

A Help Desk Knowledge Database: The First Year

Geoffrey Marsh, Division of Computer Research and Technology, NIH

geoff@helix.nih.gov

February 7, 1997

Corrected/Updated: February 11, 1997

Introduction

Why a Knowledge Base? What are the benefits that a Knowledge repository (also known as a Knowledge Database or an Organizational Memory Engine) provides to an organization?

A Knowledge Base is a central repository of information that can be used both inside and outside an organization. Inside the organization it can be used to help customers who call in looking for technical assistance and support. Outside the organization it can be forward-deployed to help prevent calls from coming in, thereby reducing the load on a help desk staff and allowing their time to be more productively used.

A Knowledge Base allows for consultants and, possibly, customers, to view data and information in a standardized format, thereby enhancing retrieval and retention of information by the user. One of the downsides of trying to use the World Wide Web as a huge Knowledge Base is the inconsistency of the data and the difficulty in locating it. Part of the job of the Knowledge Base administrator is gathering that data and placing it in the database for the users.

A Knowledge Base allows for complete local control over the information and data that is being disseminated. It can be reviewed and audited for accuracy and content, and may be edited for timeliness of content.

Over the past year, we have attempted to start building such a Knowledge Base for use by our help desk consultants. The purpose of this white paper is to describe what our help desk has learned over the past year in the hopes that others might learn from our successes (and mistakes). In return, I hope that others will share with us their experiences and that we might learn from each other. Understand that I am looking to provide this information in the hope of initiating dialog in the help desk and technical support community. This paper is not the last word on the subject, it is merely a statement of how we did it and where we are one year into the process.

Organizational History

The Division of Computer Research and Technology (DCRT) is the part of the National Institutes of Health (NIH) which provides computer assistance and research to enhance the biomedical endeavor with which NIH is charged.

In January 1994, DCRT's Customer Services Branch (CSB) opened the Technical Assistance and Support Center (TASC), a consolidated help desk to provide technical support to customers across the NIH campus and in non-NIH facilities which make use of NIH's centralized computing resources. TASC provides assistance or referrals for virtually all facets of computing at NIH and allows NIH customers to call one number (301-594-3278, or 4-DCRT in NIH parlance) regardless of their question or platform.

The computing environment at NIH is as diverse as the research itself. TASC receives calls on a wide variety of platforms (MVS, Unix Workstation, Unix Supercomputer, Macintosh, PC, etc) and a wide variety of applications (from office and secretarial applications to batch processes to highly specialized scientific software).

For the purposes of tracking and logging customer calls, TASC uses Remedy Corporation's Action Request System ("Remedy"). We (CSB) have customized Remedy to include schemas for customer information (the Customer Database), problem tracking (the Service Ticket) and other internal functions. The Knowledge Base used by TASC, which is the focus of this paper, resides in Remedy and encompasses several schemas. While this paper will discuss the overall structure of the Knowledge Base in Remedy, implementation details specific to Remedy are omitted. If you wish to obtain a more detailed understanding of the Remedy implementation, please contact the author directly.

Knowledge Base History

Our first attempts at setting up a Knowledge Base for TASC were made about nine months after the help desk was established, in late September of 1994. Our Remedy Administrator created a schema called Knowledgebase and I (the author, Geoffrey Marsh) created a couple of test records in it, fine-tuning as we went along. After a couple weeks of experimentation, on September 23rd of that year we opened up the first version of the Knowledge Base for everyone to use, query, and update.

Included in the mail announcing the availability of the KB (as we have since come to know and love the later incarnations) were some very specific instructions.

We gave the users the ability to spawn existing Service Ticket records into the KB, as well as the ability to create brand new records. There was a field called "Keywords" into which the consultant was supposed to enter any relevant words or terms that might not otherwise appear in searchable fields. Additionally, we called upon all full-time help desk consultants to enter at least two records into the database each week.

Well, sadly, this wonderful idea lasted a total of 19 days. I asked our Remedy Administrator to disable the schema on October 12 and he permanently removed it about a month later. There were only about a dozen records in the database and they were mostly poorly written copies of solutions from Service Tickets. They were virtually unsearchable and useless to anyone looking for helpful information from the KB.

Thus followed a hiatus of over a year before the current incarnation of our system was launched. However, in the interim, a grass-roots effort sprang up which provided the KB impetus and helped speed its acceptance.

One of our help desk consultants began maintaining a small rolodex-style referral and information file in a Windows Cardfile. She would update it regularly and distribute it to all help desk staff monthly. This small file slowly grew until it began to push the limits of the capability of that technology. However, before the absolute limits were reached, she left for another position, and the Cardfile became the responsibility of the author.

