# Profiles Research Networking Software Architecture Guide

**Documentation Version**: June 1, 2012

**Software Version**: ProfilesRNS1.0.0

## Table of Contents

## Introduction

Based on user feedback from Profiles RNS Beta and the rapid emergence of VIVO RDF as a standard vocabulary for describing faculty scholarly activities and institutional resources, Profiles RNS 1.0.0 was rewritten to be more modular and extensible and to take advantage of the Resource Description Framework (RDF) data model. Below is an excerpt from Wikipedia describing RDF:

> The RDF data model is similar to classic conceptual modeling approaches such as Entity-Relationship or Class diagrams, as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions. These expressions are known as triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one way to represent the notion "The sky has the color blue" in RDF is as the triple: a subject denoting "the sky", a predicate denoting "has the color", and an object denoting "blue". RDF is an abstract model with several serialization formats (i.e., file formats), and so the particular way in which a resource or triple is encoded varies from format to format.
>
> This mechanism for describing resources is a major component in what is proposed by the W3C's Semantic Web activity: an evolutionary stage of the World Wide Web in which automated software can store, exchange, and use machine-readable information distributed throughout the Web, in turn enabling users to deal with the information with greater efficiency and certainty. RDF's simple data model and ability to model disparate, abstract concepts has also led to its increasing use in knowledge management applications unrelated to Semantic Web activity.
>
> A collection of RDF statements intrinsically represents a labeled, directed multi-graph. As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representation than the relational model and other ontological models. However, in practice, RDF data is often persisted in relational database or native representations also called Triplestores, or Quad stores if context (i.e. the named graph) is also persisted for each RDF triple. As RDFS and OWL demonstrate, additional ontology languages can be built upon RDF.

## Conceptual Models

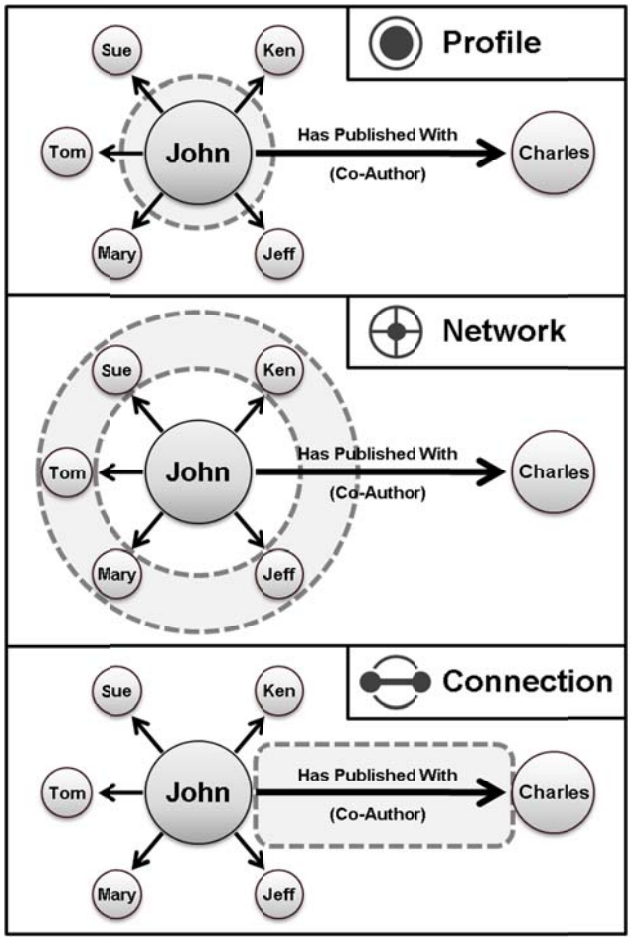
Profiles RNS pioneered two conceptual models, “Profiles-Networks-Connections” and “Passive & Active Networks”, which make the software unique among research networking platforms by (1) providing three ways of viewing and exploring RDF data, and (2) providing two ways of generating new triples.

