistency of topological relationships between spatial entities (Egenhofer and Franzosa 1991) by automatically updating the topology of a dataset when existing elements are modified or new elements are introduced and by verifying whether inconsistencies have been generated. Topological queries (e.g. are two objects disjoint or overlapping) as well as other types of spatial queries (e.g. range/window queries, spatial joins, etc.) can be performed by using predefined spatial operations and functions. However, in order for this core functionality to be fully exploited, it must be integrated into a more sophisticated technology that allows for real-time spatial data display, collection, editing, manipulation, and query (e.g. using standard web browsers). *e*-Spatial™ technology extends an Oracle Spatial database by allowing users to exploit this range of advanced spatial data handling functions that are typically provided by specialized GIS application packages. Within this technology, all spatial analysis functions are implemented as Java stored procedures. Therefore, in contrast with other GIS applications, this technology allows one to build and deploy spatially enabled Internet solutions without requiring any application specific source code. *e*-Spatial™ technology is ideally suited for interactive real-time analysis and decision making in applications such as utility and government mapping, navigation systems, and the emerging GPS enabled mobile computing applications, e.g. for cultural heritage interfaces (Carswell et al. 2002).

In the past spatial information was utilized within specific applications and exclusively by expert high-end users. More recently, thanks to the diffusion of desktop GIS and the Internet, its integration within the widest range of information systems is becoming a common requirement and therefore non-expert users (with no specific programming, GIS, or database management background) are now required to visualize, query and manipulate spatial data. These users must be provided with an easy-to-use environment to work with their spatial applications. To this purpose, additional tools and user-friendly environments to build and customize graphic user interfaces that facilitate interaction with the *e*-Spatial™ basic platform have also been developed.

Another important concern for the design and development of contemporary and next-generation (web-based and mobile) information systems relates to interoperability issues. Interoperability refers to the capability of autonomous systems to exchange data and to handle processing requests by means of a common understanding of data and requests (Sondheim et al. 1999). In GIS, data modeling is an important issue within the context of interoperability. Agreement at the representation level is essential for exchanging spatial data. Specifications on the conceptualization of spatial entities and the space/time reference systems where they reside have been provided by the OpenGIS Consortium (URL5; URL6). The data model used by *e*-Spatial™ technology is the Oracle Spatial object-relational model. Such a model conforms to interoperability and standardization requirements as it corresponds to the "SQL with Geometry Types" implementation of spatial feature tables described in the OpenGIS ODBC/SQL specification for geospatial features.

*e*-Spatial™ technology has been utilized for the realization of a Land Information Management System (LIMS) application, developed for the Irish Department of Agriculture. This system delivers a spatially enabled Internet solution for tracking and managing land information based on land usage, land classification and land ownership changes over time. It allows for a wide variety of spatial analysis functionality and therefore provides an effective decision making tool. An important characteristic of this system is that it is used not only by the Department of Agriculture officers but also by non-expert users, i.e. the Irish farmers. Therefore the graphical interface to the underlying spatial data and analysis functionality was developed with particular attention to user friendliness to facilitate human computer interaction. A detailed description of the complete LIMS is presented in this paper.

The remainder of this paper is organized as follows. Section 2 describes the architecture of *e*-Spatial™ technology, including its main advantages (e.g., scalability and reliability). Section 3 gives an overview of the additional tools that allow users to build and customize their applications using this

technology. Section 4 describes the functionality of the LIMS as well as some operational results. Finally, Section 5 presents our concluding remarks and further issues that are currently under investigation.

## 2. Architecture components

In this section we describe the two major components of *e*-Spatial™ technology, namely the *i*SMART™ database development technology and the *i*-Spatial™ Information Server (*i*SIS), together with their architectural characteristics (Bertolotto et al. 2001). *i*SMART™ is an intelligent Oracle application database development technology that allows database administrators and software engineers to design, build and deploy spatially enabled (as well as standard) Internet applications without requiring any proprietary or application specific source code. It therefore allows reduction of overall system development and deployment costs. All *i*SMART™ applications are automatically built in real-time directly from the Oracle database and can be used and modified on-line without having to log-off existing Internet applications or writing any client application code.

The *i*-Spatial™ Information Server is a Java plug-in component of *i*SMART™. It consists of a database management application program that adds spatial functionality to an Oracle Spatial database.

### 2.1 *i*SMART™ architecture

The *i*SMART™ development environment relies on a three-tier architecture comprising three main layers, namely the *Client Layer*, the *Application Server Layer*, and the *Database Layer* (see Figure 1).

All communications between the client layer and the database are conducted through the application server layer. The application is executed on the client using an applet that runs in a standard web browser's Java Virtual Machine (JVM). The applet communicates with the application server using the existing HTTP or RMI networking protocols.

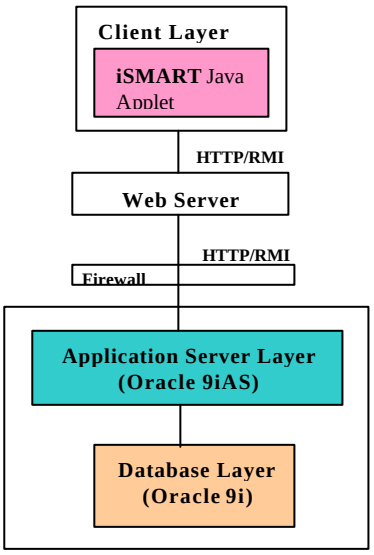


**Figure 1**. The *i*SMART™ architecture components.

**2.1.1. Client layer.** The client layer is a light-weight client machine running a Java Virtual Machine. The *i*SMART™ Java applet is a micro thin applet that executes all commands received from a Panel EJB. Upon initialisation the *i*SMART™ applet asks for a username, password, application server URL and port number. The connection information can be defined as startup parameters for the applet to prevent the login dialog from appearing. For security reasons, however, this startup method is not recommended.

**2.1.2. Application server layer.** The application server currently used by *i*SMART™ is Oracle 9iAS. This layer contains several Enterprise Java Beans (EJBs) illustrated in Figure 2 and described in the following.

The *i*SMART™ session EJB is responsible for all communication with the client. As it is a stateful session EJB, an object of this type is instantiated for each user session. This EJB validates all data submitted by the *i*SMART™ applet before passing it to the entity beans.

Panel EJBs are entity beans that, when initialized by the *i*SMART™ EJB, query the *i*SMART™ Application Specific Metadata for all information about the relevant panels (to be displayed on the client applet) from the database and send information back to the *i*SMART™ EJB which returns it to the client. Each Panel EJB has a hierarchical structure, i.e., it can contain nested sub-panels and objects and contains details only about its objects and immediate sub-panels. Each panel in the user application corresponds to a Panel EJB in the hierarchy. In Figures 3, 4, and 5 an example of panel hierarchy is shown as it appears in the Visual*i* environment described in Section 3.