At this point, in January 1996, I released one more revision of the Cardfile, and then moved the data into a new Remedy database we had designed.

On February 16, 1996 the Knowledge Base, at that time called the Card File to help maintain continuity, was released to the help desk consultants. A few weeks later, at the request of several members of our staff, it was officially renamed the Knowledge Base, or KB.

In the year since the initiation of the Knowledge Base as an entity unto itself, our organization has gained several key bits of experience that have greatly contributed to the success of the project.

Shortly after the launch of the KB in Remedy, I spent time at the University of Maryland Library searching for information on creating and maintaining knowledge databases. While there was much there in the abstract, there was very little that offered us, the beginning Knowledge Base administrators, designers, and users, much hope or guidance. Even more frustrating, attempts to locate classes and conferences on the subject turned up dry as well. On the few occasions where there were events of relevance, I discovered that our Knowledge Base was usually far ahead of the crowd and that most folks looked at me with a glassy look in their eye and said "Yeah, we'd like to do that." Usually they would then ask if we had written a white paper on what we had done. The same thing would happen when I would post messages to help desk forums on the web or to LISTSERV lists oriented to help desks. Everyone who found out what we had done here asked for a white paper, which I am at last supplying.

The Structure of the Knowledge Base

Our Knowledge Base implementation has a three-part structure, made up of the Knowledge Database itself, an Administrator's Database and an Audit Trail Database. Schematically, the KB looks something like the following:

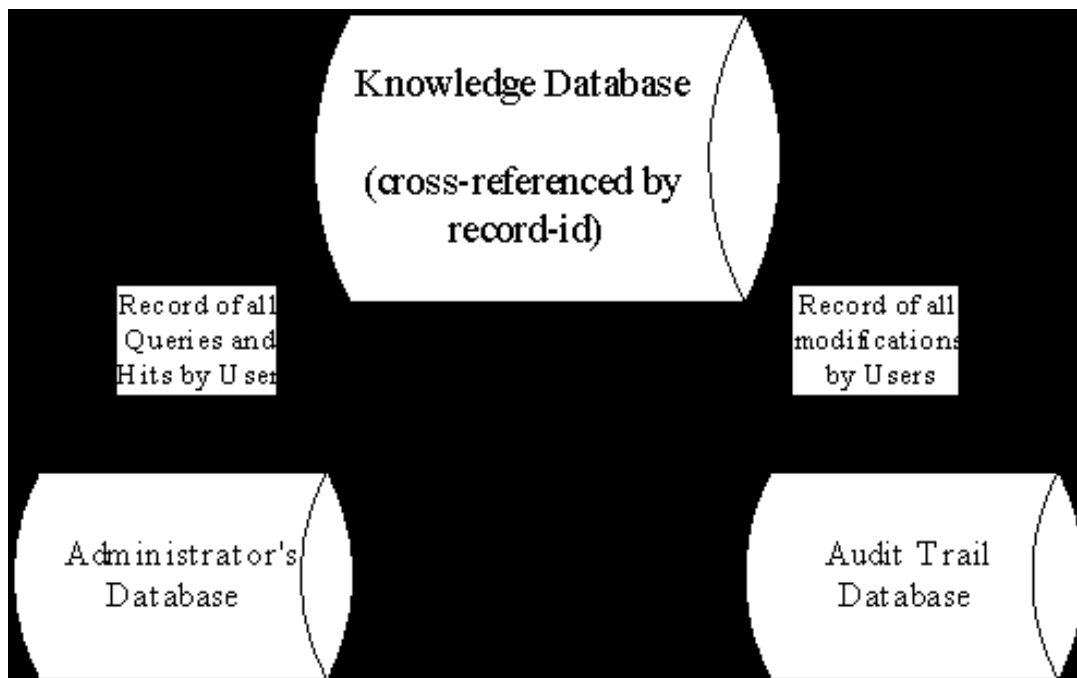


Figure 1: Knowledge Base Structure

The knowledge information itself resides in the KB Database at the top of the diagram, while the other two databases act as "behind the scenes" repositories of important information.

The Administrator's Database, as I have called it, contains a record of all queries and hits against the knowledge data. Every time a KB user types in a query and runs it against the database, a copy of that user's query is recorded in a record in this database. Subsequent to the query, if the user selects a record from a pick-list of possible choices, that action is recorded as well. Thus the Administrator's Database contains a complete record of what the users of the KB are looking for, and what they are finding. As is discussed later in this paper, this is critical to helping the KB administrator (or Gatekeeper) determine what information needs to be in the KB.