### Profiles, Networks, and Connections

Profiles RNS introduces a novel concept called Profiles, Networks, and Connections. Consider the RDF triple in which “John”-“has published with”-“Charles”. In other words, John and Charles are co-authors. By itself, this is a simple fact, but for a user of the Profiles website, it leads to three types of more complex questions:

1. Who is John? To answer this question, Profiles contains “profile” pages dedicated to each entity, which lists its various RDF properties. John’s profile page might include properties such as his name, title, affiliation, contact information, photograph, and a research narrative.
2. Who are John’s co-authors? This question explores one of John’s RDF properties, “has published with”, in more depth. A profiles “network” page lists an entity and all the other entities that are connected to it through a particular property, along with additional information about those connections. In other words, it displays all RDF triples that have a given subject and predicate. There are many ways to present a network to users, depending on exactly what they want to know about that network. For example, a geospatial visualization of the network can show whether John’s co-authors are mostly located in one city or spread across different geographical regions, and a temporal view of John’s co-authors show how his collaborations have changed over time.
3. How are John and Charles connected? This question is about the particular co-authorship relation between John and Charles. How many articles have they published together? When were these articles published? Who was the first author on those articles? What were the topics of those articles? Profiles contains “connection” pages, which enable users to view any metadata associated with a single RDF triple. This is especially useful for Profiles’ derived “passive” networks. For example, Profiles automatically creates a “similar research” connection between investigators if their publications have a certain number of Medical Subject Headings (MeSH) in common. The connection page lists those subject headings.

The diagram below illustrates the differences between Profiles, Networks, and Connections. On the Profiles website, the three circular icons indicate to users which of the three types of pages they are viewing. The Profile icon has a large center circle with a thin outer ring around it. This represents the fact that the page is primarily about the entity, with only a sample of the surrounding networks being shown. The Network icon has a small center circle with lines connecting it to the outer ring. This represents the fact that the page is more about the surrounding entities. The Connection icon has two circles connected by a thick line to show that the page is about a particular triple.




**Passive & Active Networks**

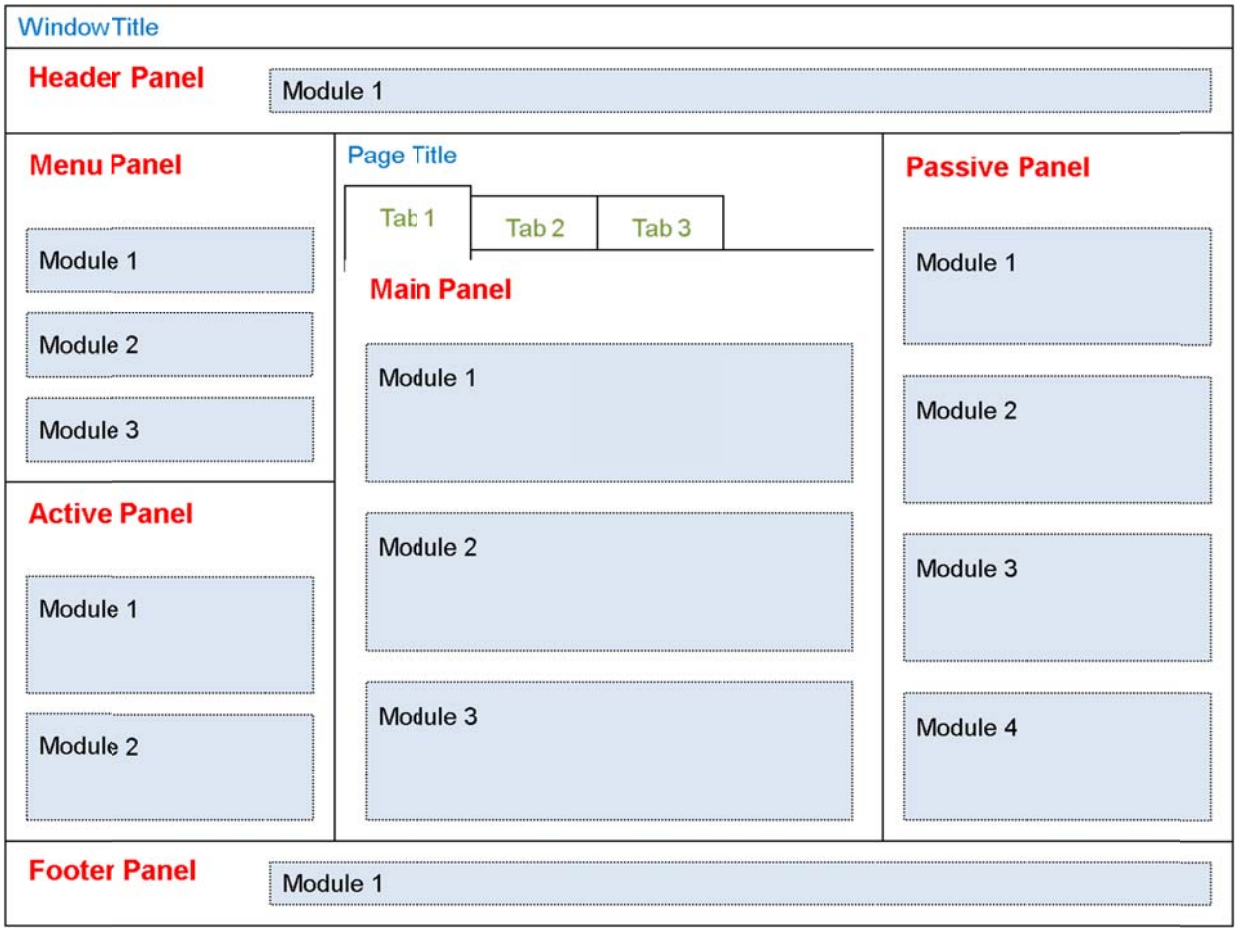
"Passive" and "Active" networks build upon the raw RDF data loaded into Profiles by creating new types of triples. This not only enables the website to be a more useful and exciting tool on day one, but it also allows users to expand its content with information about social networks that only they know.