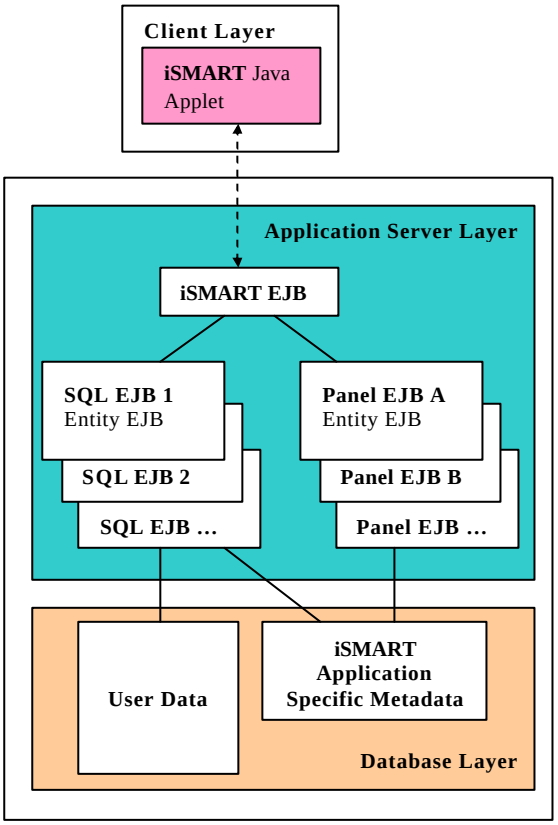


**Figure 2.** Application server layer EJBs and communications.

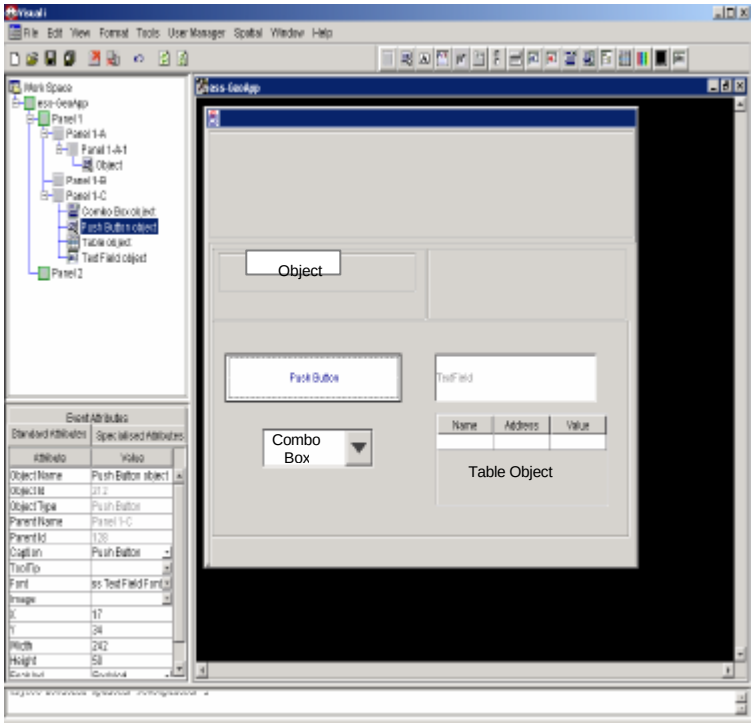


**Figure 3.** Sample user application under construction.

Figure 3 shows a screenshot of the user application. On the right hand side of the screen, the user can build panels in a workspace using drag-and-drop operations (Figure 4); while on the left hand side, the panel hierarchy is automatically maintained and displayed (Figure 5).

SQL statements are needed to retrieve, from the user database, the information to be displayed on the client site. For example, when the user presses a button, some values may need to be inserted or updated in the user database. In this case, an event is associated with that particular button to call an SQL statement. Pre-defined SQL commands are stored in the *i*SMART$^{TM}$ Metadata part of the database. The entity beans responsible for the execution of SQL statements are called SQL EJBs.

When an event is triggered by a user action, the *i*SMART$^{TM}$ EJB retrieves the SQL identifier (id) of the command that is to be executed and instantiates an SQL EJB passing such an id to the corresponding constructor method. The SQL EJB then retrieves the correct command from the *i*SMART$^{TM}$ Metadata table and in turn sends it to the Oracle database engine for execution.

The result of the query is returned to the *i*SMART$^{TM}$ EJB, which communicates it to the client. If the same query is run again, the *i*SMART$^{TM}$ EJB will re-invoke the SQL EJB that was previously instantiated. The parsed version of the previously executed SQL command will be cached by the SQL EJB thereby eliminating the need to re-interpret the query. Bind variables are used to optimize the execution of the SQL query and to assign values to object variables.

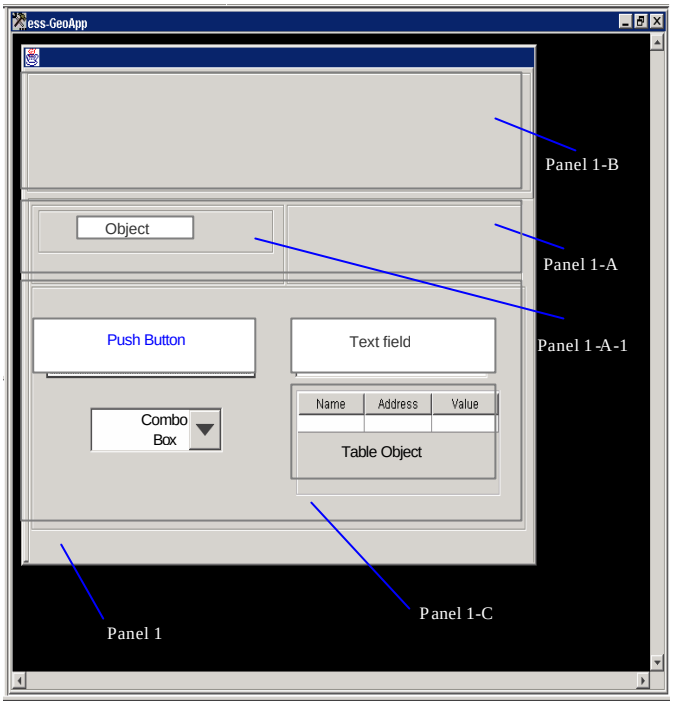


**Figure 4.** A sample workspace where users can build their application panels.

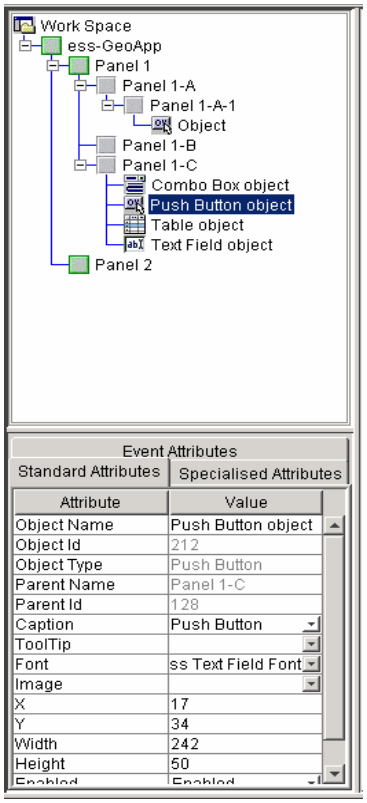


**Figure 5.** Panel hierarchy corresponding to the user-defined panels displayed in Figure 4.

**2.1.3. Database layer.** The database layer, as described in this paper, refers to a single or multiple database instances. This layer comprises two parts: the *i*SMART™ application specific metadata and the user data. The first component is a collection of standard database tables. The core behavior and characteristics as well as the functionality of each user-built application are defined in these metadata tables. All pre-defined SQL statements are also stored in these tables.

## 2.2 *i*SIS architecture

Similarly to the *i*SMART™ architecture, the *i*SIS architecture presents three main components: the *Client Layer*, the *Application Server Layer*, and the *Database Layer* (see Figure 6). Each component is described in the following sections.

**2.2.1. Client layer.** The *i*SIS client layer bean is a Java plug-in that controls the display and manipulation of the vector and raster data. The bean requests the specified data from the Dispatcher EJB (on the application server layer described later), which in turn renders all appropriate data for client display using the bean.