The Audit Trail Database (or AT) contains a record of every modification action performed against "auditable" data. By auditable I mean actual knowledge information that users will use in solving problems and routing customer calls. Every time a user of the KB modifies a field and applies that modification to the database, a copy of both the old and new information is spawned into the Audit Trail. As is discussed later in this paper, keeping an Audit Trail is an essential function once you begin to open the Knowledge Base up and allow all users to update the data.

Both the Administrator's Database and the Audit Trail Database are cross-referenced back to the Knowledge Database via the KB record id (what I call the "Card Number").

Information Fields

A Knowledge Base must be made up of knowledge, of course. Here is some of the information our organization has found useful to keep in our KB. Each of these fields is filled in based upon applicability for that record. Some records have very few of these filled in, others have nearly all.

- Header or Title for each record. This is what comes up in a listing of matches (or "hits") from a query of the KB.
- Contact Name. Who is responsible for the product or service described in this record (sometimes referred to in our environment as a "card", a throwback to the old CardFile days)
- Phone: The phone number of the contact.
- Fax: The fax number of the contact.
- Email: The email address of the contact.
- URL: Any websites containing information relevant to the product or service described in the card.
- Files: Any files on a server or available via anonymous FTP with relevant information.
- FAQs: Any collections of "Frequently Asked Questions" with relevant information.
- Text References: Any documentation or other textual hardcopy material available that pertains to the subject at hand.
- Platform: The platform on which a product or service resides.

- **Information:** The meat of the Knowledge Base entry. This is a large field in our KB containing the "answers".
- **Info Distribution:** This field has three choices: Internal Only, NIH Only, and Public. It indicates how the information may be disseminated to customers.
- **Info Usage:** Currently there are three choices here: Card File, Matrix, and Answer Base, but this field is being rethought.

Additionally, there are two other important fields related to KB searching which will be discussed again in the next section. They are:

- **Synonyms:** Other terms used to describe the product or service, or common/popular misspellings.
- **Keywords:** A selected list of key terms that are used to describe each record.

Visually, the window with this information in it looks like the following:

