




Snap™

POINT PAPER

What is Snap™?

 Snap™ is a *framework* incorporating the necessary logic to tie information from distinct and/or heterogeneous databases and files into a single browser-centric view.

In essence,  Snap™ is a way to view desired data from disparate sources -- a common gateway -- that is targeted to the end-users and developers alike. For the end-users, it offers a simple point-and-click interface that allows them to query and view information from different sources in a single container. For the developers, it provides a set of programmatic interfaces and rules that help them to efficiently and rapidly bring to the web information from various locations.

Contrary to the traditional web publishing metaphor,  Snap™ gathers the information directly from the source and formats it on the fly. The user is assured that the information that is displayed is the latest one available. In addition, to achieve high responsiveness and scalability, the system is designed around the load-on-demand principle and implements a drill-down approach where only the requested information is retrieved.

Why Snap™?

In order to bridge databases at least three approaches (each with different variations) can be envisaged. The first approach is to build a new database that aggregates partially or totally tables from other databases, and populate it on a regular basis from the original sources. This approach proves to be inefficient because each addition of a database, table or even a field requires a structural rebuilding of the aggregating database and entail revisiting the code to take into account the new elements. In addition, the information is as new as the latest transfer or dump (replication might prove thorny in heterogeneous databases).

The second approach is to build a data warehouse. Data warehouses are complex applications that require careful analysis and planning to succeed. They take a long period of time to implement and the associated cost is usually high. They work well when the business process is stable, and when trend analysis (time series data) or data mining is an objective. However, data warehouse are simply inept to provide the flexibility needed in a dynamic and changing environment. They fall in the same trap as the first approach. For every new requirement, the data warehouse must be redesigned and structurally altered to adhere to this requirement.

The third approach is to establish a meta-dictionary that provides a mapping between the business terminology (or semantics) and the fields pertaining to the tables and databases. The application logic would then be based on the business terminology, and users requests will be re-routed to the appropriate sources. While this approach is the most flexible it does lend itself to two major flaws. First, in case the mapping is not one-to-one the rules might be difficult to implement. Which mapped-fields should we choose? (One field, a combination, or all referenced fields). Will these rules be enforced across all mappings or across some? Second, under certain conditions, this approach might engender severe performance degradation in terms of connections and resources. To illustrate this point, imagine that a user have requested a page consisting of 20 fields that happen to be mapped to 20 different databases. This means, that this page will need 20 different connections and 20 different queries to gather the

requested information. While this situation is less than optimal for a single user, it should definitely be avoided for a large number of users (the Internet being a case in point).

Parting completely from these strategies 3H Snap™ adopts a minimalist approach. To link distinct databases, the need to identify the commonalities amongst these databases is necessary. These common fields can be accessed directly either as a base table, static query (view), dynamic query (select statement), or stored procedure. This information is the only requirement for 3H Snap™ to be able to bridge these databases. Once this information is collected 3H Snap™'s framework can perform automatic search on these databases.

For every referenced database, the developer will code a specific ASP page(s) (Active Server Page) that will retrieve the requested information. The coding process is simple, fast, and straightforward due to 3H Snap™'s published set of API (Application Programming Interface) designed to speed up the development process and to reduce errors. The minimalist approach shields the developer from any structural changes. The ripple effect of any change is confined to the page that references the database that has been altered. This enables the developer to localize the maintenance to a well-identified boundary. In addition, adding new databases, does not require redesigning 3H Snap™'s framework, only the adjunction of a new page (or pages). As far as directories are concerned, there is no need for development, 3H Snap™ handles it automatically.

How does 3H Snap™ work?

At the heart of 3H Snap™ is the concept of snapin. A snapin is an object that indicates to the system which database table or directory it references and provides methods and properties allowing the user and the developer to interact with it. Once a snapin is created (i.e. registered), the system has enough information to build a search based on this snapin. To be able to search across all registered snapins, 3H Snap™ relies on the “crawler”. The crawler is a program that works in the background (implemented as an NT Service), periodically updating a search table. The crawler updates only the key search (primary identifier) and possibly several alternate fields. Whereas registering a snapin is a relatively simple process (a matter of few minutes) - the crawler being automatic -, the bulk of the work is to develop web-based dynamic pages (ASP) for each snapin. As mentioned above, this process is greatly simplified by 3H Snap™'s native APIs and templates that can be used to generate new ASP pages.

From an architectural standpoint, 3H Snap™ revolves around three axes:

- A database that acts as a repository for the system.
- A crawler that runs in the background and constructs search references.
- A set of API and templates that enable system updates and rapid deployment of ASP pages.

Where can I use 3H Snap™?

When there is a need to bridge multiple “data stores” (databases, mail systems, file systems, etc.) In this case, there must be a common denominator in order to tie in the different sources (e.g. job number, client number, building name, ship number, aircraft number, etc.).

When there is a need to add a new “skin” (web interface) to legacy applications. In this case, 3H Snap™ makes it easy to publish information without rewriting the current applications. This would be applicable when the applications are console-based (text mode). In addition, 3H Snap™ can be implemented for applications that don't have an interface; those who simply consist of a data store. Data warehouses typically fall in this scenario but also any in-house database gateways.

When there is a need to reorganize/categorize the current information without restructuring the existent software architecture (both client and database). In this case, 3H Snap™ allows the creation of “virtual

containers” that would offer a new organization scheme. The virtual containers are a part of the Snap™ repository thus eliminating any need to modify the current software implementation.

When there is a need to apply “dynamic security”. In general, the security model – for databases – is sort of all-or-none. For example, in the database world, consider a user has “read or view” privilege on a particular table. What if what is needed is to allow the user to view the information that he is responsible for but not be able to access all other info in the same table? Out-of-the-box databases do not offer this type of dynamic security. This feature is built-in in Snap™’s framework.

When there is a need to consolidate information coming from heterogeneous environments (e.g. Operating Systems – Unix, Windows, OS/400, etc.) and disparate data store (SQL Server, Oracle, Informix, Sybase, etc.). In this case, Snap™ acts as a universal thin client to the enterprise information.

Snap™ key points:

- Snap™ is a data integration and information sharing tool.
- Snap™ provides real-time information and formats it on the fly.
- Snap™ bridges islands of data in a single web-centric view.
- Snap™ helps identifying inconsistencies amongst registered databases.
- Snap™ does not disrupt or interfere with registered databases.
- Snap™ preserves the current investment and builds on it.
- Snap™ is low maintenance.
- Snap™ does not require data input.
- Snap™ is scalable by design (load-on-demand principle and drill-down approach).
- Snap™ is modular by design (the framework offers seamless integration of legacy and new databases as well as directories and files).