**2.2.2. Application server layer.** The application server layer contains several EJBs. The Dispatcher EJB is a stateless session bean that handles communication

with the client and coordinates the display of the different Sector EJBs. When the Dispatcher EJB receives a display request, it hands such a request over to the Sectors responsible for the portion of the map that is being viewed. It then combines the results and sends them back to the client.
Source EJBs are entity beans that control the access to the SI Server metadata in the database. This information is used to determine whether the subset of data considered (called source) is a vector or a raster dataset and where the source information can be found. A Source EJB also controls which feature table (storing the feature symbology) is used to represent the source information.

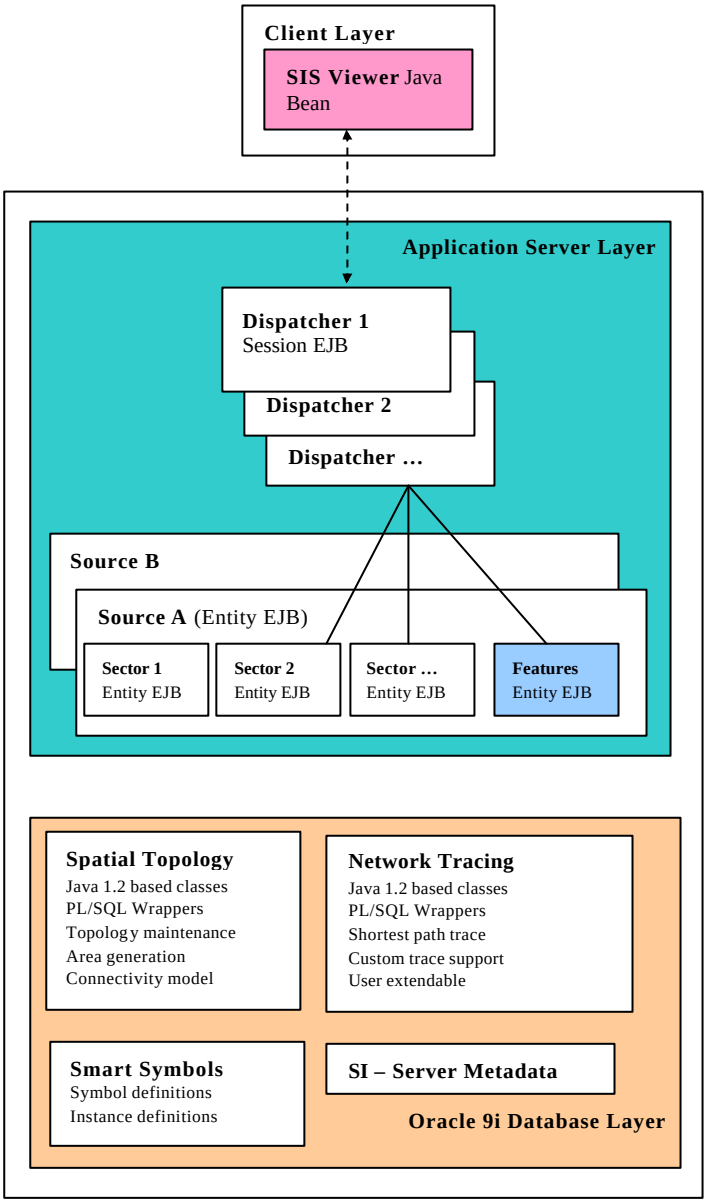


**Figure 6.** iSIS architecture components.

A Feature EJB is an entity bean that controls access to the feature tables in the database. The feature information is used by the Sectors to display object information. The Feature tables can hold, for example, data pertaining to how an object should display at varying zoom levels, colour, style, weight, etc.

Finally, Sector EJBs are entity beans that handle the retrieval of the data from the database for a given area (called a sector). These Sectors in turn send the information to the Dispatcher which combines all required sectors for return to the client.

**2.2.3. Database layer.** The *i*SIS database layer consists of a collection of Oracle database tables and stored procedures written in Java that run inside the Oracle Aurora Virtual Machine. This way, all functions and procedures have fast and efficient access to the information stored inside the database without adding to the network overhead. These functions can be accessed via PL/SQL wrapper code. PL/SQL is the procedural extension of SQL defined by Oracle.
The database has four distinct features.

1. *Spatial Topology*: the spatial topology package builds and maintains a topology model in the database for the spatial objects. This topology is built and maintained from the geometry objects in a specified source table.

Functions include:

- Topology creation from tables containing Oracle Spatial objects.
- Topology maintenance functions.
- Maintaining object connectivity model.
- Dynamic generation of areas from the boundaries in the topology.

2. *Smart symbols*: these are representations of map symbols that can have an internal connectivity and a number of external ports. The smart symbol definition models the possible states for the internal connections between the ports of the object. The smart symbol instance models the current state of all the internal connections of a given smart symbol. This state information can then be used by the trace modules to conduct a network trace inside the database.

3. *Network Tracing*: the network trace classes are a series of predefined classes that can be extended by the user to perform any type of trace. These classes work with the topology information and the smart symbols to determine connectivity and connection information. The base trace classes use topological connectivity to determine if objects are traceable and connected. Users can easily extend this model to determine if components are traceable by accessing other criteria such as attribute data from secondary tables or other criteria. The definition for the trace algorithms is also stored in a set of metadata tables that can be modified and extended with the use of SQL. This allows the users to create their own trace definition on the fly.

4. *Spatial Information Server Metadata:* since the spatial information server is a generic tool to display spatial information from any table that contains Oracle spatial data columns or Oracle Intermedia images, it requires some information on where it can find these objects and images and how it should access them. This information is stored in the SIS metadata tables.

**2.3 Architecture scalability and reliability**

The *i*SMART™ system retrieves operational metadata from a single database, but can access user-based information across a number of different databases. Similarly, the *i*SIS transparently supports displaying of graphical data from different databases. Thus each individual database server could be optimized for the data it controls. Aside from this, the usual database tuning options are still available since only standard database formats and structures are being used to store the Application Specific Metadata.

The application server layer can be scaled in a number of ways. In its most advanced form, the application server supports the creation of a single virtual application server that is built across a number of physical machines. The application server then handles all the load balancing and connection issues arising from this set-up. Alternatively a number of parallel application servers could be created and the user connections can be distributed across those machines. The Dispatchers in *i*SIS can also handle load balancing by sending the process requests to Sectors residing on different machines.

The *i*SMART$^{TM}$ applet is a micro thin applet. The nature of the applet is to execute commands received from the application server layer, and therefore requires no application specific coding on the client for customization. Whether the Application Specific Metadata contains information for one or many applications, the client applet never gets any larger. The *i*SMART$^{TM}$ applet serves the client layer with exactly what it needs, when it needs it, eliminating the need for the client to interpret compiled application specific source code. This dramatically reduces the storage and memory requirements of the client layer.

To ensure reliability for the database layer, Oracle's fail-safe mechanisms are utilized. Parallel server technologies can also be used. The Oracle 9iAS application server handles fail-over between nodes transparently and hence controls application server reliability.

## 3. Additional utilities

Additional tools are provided for facilitating the building of applications that exploit spatially enabled datasets. These tools are particularly useful for non-expert programmers as they allow the creation of features together with their symbology, the importing of digital imagery and incorporate the drag-and-drop methodology typical of graphical user interface design.

