Introduction
As GIS professionals, many of us use geodatabases routinely, whether we are making maps, performing data analysis, or administering and maintaining them. There are many parts to successful geodatabase implementation and ongoing usage. This whitepaper covers what a geodatabase is, and geodatabase design and development basics. A second whitepaper, Geodatabase 201, covers geodatabase management, maintenance, and troubleshooting.

What is a geodatabase
A geodatabase (also known as a spatial database) is a relational, object-oriented database built to store and query geographic data, i.e., data that has a defined location on the Earth. All relational databases are built to capture and manipulate various numeric and character data types, and the geodatabase is no exception. However, the geodatabase is equipped to work with various feature or geometry types as specified by the Open Geospatial Consortium (OGC). Most simply these can be stand alone personal or file geodatabases, or can be geodatabases configured for multi-user access across a broad organization using relational database structures (RDBMS) such as PostGres, SQL, Oracle, and others. One key geodatabase characteristic is the ability to support multiple feature layers, tables, and relationships between layers and tables. This distinguishes “geodatabases” from file formats that only contain one layer and attributes, such as a shapefile or a keyhole markup language (.kml) file.

While geodatabases and non-spatial relational databases share similarities, they differ in some important ways. Both types of databases allow for relationships to exist between objects. For example, in a non-spatial database, a table of employee names and contact information can be related to a table of employee departments and job titles based on an employee ID number. That same relationship also exists in a geodatabase, but also contain spatial data that is in one or both tables, like the geocoded locations of the employees’ addresses.

Other notable differences between geodatabases and non-spatial relational databases are spatial queries and indexes. Spatial queries allow us to look at the relationships between spatial features. Spatial features can be simple geometry (e.g., points, lines, polygons) or more complex types (e.g., 3D, rasters). An example of a spatial query can be whether features in two different polygon layers overlap. Indexing in a relational database involves creating a data structure to improve the speed at which data is retrieved from the database. Spatial indexes allow for spatial data to be indexed to improve spatial query performance.

For additional information on geodatabases, refer to the Resources section.
**Geodatabase Design Basics**

Geodatabase design can be a challenging topic to cover because it is unique to each individual situation. A geodatabase design for a small, non-profit organization is likely completely different than one for a large, multi-national corporation. Even within the same organization, design may differ between projects or departments. This section covers broad design categories and things to consider as you design your geodatabase.

The first step to any good geodatabase design is to understand why you are designing a geodatabase in the first place. What is the end goal you are trying to accomplish? This question cannot be answered in a vacuum unless you happen to be designing a geodatabase just for your own use. More often than not, geodatabase design starts with a needs assessment. A needs assessment can be simple or complex depending on your situation, but it generally always starts with working with the stakeholders involved in the project and understanding and documenting their needs. Keep in mind that stakeholders are anyone from the end users of the data to the data editors, IT staff, or others who may be involved in creating and/or maintaining the data or infrastructure once the geodatabase is built.

During the needs assessment, you may need to discuss the following:

- Determining where the needed data will come from
- Determining how often the data needs to be updated and how that will be done
- Determining what data should be spatial and what should stay in a tabular format
- Determining what types of analysis may need to be run on the data and how that will be done
- Determining how QA/QC will be handled

These are key issues to tackle when designing a geodatabase, and they can be the difference between a good or bad end user experience.

Once you understand the above issues, you can decide what you need to store in the geodatabase and how it will be related. An Entity Relationship Diagram can be a useful tool for this step (see next page).

An Entity Relationship Diagram is a graphical representation of the entities that will exist in the geodatabase and their relationship to one another.

After the data is mapped out in a logical way and you understand how it will work together, the next step is to build a proof of concept. This is simply a test geodatabase that contains a subset of all the entities that will exist in the final geodatabase, configured exactly as they will be in the final version. This proof of concept will allow your end users, data editors, IT staff, and others to get a feel for how this geodatabase will function once complete. Issues that crop up during this time can easily be addressed before a final version is built.
Lastly, once the final version of the geodatabase is designed, you want to document the geodatabase design and the decisions that were made to get there. This is a crucial step that can often be overlooked. Good documentation can help troubleshoot issues, explain design decisions, and more easily on-board new staff to the geodatabase down the road. A good geodatabase design project is not complete until this step is done.

**Geodatabase Development**

Once the geodatabase is designed, you need to build out the concept into a real, functioning geodatabase. Some things to consider during this stage are:

- What tables will I need to create?
- Do I need to create views to display data from another database instance?
- What spatial reference will I be using?
- What will need an attribute or a spatial index to improve query performance?
- Will I need domains on any of the fields for quality assurance?
During this stage, you also want to consider access to the data as well. How will permissions be handled, and who will have access to what? Role-based permissions are generally recommended because they provide the most ease in managing access. Although the specific setup for role-based permissions will vary between different relational database management systems, permissions are generally assigned by role, and users are assigned to roles rather than assigning permissions directly to users. Roles can then be assigned to databases or kept at the database instance level.

**Resources**

<table>
<thead>
<tr>
<th>Wiki.GIS.com</th>
<th>Geodatabase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spatial Database</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>Spatial Database</td>
</tr>
<tr>
<td>Esri</td>
<td>What is a geodatabase?</td>
</tr>
<tr>
<td></td>
<td>An overview of geodatabase design</td>
</tr>
<tr>
<td>Open Geospatial Consortium</td>
<td>Simple Feature Access</td>
</tr>
</tbody>
</table>

**Further Reading**

Proper geodatabase design and development will ensure that management and maintenance are made easier. In Geodatabase 201, we'll go into detail on managing, maintaining, and troubleshooting geodatabases. A well designed and maintained geodatabase ensures your end users have the best experience possible while freeing up staff time better spent elsewhere.