- Passive networks are automatically derived from existing RDF data, such as co-authorship history, organizational relationships and geographic proximity. It extends these networks by discovering new connections, such as identifying "similar people" who share related keywords. Offering these additional suggestions, Profiles RNS can lead users to unexpected opportunities for collaboration and new sources of expertise.

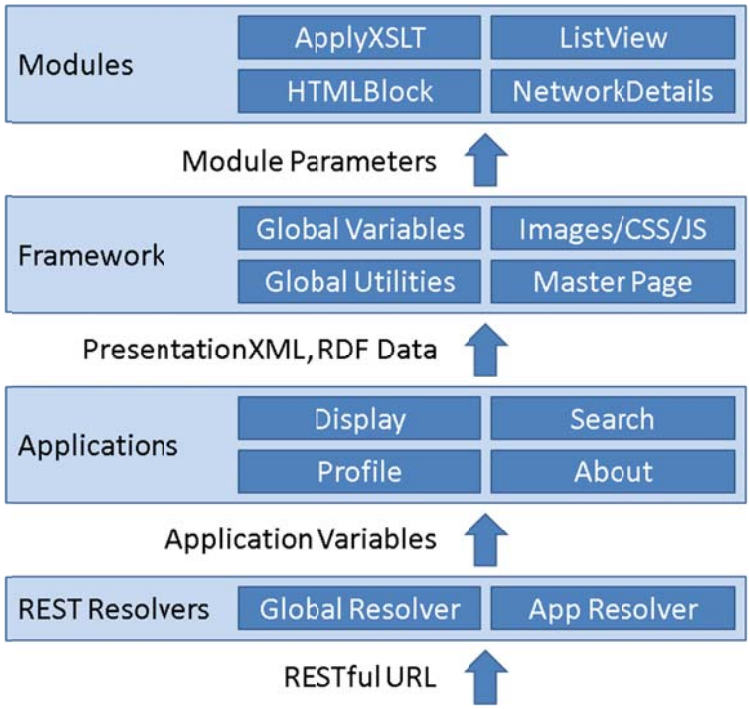
- Users can manually create active networks by identifying advisor, mentor and collaborator relationships with colleagues. Profiles RNS will soon support the OpenSocial standard, which will let researchers use the same types of plug-in collaboration gadgets found on LinkedIn and Google within their active networks.

## Website Architecture

The website's graphical user interface (GUI) is divided into sections called "panels". Each panel contains a list of "modules". Modules are implemented as .NET controls. Most modules function by requesting RDF data and applying an XSLT file to render it as HTML. However, some perform more complex tasks. The website "Framework" creates the HTML "shell" illustrated below. The shell provides the page layout for the panels. The Framework uses the data in an XML file called the PresentationXML to determine the window and page titles and which modules to load into each panel.




Below is a box diagram of the website components. When the server receives a RESTful URL, a set of "resolvers" parse the query string path to determine which application is being requested and what variables should be passed to the application. Each application performs different functions. The Profile application takes a URI and returns an RDF document. The Display application takes a URI and renders it as HTML. Applications that have a user interface component, such as Display, Search, and About send the Framework an application-specific PresentationXML and an optional RDF Data object. The Framework's Master Page creates the HTML shell, which includes images, CSS, and JavaScript files. The Framework uses the data in the PresentationXML to select which modules to load, and it passes parameters listed in the PresentationXML to the modules.