The Pyramid Builder is a utility designed to load scanned aerial/satellite photos and topographic maps into the database (see Figure 7). The loaded imagery is then available for heads up digitizing.

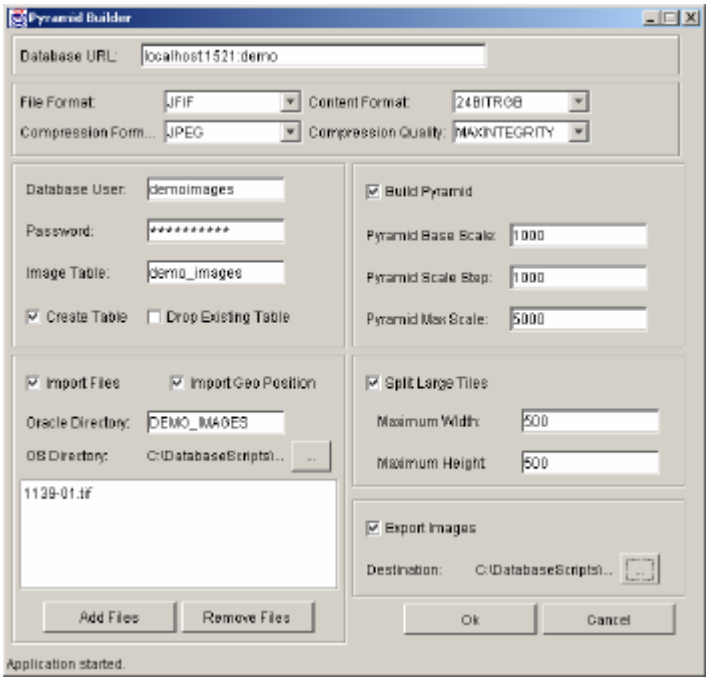


**Figure 7.** The Pyramid Builder.

This image loading utility follows the long standing approach to digital image handling within existing dedicated image processing systems by pre-processing the image data into multi-resolution representations (Carswell and VdLan 1992), hence the name "Pyramid Builder". The idea is to eliminate the need to resample the image at run time by loading the best fit resolution image to the zoom factor currently active within the web browser.
*e*-Spatial™ technology advances this trend by further subdividing the individual reduced resolution image files into tiles of a pre-defined pixel size. Thus, when a web browser is viewing a particular geographic location, only those individual tiles at the requested zoom factor will be physically sent over the network, and not the entire image file.

This of course is essential to real-time viewing of digital imagery due to their inherent size restrictions. For example, a typical black and white aerial photo scanned at 30µm (850dpi) resolution requires approximately 60Mb of disk space (uncompressed) (Kern and Carswell 1994). Colour imagery of course at the same scanning resolution will require three times as much space; i.e., 60Mb for each of the three colour bands of red, green, and blue.

The Feature Builder is a utility that allows users to define new data sources and the feature classes that the source controls (see Figure 8). All feature characteristics (e.g., colour, weight, pattern, etc.) are defined using this utility.

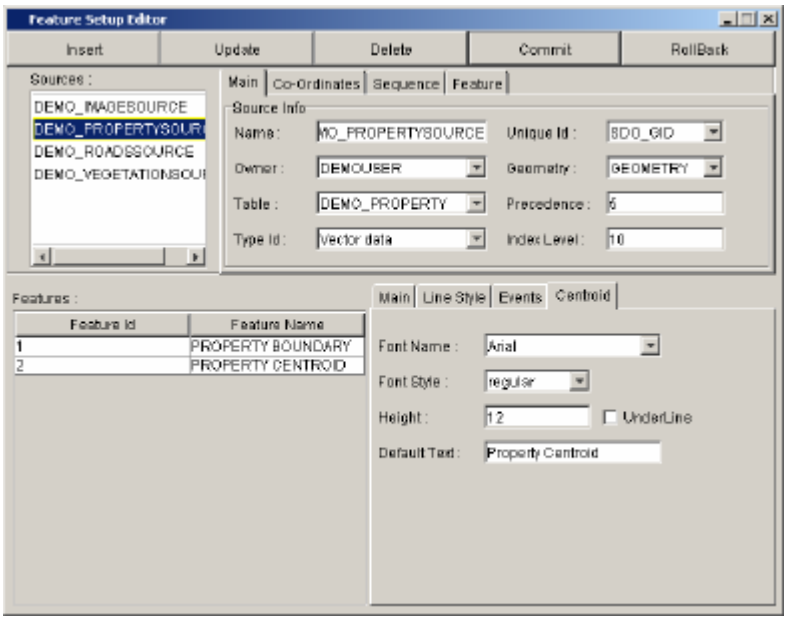


**Figure 8.** The Feature Builder.

Also incorporated within the Feature Builder is the capability to add events to specific feature digitizing operations. For example, in a land parcel application, when a "Property Centroid" feature first gets digitized, an event can be associated with this operation that calls a pre-defined SQL query to populate the "Property ID" column in the user attribute "Property" table and subsequently display this attribute value as the centroid text in the graphics. Alternatively, if a Property Centroid gets deleted from the graphics, an SQL query previously attached as an event to this type of an operation (on this type of feature) can delete all the user attributes associated with this centroid feature.

The *i*SMART™ application builder, called Visual*i*, has been designed to allow any user to build a customized graphic user interface for both standard MIS and spatial applications without writing any source code. This application development environment incorporates drag-and-drop

functionality to add objects, such as buttons, tables, combo boxes, etc. to panels and sub-panels in a hierarchical fashion (see Figures 3,4,5 for screen shot examples of a user application under construction). Therefore users are not required to have any Java or database management programming experience in order to create their own application interface. Visual*i* works similarly to building applications in Visual Basic where all objects and commands that act on the objects are accessed via drag-and-drop tools. These actions in turn correspond to Java procedures that allow for real-time two-way interaction between the client and the database.

Visual*i* allows the building of "business rules" that are attached as events to the buttons and other GUI objects contained within the application panels. For example, a business rule can be created and attached as a "mouse pressed" event on a "Property Value" button object that subsequently shades all the property polygons according to their "Property Value" attribute.
SQL commands are also created within the Visual*i* application builder through the use of the SQL Command Editor (see Figure 9).

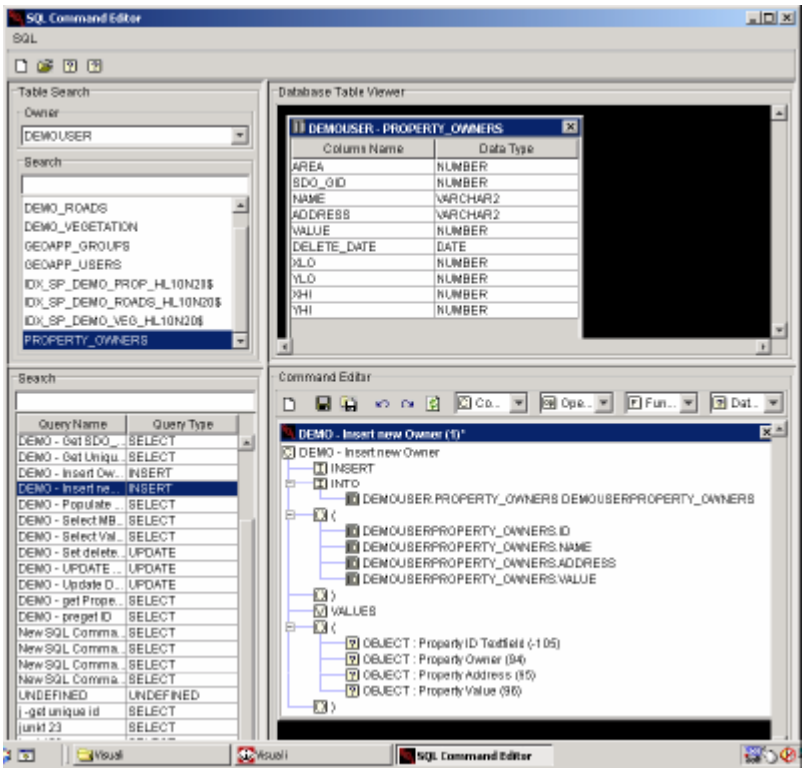


**Figure 9.** The SQL Command Editor

Using drag-and-drop operations, SQL queries are built where table columns can be dynamically attached to their respective "textfield" objects on the application panels. Thus both retrieving data from the database and inserting/updating data are easily accomplished.

