

Paper Title

Dynamic Additions and Updates to Spatial Data Through the Internet

Paper Authors

Brent Porter

Chris Williams

Paper Abstract

TNRIS (Texas Natural Resources Information System) has designed a Web interface and database design for collecting and displaying information about emergency facilities (i.e., evacuation shelters, including location, status, photographs, type of facility, etc). This application uses ArcIMS, ColdFusion, and Java API for SDE and SDE/SQL Server to obtain information about shelters for visualization and logistics support through a Web-based tabular interface and a custom ArcIMS application. Specifically, the application will help local, regional, and state officials make better decisions about where and when to open a shelter in the event of an emergency such as a hurricane. More generally the application shows a proof of concept for dynamic updates to spatial datasets over the Internet as well as good database design and implementation strategies for such an application.

Introduction

The Texas Natural Resources Information System (TNRIS) entered into an agreement with the Texas Engineering Extension Service (TEEX), part of the Texas A&M University System, to develop an online GIS application that could be used to track the status of emergency shelters used during hurricane events. From an IT stance, the project was comprised of two main tiers: 1) the database tier, and 2) the web-mapping tier. TNRIS staff has extensive experience creating applications that utilize ESRI's ArcSDE and ArcIMS server products. Those products were natural choices for satisfying the spatial storage and display requirements of the projects. Other software utilized in this project are SQL Server, Cold Fusion, Java Server Pages and Java Script.

The project had many levels of sophistication that are usually not seen in web-based GIS applications. The requirement for the basic functionality of the application included:

- 1) Shelters should be allowed to have their attributes, including status and issues, modified in a real-time process.
- 2) The application should allow the creation of new shelters in a real-time process.
- 3) The application should be able to support facilities that change location (ie. Mobil Feeding Kitchens that are called into service during storm events)
- 4) The application must remain secure because of the sensitive nature of the data
- 5) The application needed to have a visual component that would allow different levels of users the capacity to print and assess the status of the shelter network.
- 6) Functionality for archiving shelter data via the web front end will be developed
- 7) The application would need to provide different levels of users for different access levels once logged on.

- 8) The application needed to provide a non-GIS method for update and addition of points as well as a means by which to report status of the shelters based on some set of criteria (e.g. By county, By city, etc.).
- 9) The application needed a way to allow addition and update of shelters within a geographic context in the event that there was no other means for capturing location information.

Technically, the largest hurdle to overcome was the addition of new shelter locations to an SDE layer in a real-time. Knowing the limitations of out-of-the-box ArcIMS a plan was developed to overcome this major hurdle by leveraging web-based technologies against the Java API for SDE to overcome this technology hurdle. Eventually, a solution was developed that utilized these technologies to add new point location to an existing ArcSDE layer. And, because ArcIMS re-queries an SDE layer upon every redraw, the solution satisfied the project requirements for creation of new shelter points in a real-time fashion.