Modules typically obtain data by performing an HTTP POST to a URI that returns an RDF document. That URI can exist anywhere on the internet. If the URI happens to be in Profiles, then the HTTP POST will pass through the REST Resolver and be handled by the Profile application. Thus, the Display application usually calls the Profile application multiple times indirectly via the Modules that it tells the Framework to render.

When a user enters a Profiles URI into the web browser, the following events occur:

1. A 303 redirect sends the user to a URL that corresponds to the Display application.
2. The Display application retrieves a PresentationXML specific to the RDF class of the URI from the Profiles database and sends it to the Framework.
3. The Framework parses the PresentationXML and loads the appropriate modules.
4. The modules request RDF data via an HTTP POST to a URI. The request header will contain an “application/rdf+xml” content type.
5. If it is a Profiles URI, then Profiles will recognize the “application/rdf+xml” content type in the request header and perform a 303 redirect to a URL that corresponds to the Profile application.
6. The Profile application will first see if the RDF data for the requested URI exists in its cache. If so, it will return the data from cache. Otherwise, it will retrieve the RDF from the Profiles database and store it in cache before returning it.
7. The modules apply XSLT to the RDF to render HTML.

Below is a list of applications included in this release of Profiles RNS.

| Name | Description |
|---|---|
| Profile | Returns the RDF document for a URI. |
| Display | Renders a URI as HTML. |
| Search | Search identifies all RDF nodes that have a property whose value matches a search string. It displays a list of those nodes and links to their URIs. Faceting allows users to narrow the search results by type (class group) or subtype (class). Any property can be |

| | |
|---|---|
| | used to sort search results. Search incorporates stemming (to match different parts of speech), removal of stop words (e.g., “the”, “of”), and term expansion through the use of a thesaurus (e.g., “cancer” -> “neoplasm”). |
| About | Displays general information about the Profiles RNS website. |
| SPARQL | This is an interface to test the Profiles RNS SemWeb SPARQL engine. Users can enter an arbitrary SPARQL query and view the results. In the final Profiles RNS 1.0.0 release, this front-end tool will only be available to administrators by default, though the ability to pass SPARQL queries to the SemWeb web services can remain open to the public. |
| Edit | This application allows users to manage the content on their profiles. |

Overall, there is relatively little C# code in Profiles. The complexity of how to store and process RDF exists in the database, and the details of how to render a page are coded as PresentationXML and XSLT files. As a result, little or no C# programming should be needed to configure and customize Profiles. Extending the functionality of Profiles can be done in several ways: (1) adding new classes or properties to the ontology, (2) editing the PresentationXML files for existing applications, (3) creating a new application, or (4) creating a new module.

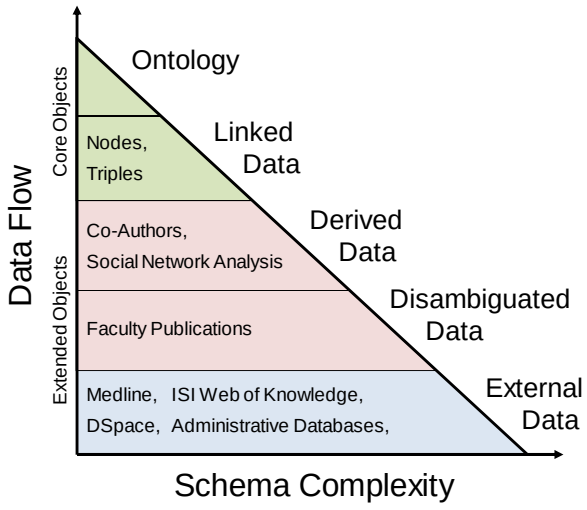
# Database Components

## Schemas Overview

The Profiles database is organized in a hierarchy, with the top two levels defined by schemas. We use a convention where each schema name has two parts, separated by a period. Note that because of the period, the schema name must always be written in brackets. For example, the table [Profiles100].[RDF.Stage].[Log.Job] has the table name "[Log.Job]", it is in the schema "[RDF.Stage]", and it is in the database "[Profiles100]". When writing queries, you do not need to include the database name if you are already connected to the database, but you will always need to include the schema name.