## 4. Land information management system

In this section we focus on one particular solution implemented using *e*-Spatial$^{TM}$ technology. This LIMS application, developed for the Irish Department of Agriculture and given the acronym "iMap" (Internet Mapping and Area Payments), is a web-based spatial information system that serves 125,000 farmers in the territory of the Republic of Ireland.

The developed system provides a seamless Oracle 9i Spatial database environment for the combination of multiple land information datasets. Integrated database topology ensures the integrity of the topology relationships within the spatial database. The normal edit (i.e. create, modify, and delete) and spatial analysis functions associated with traditional GIS based land management applications, are deployed as Java stored procedures in the Oracle Spatial database. The implemented functionality includes:

- Vector data manipulation (pan, scroll, zoom, locate, window queries, etc.);
- Vector and attribute data entry and editing;
- Spatial and attribute data analysis (polygon overlay and network tracing);
- Thematic mapping (display of query results with assigned symbology);
- Raster/Vector integration.

A functional overview is provided in Section 4.1 while Section 4.2 discusses some real operational results of the completed working system.

### 4.1 Functional overview

Some functionality provided by the Land Information Management System include:

- View farmers' details online,
- Locate a parcel/area using different criteria ,
- Digitizing,
- Spatial queries,
- Printing,

Accessing iMap requires all users to logon with a valid name and password. Different security restrictions are applied to different groups of users (e.g., only viewing/querying, viewing and editing, etc.). In the following we describe how the system is used after authorized logon. At first the user is prompted with a homepage from which several options can be chosen to display the requested iMap applet (Figures 10 and 11).

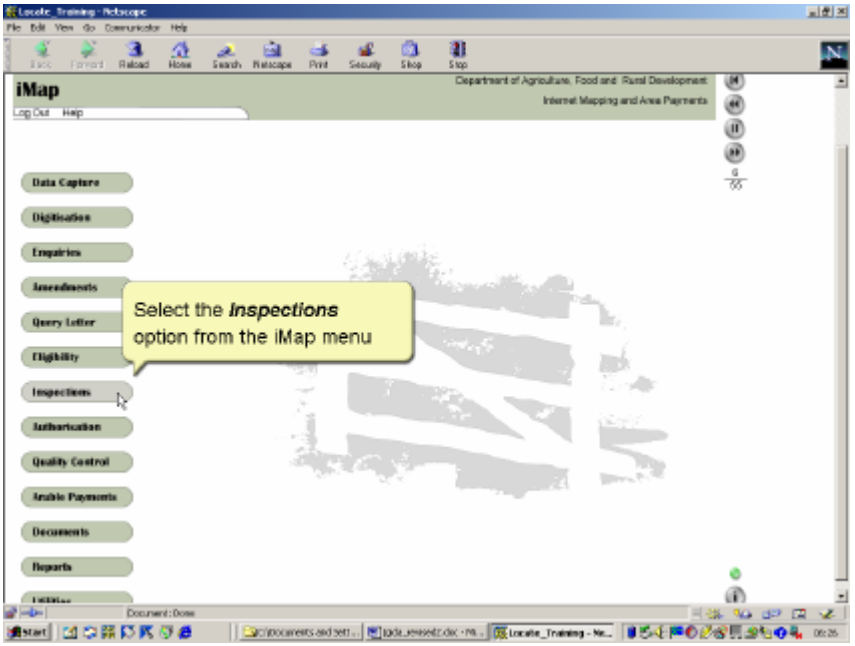


**Figure 10.** The iMap homepage displays all available options to authorized system users

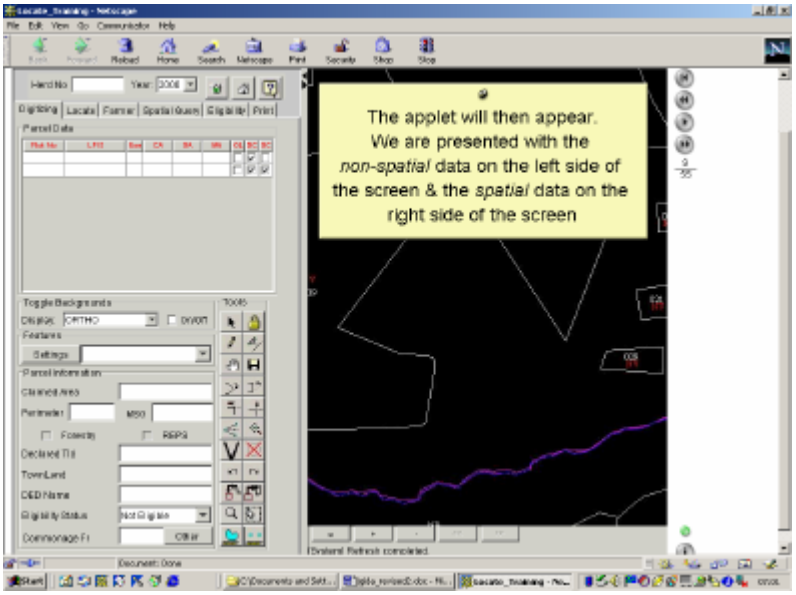


**Figure 11.** Selecting the "Inspections" option on the homepage loads the above applet

From this screen, the Farmer/Locate options can be selected to view farmer details and locate parcels. By clicking on the locate button (Figure 12) it is possible to locate a parcel/area using several different criteria: cadastral locate (specifying a county), Ordnance Survey locate (entering map numbers), coordinate locate (enter *x* and *y* map coordinates), LPIS number (land parcel ID).

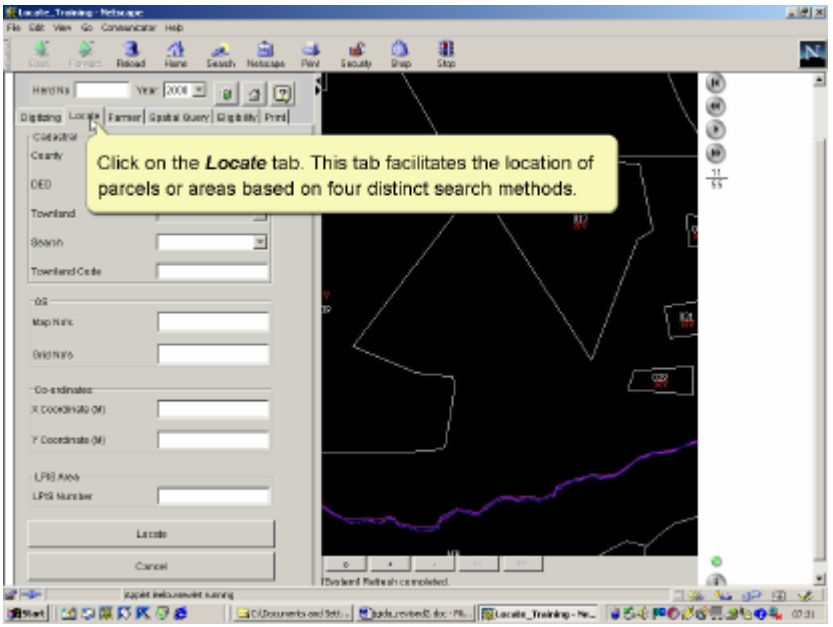


**Figure 12.** Locate panel: example of locate functionality using any of 4 search methods.

Alternatively, the user could enter a herd number and a year to display all parcels where the specified herd is located. One specific parcel can then be selected and viewed (Figures 13 and 14).