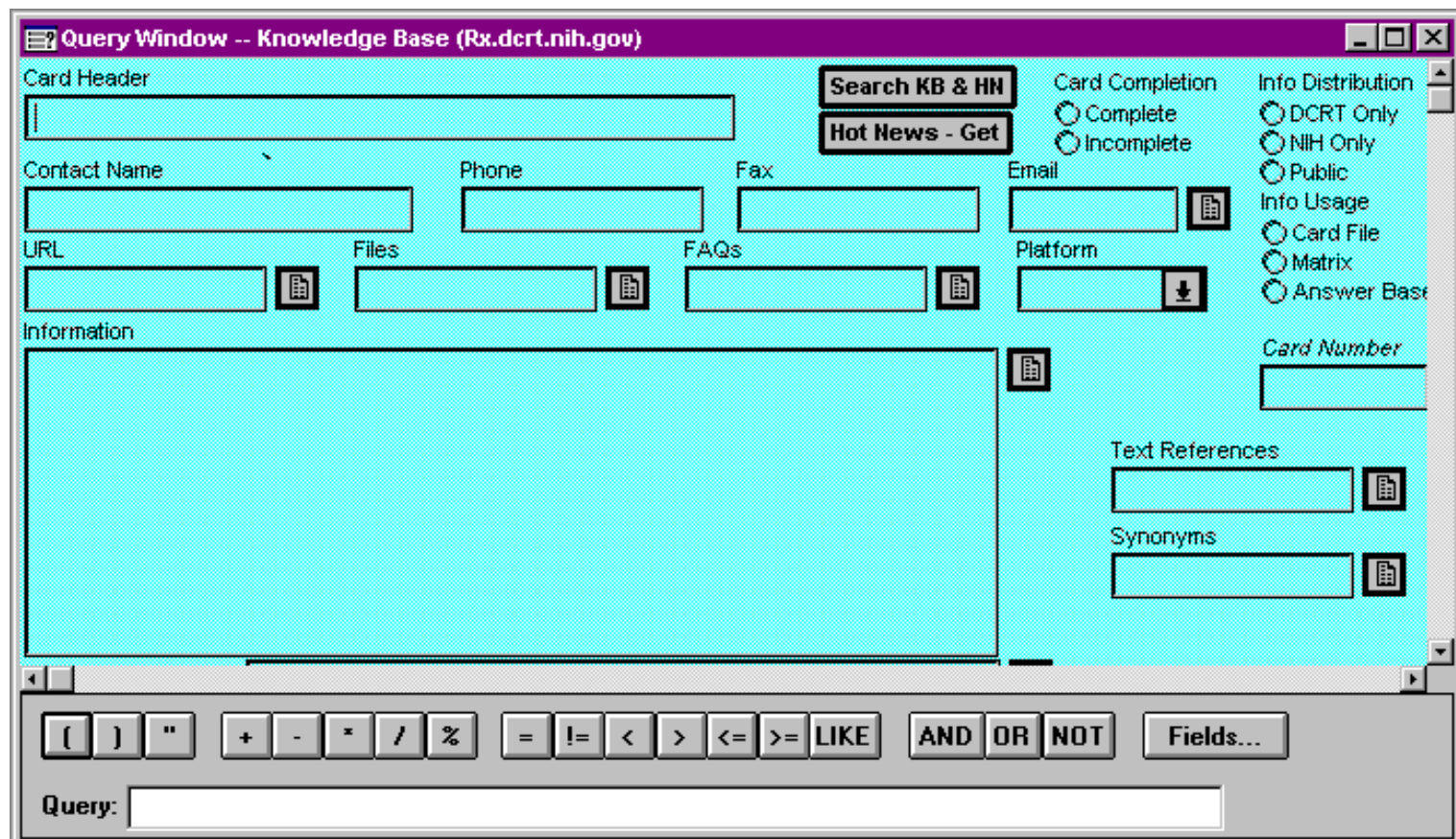


Figure 2: Knowledge Base Query Window

Searching and Results

As with any database package, the ability to query is limited by the search capability of the software. Our chosen solution, Remedy, has both advantages and drawbacks.

When searching the KB we only truly have the ability to search for *exact character matches*. Remedy has some wild card capabilities which we have not, at this time, fully explored. It does not have the ability to perform fuzzy-logic searches.

Our search strategy is implemented based upon the user pulling up a blank KB query screen and using that as the launch point for their search. The very topmost field on the query screen is called "Card Header". In this field, the user of the KB types their search string and hits the "Return" key on their keyboard. The database then performs a case-*insensitive* search of three fields for an exact textual match (including any embedded blanks). Those three fields are the "Card Header" field, the "Synonyms" field (discussed at more length later), and the "Keywords" field. If the user includes any wild cards in the search, these are interpreted as such by Remedy and are applied to the search criteria.

After searching these fields, the Knowledge Base then displays a Remedy Query List window showing one line for each hit.

Query List -- Knowledge Base (Rx.dcrtnih.gov)						
KB0000523	CANDYLAN - Web URL	Tiddi K...at	5-13887		Card F	NIH
KB0000258	DCRT Web Initiative	See URL			Card F	DCR
KB0000143	DCRT Web Services; H				Card F	DCR
KB0000425	HHS Web Site				Card F	DCR
KB0000546	Mantis Web Sites				Card F	DCR
KB0000525	SILK				Card F	Pub
KB0000125	WIG - Worldwide Web				Card F	NIH

7 Entries Polling Off (January 28, 1997 10:50:43 AM) 0 Selected

Figure 3: Knowledge Base Query Results

Included in the display (from left to right) is the following information:

- Card Header
- Contact Name
- Contact Number
- Info Usage
- Info Distribution

The user of the KB can then double-click on the record they wish to view, and it will be displayed on their screen.

Modify Individual -- Knowledge Base (Rx.dcrtnih.gov)

Card Header: Card Completion: Complete Incomplete Info Distribution: DCRT Only NIH Only Public

Contact Name: Phone: Fax: Email:

URL: Files: FAQs: Platform:

Information

SILK is an MVS product that allows folks to make any dataset on the mainframe available via the web. The charge to a user's account is a flat rate of \$20 per month for each account/initials combination with web-available datasets (in addition to standard mainframe storage charges).

Questions should be referred to _____ (internal use only)

To use SILK, the user saves the dataset under their own account and initials with the name:

Card Number:

Text References:

Synonyms:

Information Request:

Number 1 of 1

Figure 4: Knowledge Base Record

Useful Functionality

The Knowledge Base has several capabilities that we have found enhances its usefulness to our help desk consultants.