The schemas are divided into two categories: (1) The "core" schemas are essential for Profiles to function properly. Every instance of Profiles must contain the database components in these schemas. (2) The "extended" schemas facilitate the data loading process or provide other types of functionality that are specific to a domain. For example, there are tables to store attributes associated with Pubmed articles; however, these are only relevant in biomedicine. A financial company using Profiles to build networks among its global employees might create different extended schema tables that store information about countries and markets.

There is a general direction of data flow in Profiles, as illustrated in the diagram below, which begins with external data from many different sources. This is then parsed and disambiguated to identify distinct objects and their relationships. Derived data may be obtained from the disambiguated data through computational methods such as social network analysis. Finally, disambiguated and derived data is described as linked data, which consists of nodes and triples. An ontology describes the classes and properties in the linked data. At each step in the data flow, there is a reduction in the number and complexity of database objects, as the model used to represent the data becomes simpler and more generalizable.




## Core Schemas

| Schema | Description |
|---|---|
| [Ontology.] | Contains the RDF ontology as well as global parameters used throughout the |

| | |
|---|---|
| | website. |
| [RDF.] | Contains the RDF nodes and triples specific to an instance of Profiles. |
| [RDF.Search] | Contains database components required for search. |
| [RDF.Security] | Contains information about who can access secure/private nodes and triples. |
| [RDF.SemWeb] | Used to format [RDF.] data so that it can be used by the SemWeb SPARQL engine. |
| [RDF.Stage] | Used by the bulk data loading process to store temporary data before it is loaded into the [RDF.] tables. |
| [User.Account] | Contains information about authorized users of the website. |
| [User.Session] | Contains information about website sessions. A public user of the website will have a session even if she has not logged in and linked the session to a specific user account. |
| [Utility.Application] | Contains functions and procedures that are used in a variety of contexts. |
| [Utility.Math] | Contains mathematical lookup tables and functions. |
| [Utility.NLP] | Contains lookup tables and functions related to support natural language processing for search and other features. |

## Extended Schemas

The Profiles website is structured as a collection of "applications". In the extended schemas, the first part of the schema name corresponds to the primary application that uses it, such as "Profile". Below are the default extended schemas included in Profiles.

| Schema | Description |
|---|---|
| [Module.*] | Used by module * within the Profiles website. Each module has its own schema. This can be used to limit a module's access to other parts of the database if needed. |
| [Profile.Cache] | Contains a processed copy of [Profile.Data] data in a format that is optimized for performance. No original/raw/source data is contained in this schema. |
| [Profile.Data] | Contains original and parsed data for people, publications, resources, and other types of entities and relationships. |
| [Profile.Framework] | Contains database components to resolve URLs and interact with the Profiles application framework. |
| [Profile.Import] | Used for importing external data into Profiles. |

## Core Database Components

Items in yellow are called directly by the website; items in green are used only during the install or data update processes; items in white are used during normal use of the website; items in grey will either be used in the future or they are for debugging or preparing install packages.

### Core Tables

| Table | Description |
|---|---|
| [Ontology.].[ClassGroup] | Lists top-level RDF Class Groups used in search and browse. |
| [Ontology.].[ClassGroupClass] | Maps Class Groups to individual RDF Classes. |
| [Ontology.].[ClassProperty] | Defines which RDF properties should be returned and expanded when data is requested. |
| [Ontology.].[ClassTreeDepth] | Contains the class hierarchy. Used by Search. |
| [Ontology.].[DataMap] | Maps extended schema data to the ontology. |