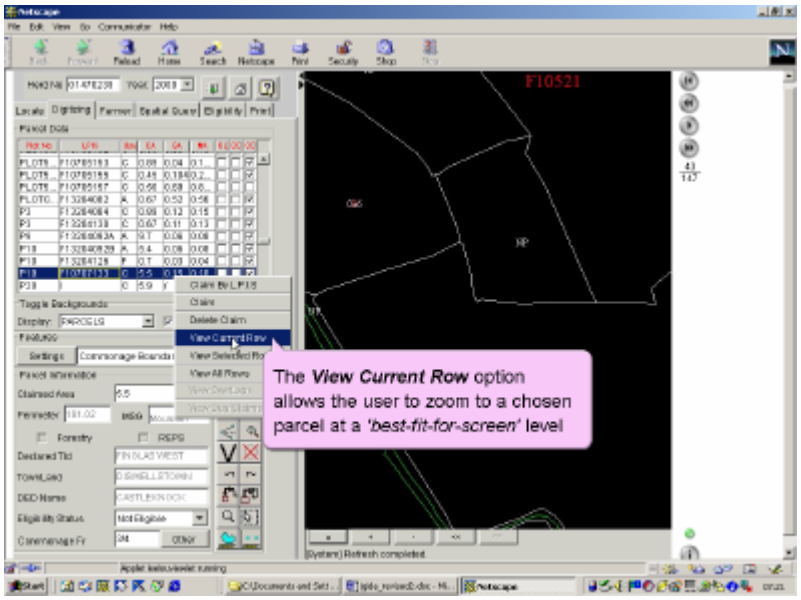


**Figure 13.** Alternative locate method: the user enters a herd number and year at top of the form; then clicks on the Digitising tab (Parcel Data fields will subsequently be populated for the chosen herd number); the specific parcel is then selected and viewed by right clicking on it and choosing the *View Current Row* option.

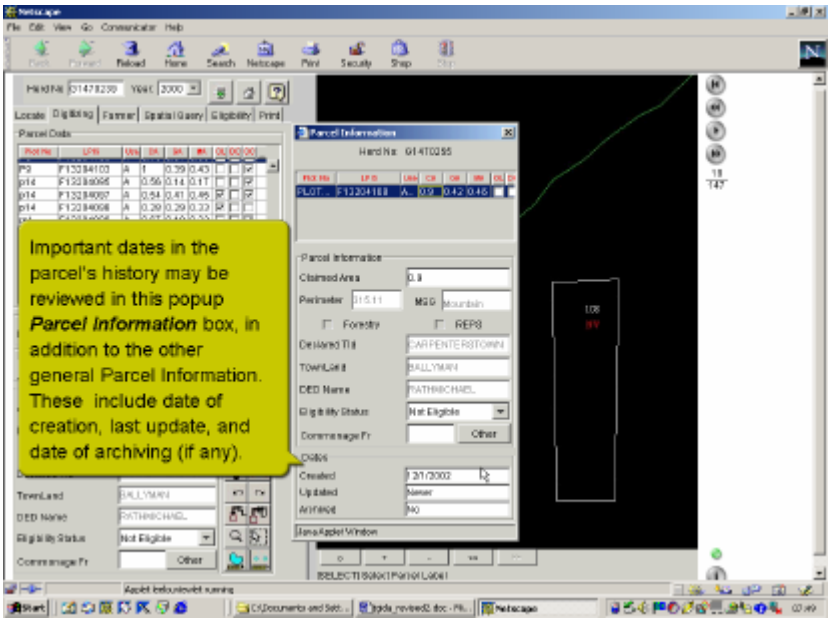


**Figure 14.** Additional information for a selected parcel is available in popup message windows.

The on-line digitising functionality is accessible directly from the result screen of the locate search described above by clicking the digitising tab (see Figure 15).

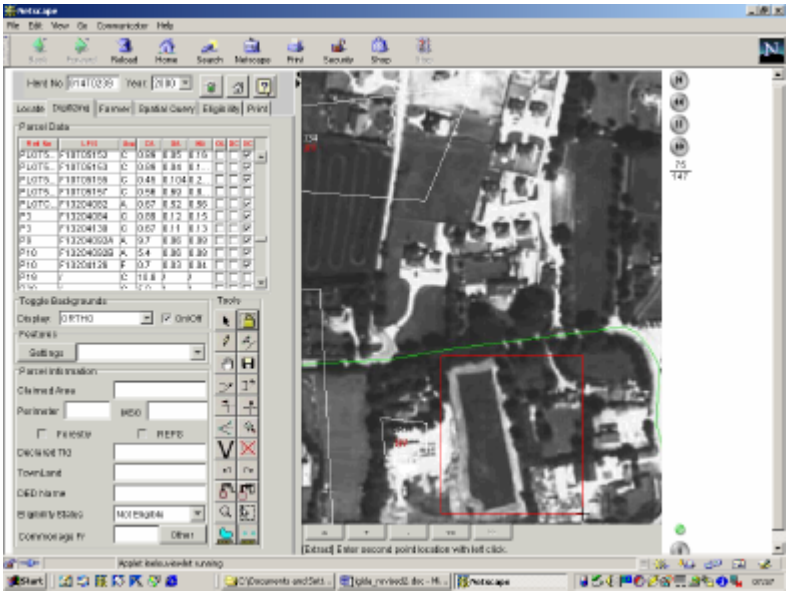

**Figure 15.** Digitizing panel: besides viewing parcel data and changing visualization settings, the user can utilize the standard digitizing and validation tools shown.

Once the user has digitized a new parcel boundary, topology updating algorithms can be automatically or manually invoked within the Oracle 9i Spatial database (Figure 16).

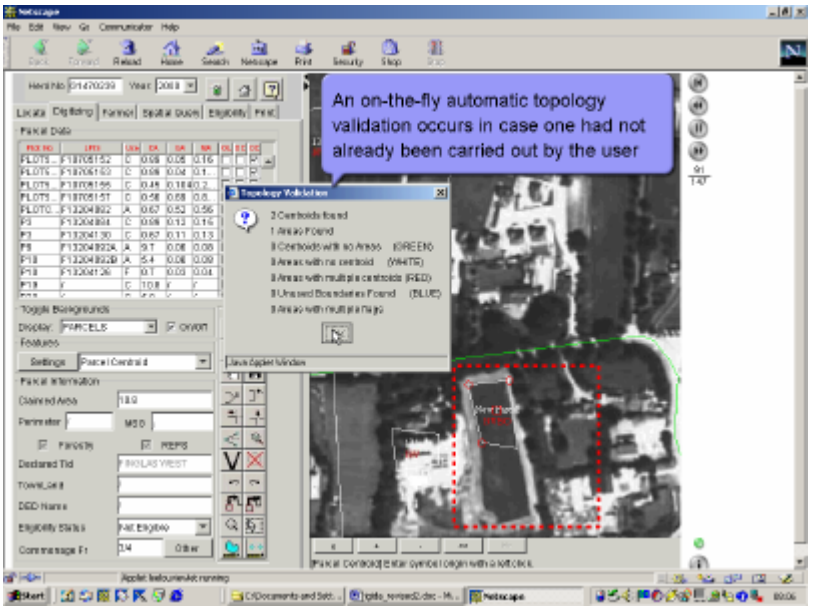


**Figure 16.** Validating topology on-the-fly

Spatial query functionalities of the system include: retrieving parcels of a given shape, area, centroid coordinates; buffer queries; window queries; topological queries (e.g., based on overlapping, containment, etc.). Thematic queries are also allowed where re-symbolisation of an area on a map is based on its attributes. In Figure 17, setup for a Spatial Query example is illustrated.