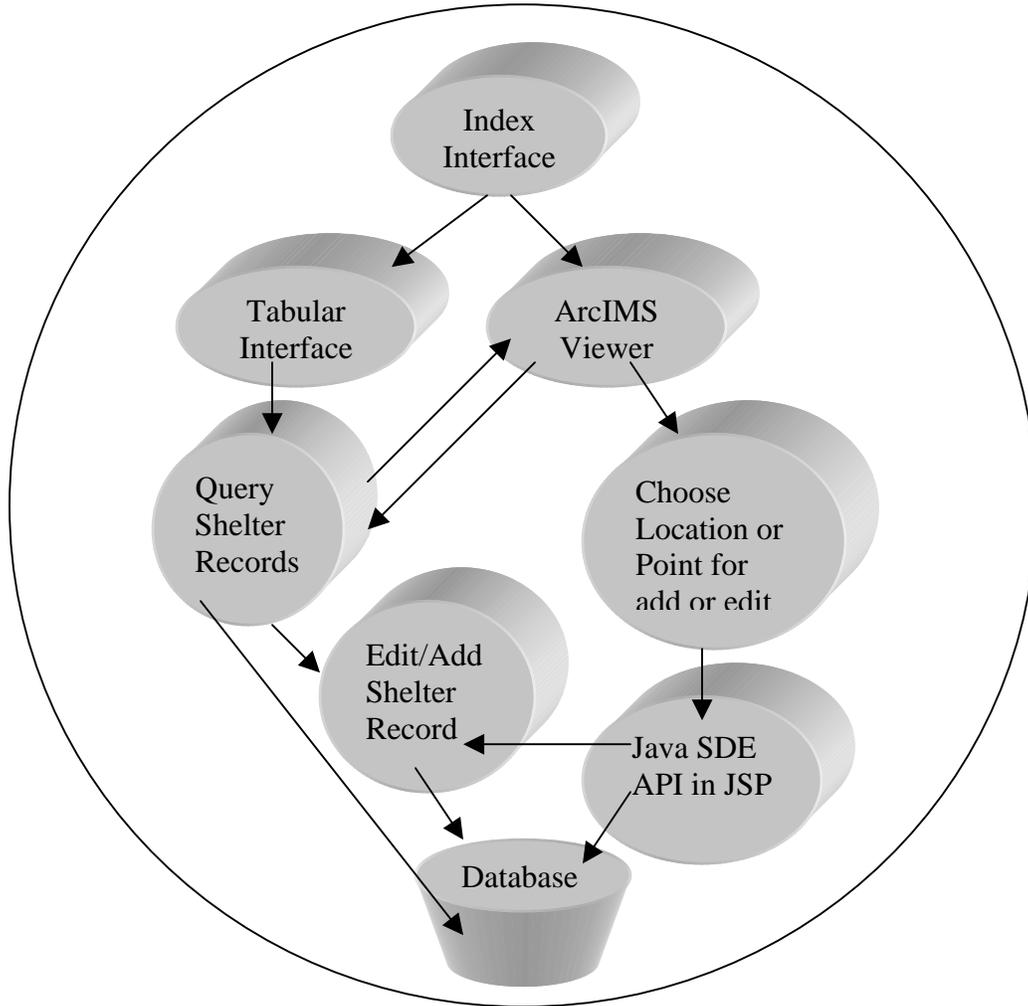
The database tier was not as difficult with regard to satisfying project requirements. Standard database practices were evoked to satisfy the bulk of the project requirements. However, because of the approach decided upon, interesting design issues had to be addressed. In the database tier, many considerations were given to security, data design, security, and data reporting.

Security – Security was a significant issue for the project. Four main user groups were documented and a security solution was developed. The solution involves the use of an ACL that is housed in the database. The other issue that was of concern was SQL injection attacks if the initial level of security of the site (secure login) were to be compromised. To mitigate this, all information collected in the data entry pages of the web front end were sent as parameters to a series of SQL Server stored procedures. The parameters were then evaluated for data type and value where appropriate.

The Solution

1. Early on it became apparent that different technical solutions were going to be necessary for different issues.
2. SQL Server 2000 was used, running ArcSDE on top of it.
3. Java Database Connectivity (JDBC) and Cold Fusion ODBC connected the application layer to the database layer.
4. The ArcSDE Java API, integrated into a Java Server Pages (JSP) application communicated directly to ArcSDE.
5. ArcIMS html viewer and some custom javascript would be used for data visualization and the customized tools used to digitize new shelter locations and shelter selection for status updates.
6. Other scripting components were written with javascript and CFML scripting language.

Flow Diagram



Design – The design for the database was not known. TEEX provided TNRIS with an initial list of required fields. From those fields and from discussions involving actual use of the infrastructure a table schema was developed (see image below).

- 3) Typically, it is not a good idea to be in the design phase of the database schema while concurrent application design is underway. Developers do not like it when schemas are changed, without warning, under their applications.

Best Practices

- 1) Standard naming conventions for tables, stored procedures, and views.
- 2) Documents that describe changes to schemas, both proposed and accepted.
- 3) Writing code in a modular manner so that it can be re-used
- 4) Following accepted coding procedures for variable names and calling functions and other stored procedures.