| | |
|---|---|
| [Ontology.].[InstallData] | Used for bulk import of ontology data during the installation process. |
| [Ontology.].[Job] | Lists steps that are executed during scheduled data updates. |
| [Ontology.].[JobGroup] | Describes related sets of jobs. |
| [Ontology.].[Namespace] | Lists namespaces and their prefixes. |
| [Ontology.].[OWL] | Contains RDF ontologies in OWL XML format. |
| [Ontology.].[Parameter] | Global parameters used by Profiles. |
| [Ontology.].[Presentation] | Templates for how different types of profiles, networks, and connections should be displayed on the website. |
| [Ontology.].[PropertyGroup] | Lists the broad groups of related properties. |
| [Ontology.].[PropertyGroupProperty] | Lists the properties within each group. |
| [Ontology.].[RestPath] | Lists URL prefixes that should be treated as RESTful paths and contains pointers to optional stored procedures that can map paths to actual files. |
| [Ontology.].[Triple] | Presents OWL data as triples. |
| [RDF.].[Alias] | Lists alternative names for particular nodes. |
| [RDF.].[Node] | Lists RDF nodes. |
| [RDF.].[Triple] | Lists RDF triples. |
| [RDF.Search].[History.TopSearchPhrase] | Lists the most commonly used search phrases. |
| [RDF.Search].[Cache.Private.NodeClass] | Improves search performance. Updated automatically from the RDF tables. Lists the classes of all searchable nodes. |
| [RDF.Search].[Cache.Private.NodeExpand] | Improves search performance. Updated automatically from the RDF tables. Indicates what information about matching nodes should be included in the search results. |
| [RDF.Search].[Cache.Private.NodeMap] | Improves search performance. Updated automatically from the RDF tables. Indicates how the nodes matching a search can be increased to include additional related nodes. |
| [RDF.Search].[Cache.Private.NodePrefix] | Improves search performance. Updated automatically from the RDF tables. Contains the first 800 characters of a node's value (the largest size for which an index can be created). |
| [RDF.Search].[Cache.Private.NodeRDF] | Improves search performance. Updated automatically from the RDF tables. Contains an RDF/XML representation of each searchable node. |
| [RDF.Search].[Cache.Private.NodeSummary] | Improves search performance. Updated automatically from the RDF tables. Contains the information displayed in the search results table on the website. |
| [RDF.Search].[Cache.Public.NodeClass] | Same as the Private version, but only includes public data. |
| [RDF.Search].[Cache.Public.NodeExpand] | Same as the Private version, but only includes public data. |
| [RDF.Search].[Cache.Public.NodeMap] | Same as the Private version, but only includes public data. |
| [RDF.Search].[Cache.Public.NodePrefix] | Same as the Private version, but only includes public data. |
| [RDF.Search].[Cache.Public.NodeRDF] | Same as the Private version, but only includes public data. |
| [RDF.Search].[Cache.Public.NodeSummary] | Same as the Private version, but only includes public data. |
| [RDF.Security].[Group] | Lists Security Groups, which are groups of people with certain access rights. |
| [RDF.Security].[Member] | Lists the users that belong to a group. |

| | |
|---|---|
| [RDF.Security].[NodeProperty] | Lists custom security groups for a property of a specific node. |
| [RDF.Stage].[InternalNodeMap] | Used to map RDF NodeIDs to IDs used in the extended schema tables. |
| [RDF.Stage].[Log.DataMap] | Keeps a log of each time the [Ontology.].[DataMap] table is used to convert extended schema data into RDF. |
| [RDF.Stage].[Log.Job] | Keeps a log of each time a step in the [Ontology.].[Job] table is run. |
| [RDF.Stage].[Log.Triple] | Keeps a log of each time data in the [RDF.Stage].[Triple] table is converted to RDF. |
| [RDF.Stage].[Triple] | Used as a temporary table to store information about triples before they are copied to the [RDF.] tables. |
| [RDF.Stage].[Triple.Map] | Keeps a record of which TripleIDs were created from StageTripleIDs. |
| [User.Account].[DefaultProxy] | Lists users who can edit other user's content based on affiliation. |
| [User.Account].[DesignatedProxy] | Lists users who have been designated by other users to edit their content. |
| [User.Account].[Relationship] | Lists "active network" relationships. |
| [User.Account].[User] | Lists authorized users of Profiles. |
| [User.Session].[Bot] | Contains a list of UserAgents known to be web crawlers. |
| [User.Session].[History.ResolveURL] | Contains a log of pages accessed during a session. |
| [User.Session].[Session] | Contains basic information about each website session. |
| [Utility.Math].[N] | A list of sequential integers starting at zero. |
| [Utility.NLP].* | Various tables used by natural language processing (NLP) routines. |