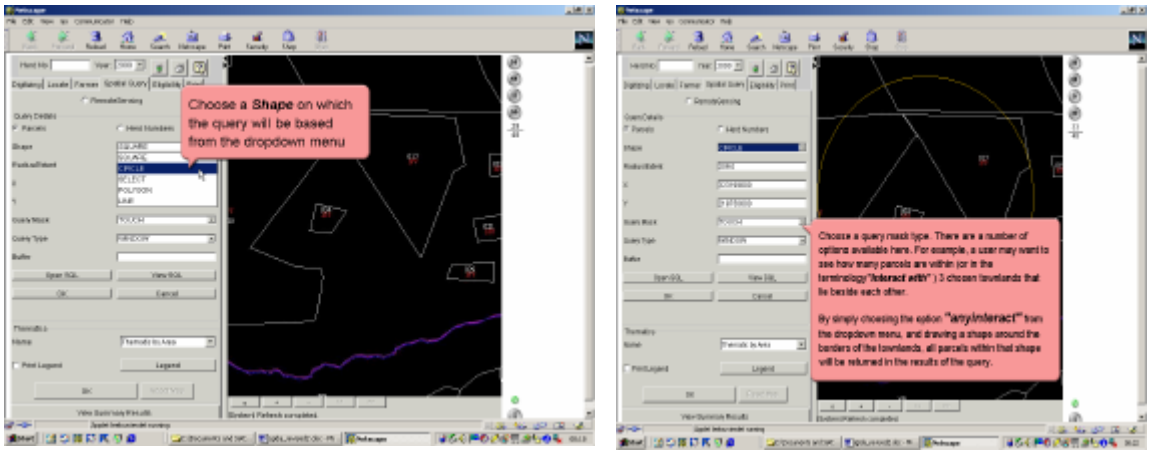


**Figure 17.** Spatial query panel: the user can enter query details such as the shape of the geometry, the distance from specified map coordinates, the query type (window, buffer, etc.), the topological relations (overlap, touch, equal, anyinteract, etc.). The corresponding automatically generated SQL statement can be viewed and manually edited if required.

The most basic queries, i.e., those retrieving the most frequently requested information, are built into the GUI. However, there is an option for viewing the automatically generated SQL statements and modifying, editing, and resubmitting them for the more SQL skilled users (Figure 18).

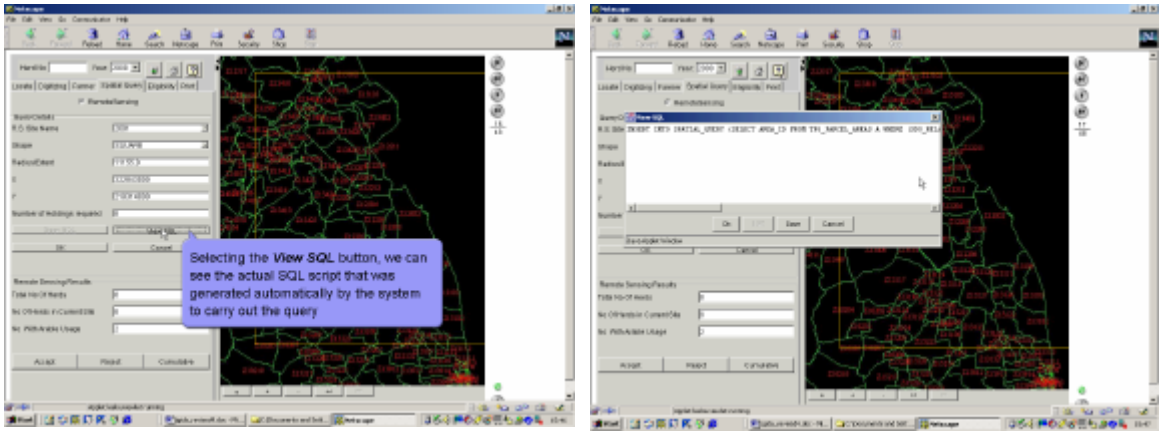


**Figure 18.** Viewing and editing the automatically generated SQL statement

Immediate printing capabilities of graphical/attribute editing updates represent additional benefits of the system. To enable printing capabilities the Print tab is selected. Two different modes are available: on-line printing and batch printing. Using the on-line mode, users can print (see Figure 19):

- Screenshots: prints what is currently visible in the viewer;
- On-screen parcels: prints a parcel based on its LPIS number, which is entered into the relevant field.
- On-screen herd numbers: prints a farmer's entire herd at a zoom level and paper size chosen by the user;

- On-screen townlands: prints a townland map with any required background depending on user preference.

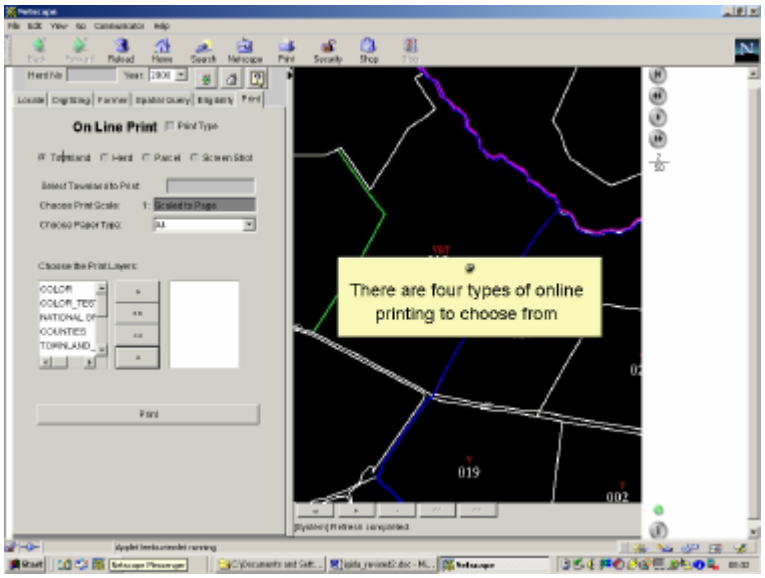


**Figure 19.** Printing panel with on-line option: a wide range of parameters can be selected by the user (zoom levels, paper size, background settings, etc.).

### 4.2 Operational results and decision support

In this section we provide some operational results of the iMap System developed for the Irish Department of Agriculture. This system (on-line since February 2002) serves 125,000 farmers in the territory of the Republic of Ireland for a total of 700,000 field parcels. Table 1 shows typical datasets sizes in terms of number of bytes and number of records.