Core Views

| View | Description |
|---|---|
| [Ontology.].[vwOwlTriple] | Extracts triples from [Ontology.].[OWL]. |
| [Ontology.].[vwPropertyTall] | Summarizes properties in [Ontology.].[Triple]. |
| [Ontology.].[vwPropertyWide] | Summarizes properties in [Ontology.].[Triple]. |
| [RDF.].[vwClass] | Summarizes classes from data in [RDF.].[Triple]. |
| [RDF.].[vwPropertyTall] | Summarizes properties from data in [RDF.].[Triple]. |
| [RDF.].[vwPropertyWide] | Summarizes properties from data in [RDF.].[Triple]. |
| [RDF.].[vwTripleValue] | Lists the node values of the subject, predicate, and object of the triples in [RDF.].[Triple]. |
| [RDF.Search].[vwCache.Public.NodeMapValue] | Indicates related nodes for search. |
| [RDF.SemWeb].[vwHash2Base64] | An indexed view that speeds the Base64 conversion of the node hash for SemWeb. |
| [RDF.SemWeb].[vwPrivate_Entities] | Lists all entity nodes in [RDF.].[Node] in the format used by SemWeb. |
| [RDF.SemWeb].[vwPrivate_Literals] | Lists all literal nodes in [RDF.].[Node] in the format used by SemWeb. |
| [RDF.SemWeb].[vwPrivate_Statements] | Lists all triples in [RDF.].[Triple] in the format used by SemWeb. |
| [RDF.SemWeb].[vwPublic_Entities] | Lists only public entity nodes in [RDF.].[Node] in the format used by SemWeb. |
| [RDF.SemWeb].[vwPublic_Literals] | Lists only public literal nodes in [RDF.].[Node] in the format used by SemWeb. |
| [RDF.SemWeb].[vwPublic_Statements] | Lists only public triples in [RDF.].[Triple] in the format used by SemWeb. |

| | |
|---|---|
| [Utility.Application].[vwDatabaseCode] | Lists the code for stored procedures and functions. |
| [Utility.Application].[vwDatabaseObjects] | Lists all database objects. |
| [Utility.Application].[vwRandomSort] | Random number generator. |

Core Procedures

| Procedure | Description |
|---|---|
| [Ontology.].[ChangeBaseURI] | Facilitates moving Profiles from one environment to another by changing the base URI path throughout the database. |
| [Ontology.].[ConvertOWL2Triple] | Parses the [Ontology.].[OWL] data and stores the triples in [Ontology.].[Triple]. |
| [Ontology.].[ConvertTriple2OWL] | Creates a record in the [Ontology.].[OWL] table by combining triples in [Ontology.].[Triple]. This is used only when building a Profiles install package. |
| [Ontology.].[CreateInstallData] | Combines all non-OWL ontology data into a single XML object. This is used only when building a Profiles install package. |
| [Ontology.].[GetClassCounts] | Returns the number of nodes associated with Class Groups and their classes. This is used primarily for search and browse in the website. |
| [Ontology.].[LoadInstallData] | Parses the InstallData.xml file and populates all non-OWL ontology tables. |
| [Ontology.].[ResolveURL] | Maps RESTful URLs to their actual file names. |
| [Ontology.].[UpdateCounts] | Updates ontology tables with counts from the [RDF.] tables. |
| [Ontology.].[UpdateDerivedFields] | After the Ontology is loaded into the [RDF.] tables, this procedure updates [Ontology.] tables with the NodeIDs that were created during the loading process. |
| [RDF.].[DeleteNode] | Deletes a node and its associated triples. |
| [RDF.].[DeleteTriple] | Deletes a triple and its dependencies. |
| [RDF.].[GetDataRDF] | Returns RDF/XML for a profile, network, or connection. |
| [RDF.].[GetPresentationXML] | Returns the PresentationXML template associated with a profile, network, or connection. |
| [RDF.].[GetPropertyList] | Combines DataRDF and PresentationXML. |
| [RDF.].[GetPropertyRangeList] | Returns classes that can be linked to by a property. |
| [RDF.].[GetStoreNode] | Creates or updates a single node. |
| [RDF.].[GetStoreTriple] | Creates or updates a single triple. |
| [RDF.].[SetNodePropertySecurity] | Changes the security group for a single node and property. |
| [RDF.Search].[Cache.Private.UpdateCache] | Updates search cache tables. Improves performance of search. |
| [RDF.Search].[Cache.Public.UpdateCache] | Same as the private version, but only queries public data. |
| [RDF.Search].[GetNodes] | Search for nodes based on keywords and other parameters. |
| [RDF.Stage].[LoadAliases] | Populates the [RDF.].Alias table. |
| [RDF.Stage].[LoadTriplesFromOntology] | Loads data from the ontology into [RDF.Stage].[Triple]. |
| [RDF.Stage].[ProcessDataMap] | Creates nodes and triples from extended schema data. |
| [RDF.Stage].[ProcessTriples] | Loads data from [RDF.Stage].[Triple] into [RDF.].[Node] and [RDF.].[Triple], creating new nodes and triples as needed. |
| [RDF.Stage].[RunJobGroup] | Runs a series of data load or update steps. |
| [User.Session].[CreateSession] | Creates a new session. |