**Table 1.** Typical dataset sizes.

| Type of Dataset | Size of Datasets | No of Records |
|---|---|---|
| Orthophotography | 80GB | 24,000 |
| OS Maps in Raster Format | 180GB | 24,000 |
| Geometries | 12GB | 7,000,000 |

In the first two weeks of on-line operation the system generated 200,000 maps. The average number of spatial users is 16, with a peak number of 64. The average response time for a locate query is 7 seconds without caching capabilities. Cached locate queries require on average 2 seconds. Updates (e.g. posting a new geometry) require an average time of 5 seconds including recalculation of topology within the database. Further testing of processing times under load and a formal analysis of usability feedback from the farmers are currently on-going. However, initial reactions have been quite positive, with farmers being able to visualize and query their parcel/herd holdings from every part of the country. This makes the LIMS a successful spatial information system for non-expert users.

As described previously, the overall system allows for a wide variety of spatial analysis functionality for both the external users (i.e., the farmers) and the Irish Department of Agriculture (DOA) itself. The centralized database is hosted and maintained by the DOA who therefore have complete control and monitoring privileges over the data. Statistical data analysis (e.g., land use, land

classification and land ownership changes over time, etc.) can be compiled using standard Oracle reporting functionality thus making this system an effective decision making tool. For example, decisions regarding subsidies, taxes, etc. can be made according to land use values. Also, the use of orthophotography and satellite images provides support for cross-referencing farmer reports on land classification claims.

## 5. Concluding remarks

This paper presents an innovative technology that extends Oracle Spatial databases by adding spatial information management functionality that can be used within web-enabled applications accessed via any device that supports the Java Virtual Machine, e.g., standard web browsers, PDAs and 3G mobile phones. A major advantage of this technology is that it does not require the use of complex GIS proprietary application specific software.

Besides reducing the complexity of system management by eliminating the hybrid architecture of typical GIS data models, additional specific benefits (from the point of view of data management) of exploiting the Oracle Spatial unified approach include:

- The possibility for users to access full function spatial information systems based on industry standards with an open interface to all data (e.g., SQL);
- The capability of storing spatial data in enterprise-wide database management systems thereby enabling the spatial development of several more enterprise applications;
- The complete integration of management information and spatial data repositories that allows for the development of geo-spatial applications providing additional analysis and reporting functionality.

Using *e*-Spatial™ technology, no application code is ever installed on the client side, thereby eliminating any redundant code. The client applet never exceeds 35k and supports an unlimited number of users and an unlimited number of applications. With the database generating all requested information (including both application functionality and attribute data) on the fly, only the minimum data required is downloaded to the client, which leads to the most economical usage of bandwith and resources for real-time response.

Existing systems have no ability to switch on/off specific subsets of spatial data. They are restricted to using all the data contained within each data layer. Therefore the data content is fixed. However, using *e*-Spatial™ technology not only are all datasets maintained in a single Oracle database, but the users can also select the particular data they wish to view and query. For example, in the case of a routing query they may only wish to see the road network from their starting location, and the buildings only within 500 metres radius of their final destination. They have no requirement to view buildings data along the actual route itself. This ability to switch on and off data and also define exactly what the users need to query greatly improves the speed with which the individual query is completed. This characteristic also conforms to the information-on-demand approach discussed by several authors in the context of web-based vector map generalization (Bertolotto and Egenhofer 2001; Buttenfield 1999; Cecconi and Weibel 2000). The potentialities of the application of *e*-Spatial™ technology in this context are currently under investigation.

From a data management point of view, a critical issue involves guaranteeing the preservation of topological consistency (Egenhofer and Franzosa 1991). The integrated Oracle 9i topology management functionality completely controls topology within the *i*SMART™+*i*SIS platform: if a spatial element is updated, all spatially related elements are automatically changed accordingly to guarantee consistency. *e*-Spatial™ technology strongly relies on this intrinsic feature of Oracle Spatial. A different database engine that does not provide equivalent functionality would not be suitable for this technology.

Another advantage of the *e*-Spatial™ technology relates to its ease of use. Indeed it includes a user-friendly environment that allows building and customizing the application by means of drop-drag functionality, without requiring any specific programming skills. In this way, even non-expert users can easily build their own spatially enabled applications.


**Figure 20.** The LIMS viewed on a PDA.

In this paper we have shown the potentialities and performance of *e*-Spatial™ technology as deployed in the LIMS developed for the Irish Department of Agriculture. This system serves 125,000 farmers in the territory of the Republic of Ireland and after two weeks of on-line operation, it generated 200,000 maps. This on-line application represents a powerful e-Government service available to the Irish agricultural community and is one of the first of its kind in Europe. In future, the land information management system will also be available on hand-held mobile devices (Figure 20). Implementation and testing of this service are currently under development.

## References

Bertolotto, M., Carswell, J. D., McGeown, L., and McMahon, J. (2001) iSmart+i-Spatial Information Server: Deploying Integrated Web-Based Spatial Applications Within an Oracle Database Environment. Proceedings of the First International Workshop on Web Geographical Information Systems (WGIS2001) held in Kyoto, Japan, IEEE CS Press.

Bertolotto, M., and Egenhofer, M. (2001) Progressive Transmission of Vector Map Data over the World Wide Web, *GeoInformatica*, 5(4), 345-373.

Buttenfield, B. (1999) Progressive Transmission of Vector Data on the Internet: A Cartographic Solution. In Proceedings of the 18th International Cartographic Conference held in Ottawa, Canada.

Carswell, J.D., Eustace, A., Gardiner, K., Kilfeather, E., and Neumann, M. (2002) An Environment for Mobile Context-Based Hypermedia Retrieval. In Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA2002) held in Aix en Provence, France, September 2002, IEEE CS Press.

Carswell, J.D., and VdLaan, F. (1992) Tools for Large Scale Operational Raster Processing. In Proceedings EGIS92 held in Munich, Germany.

Cecconi, A., and Weibel, R. (2000) Map Generalization for On-demand Web Mapping in Proceedings GIScience2000 held in Savannah, GA, USA.

East, R., Goyal, R., Konovalov, A., Rosso, A., Tait, M., and Theodore, J. (2001) The Architecture of ArcIMS, a Distributed Internet Map Server. In Proceedings SSTD'01, *Lecture Notes in Computer Science n. 2121*, pp. 387-403, Springer-Verlag.

Egenhofer, M., and Franzosa, R. (1991) Point-set topological spatial relations, *International Journal of Geographic Information Systems*, 5(2), 161-174.

Kern, P., and Carswell, J. D. (1994) An Investigation into the Use of JPEG Image Compression for Digital Photogrammetry: Does the Compression of Images Affect Measurement Accuracy. In Proceedings EGIS94 held in Paris, France.

Sondheim, M., Gardels, K. and Buehler, K. (1999) GIS Interoperability. In P. Longley, M.F. Goodchild, D.J. Maguire and D.W.Rhind (eds.) *Geographical Information Systems: Principles, Techniques, Management and Applications*, pp. 347-358, Cambridge: GeoInformation International, Second Edition.

[URL1] Autodesk Map Guide 6 http://www.autodesk.com.

[URL2] ESRI ArcIMS http://www.esri.com.
[URL3] INTERGRAPH GeoMedia Web Map http://www.intergraph.com.
[URL4] MapInfo http://www.mapinfo.com.
[URL5] OGC (OpenGIS Consortium), 1996a, *The OpenGIS guide, introduction to interoperable geoprocessing, Part 1*, http://www.ogis.org/guide/guide1.html.
[URL 6] OGC (OpenGIS Consortium), 1996b, *The OpenGIS abstract specification*, http://www.opengis.org/public/abstract.html.