| | |
|---|---|
| [User.Session].[UpdateSession] | Updates the data associated with an existing session. |
| [Utility.Application].[LICENCE] | Contains the open source license for the software. |
| [Utility.Application].[LoadXMLFile] | Imports data from an external file into a specified table and column. |
| [Utility.NLP].[UpdateThesaurus] | Populates the [Utility.NLP].[Thesaurus] table. |

Core Functions

| Function | Description |
|---|---|
| [Ontology.].[fnGetClassTree] | Returns a hierarchical list of classes defined in [Ontology.].[Triple]. |
| [Ontology.].[fnGetPropertyTree] | Returns a hierarchical list of properties defined in [Ontology.].[Triple]. |
| [RDF.].[fnTripleHash] | Returns the SHA1 hash of a triple. |
| [RDF.].[fnURI2NodeID] | Returns the NodeID of a URI. |
| [RDF.].[fnValueHash] | Returns the SHA1 hash of the language, data type, and value of a node. |
| [RDF.SemWeb].[fnHash2Base64] | Returns the Base64 encoding of a binary hash value for SemWeb. |
| [Utility.Application].* | Various functions used to encode/encrypt/decode/decrypt data or convert data from one type to another. |
| [Utility.NLP].* | Various functions used by natural language processing (NLP) routines. |

## Extended Database Components

Many of the extended database components in Profiles RNS 1.0.0 map to objects in Profiles RNS Beta database. However, names have been changed to match the conventions used throughout the application.

# Acknowledgements

Profiles Research Networking Software was developed under the supervision of Griffin M Weber, MD, PhD, with support from Grant Number 1 UL1 RR025758-01 to Harvard Catalyst, The Harvard Clinical and Translational Science Center from the National Center for Research Resources and support from Harvard University and its affiliated academic healthcare centers.

Harvard Development Team

The software implementation is led by the Harvard Medical School Information Technology Department. The current and past members of the development team include Nick Benik, Niraj Desai, Paul Gomez, John Halamka, Ken Huling, Shashank Jain, Melissa Kenny, Kevin Laitinen, Kellie Lucy, Krishna Nellutla, James B. Norman, Rob Piscitello, George Rakauskas, Jeff Rosen, Michele Sinunu, Franco Valentino, Marlon Violette, Griffin Weber, and Steve Wimberg.

UCSF Development Team

The UCSF Profiles team includes Mini Kahlon, Eric Meeks, Kristine Moss, Rachael Sak, and Leslie Yuan. UCSF has developed innovative promotional strategies for research networking, assisted with quality assurance, and are adding OpenSocial support to Profiles RNS. Mini Kahlon is co-chair with Griffin Weber of the National CTSA Research Networking Group, which is leading the efforts to create a national pilot to demonstrate interoperability among different research networking platforms.

Recombinant Data Corp.

The Profiles RNS team at Recombinant Data Corp. includes Kimber Barton, Nick Brown, Peter Emerson, Dan Housman, Mike Klumpenaar, Mark Mischke, Matvey Palchuk, and Nancy Pickard. Recombinant provides commercial support for Profiles RNS, hosts publication disambiguation services, develops administrative tools for Profiles RNS, and writes documentation (including portions of this install guide) and marketing materials.

Profiles RNS Users Group

We thank the member institutions of the Profiles RNS Users Group for their willingness to be early adopters of the software and their continued feedback. For a list of member sites, please visit the Community page on http://profiles.catalyst.harvard.edu.

## More Information

For more information about Profiles RNS, please visit

http://profiles.catalyst.harvard.edu

The Harvard development team can be reached at profiles@hms.harvard.edu. We will try to reply promptly, though we cannot guarantee that we will be able to answer all questions.

Commercial support options are available through Recombinant Data Corp. Harvard has no financial relationship with Recombinant, but we recommend them as an Authorized Service Provider for Profiles RNS. For more information, contact Recombinant at results@recomdata.com or call (617) 243-3700.