- **Mail to customer:** KB users have the ability to send an email message containing a Knowledge Base record to a customer. There are certain restrictions on this functionality, such as the fact that a user can not send a record flagged as internal only. However, aside from that, several of our help desk consultants have found this useful and use it regularly.
- **Mail to self:** Another emailing function in the KB is the ability for a user to have a record emailed to themselves. This is used for several reasons. First it can be useful if a record is long and you want to view it in email or print it out. Second it is helpful if you want to send a restricted-access record to a customer, but first want to edit out the portions that are not for general distribution. Finally, it is helpful if the user wants to send a record update to the Administrator, they can get a copy of the record and edit it, rather than typing in new information.
- **Printing:** It goes without saying that it is helpful to be able to print out a copy of a record, especially if you want to fax a copy of the information to a customer. The ability to fax directly from within the KB is under investigation currently.
- **Display all records:** As a learning tool, or for browsing, getting a sorted, alphabetized listing of all records can be useful.
- **Closing Service Ticket to KB:** User's have, though in a suboptimal manner, the ability to close an open customer service record to an existing entry in the KB. Though not heavily used at this time, as the usefulness of the KB grows I hope to use this more and more to cross-tabulate usage.
- **Thank you messages:** The Administrator of our KB has the ability to fill in a field with a userid and a short message and have a thank you note automatically be emailed to that KB user. This is used to thank users for their input or to let them know that information that was not in there when they searched before has been added by the Administrator.

Key Points

In addition to the database structure and functionality already discussed, there are several key points that we have learned in the past year. I will outline them here and then explain each one in some detail further on.

Some of the key points we have learned are as follows:

- I. *Gatekeeping:* A Gatekeeper is essential
- II. *Historical Data:* The Gatekeeper (and Administrator, if they are different) must know, must have a record, of what the users are looking for.
- III. *Potayto or Potahto:* Search flexibility, whether through fuzzy logic or a synonyms/thesaurus/lexicon function is necessary.
- IV. *Only nothing is too little:* No matter how little you know, get it in there.
- V. *Tracking vs. Resolving:* Copying resolutions directly from the problem tracking system has serious pitfalls.
- VI. *Setting the Example:* Users should have the ability to update the database, but before they do they should have enough experience in using the database that they know the tenets and standards of the Knowledge Base.
- VII. *Big Brother is Watching:* All information must be fully audited and past information quickly retrievable (not just via standard backup mechanisms but by a separate audit trail).
- VIII. *Document your processes!*

I will discuss each of these in more detail below.

I. Gatekeeping

What is a Gatekeeper?

A Gatekeeper is the person responsible for administering the data within the Knowledge Base. They may or may not be the individual actually maintaining the database or running the Knowledge Base software, but they must be a person with enough experience in the environment that they can write and structure data, intelligently and efficiently.

By intelligently, I mean that they can present information in the manner most appropriate for the information, whether it be in prose form, in a list, or whatever. By efficiently I mean they can present the information without a lot of extraneous clutter.

Keep in mind that the user of the KB is probably searching the database while they have a customer on the other end of the phone, or on the other side of the desk, or maybe even looking over their shoulder. If they pull up a Knowledge Base record that takes five paragraphs just to say "Try pressing ENTER and if that fails reboot the PC" then they are either going to have to sit there for five minutes to read all that or they are going give up and go ask someone.

Additionally, the Gatekeeper should be keeping his or her finger on the pulse of the Knowledge Base. How do they do that? By monitoring the things outlined in the rest of this white paper and taking corrective action as necessary. Most of the rest of this paper will talk about the duties of the Gatekeeper.

II. Historical Data

Recently I did a review of "successful" queries of the Knowledge Base for a three week period; that is I looked at the queries which located records and resulted in the user of the KB viewing records to solve a problem. I found that 33% of the successful queries, or "hits", retrieved data that had been entered into the database because someone else had queried on that same subject at an earlier time and gotten nothing.

One of the most important bits of functionality incorporated into our Knowledge Base is the ability to see what folks are looking for. If you don't know what they are looking for, how can you stock the database with the information that they need?

One way of keeping the Gatekeeper and/or Administrator "in touch" with the users, or the front-line help desk folks, is to have them do time on the help desk themselves. This is an excellent idea, and some time each week or month should be spent in the trenches. But Gatekeeping can be a full-time job, and if you want your Gatekeeper to maintain and expand the KB, they can not spend all their time on the desk or listening in to the users and consultants. Therefore, the next best thing is to keep a running log of what they are looking for and what they are finding.

As already mentioned, in our Knowledge Base, we have an Administrator's schema or table. Every time a user of the KB performs a query, the content of that query is spawned into a record in this schema, along with user information and a time stamp.

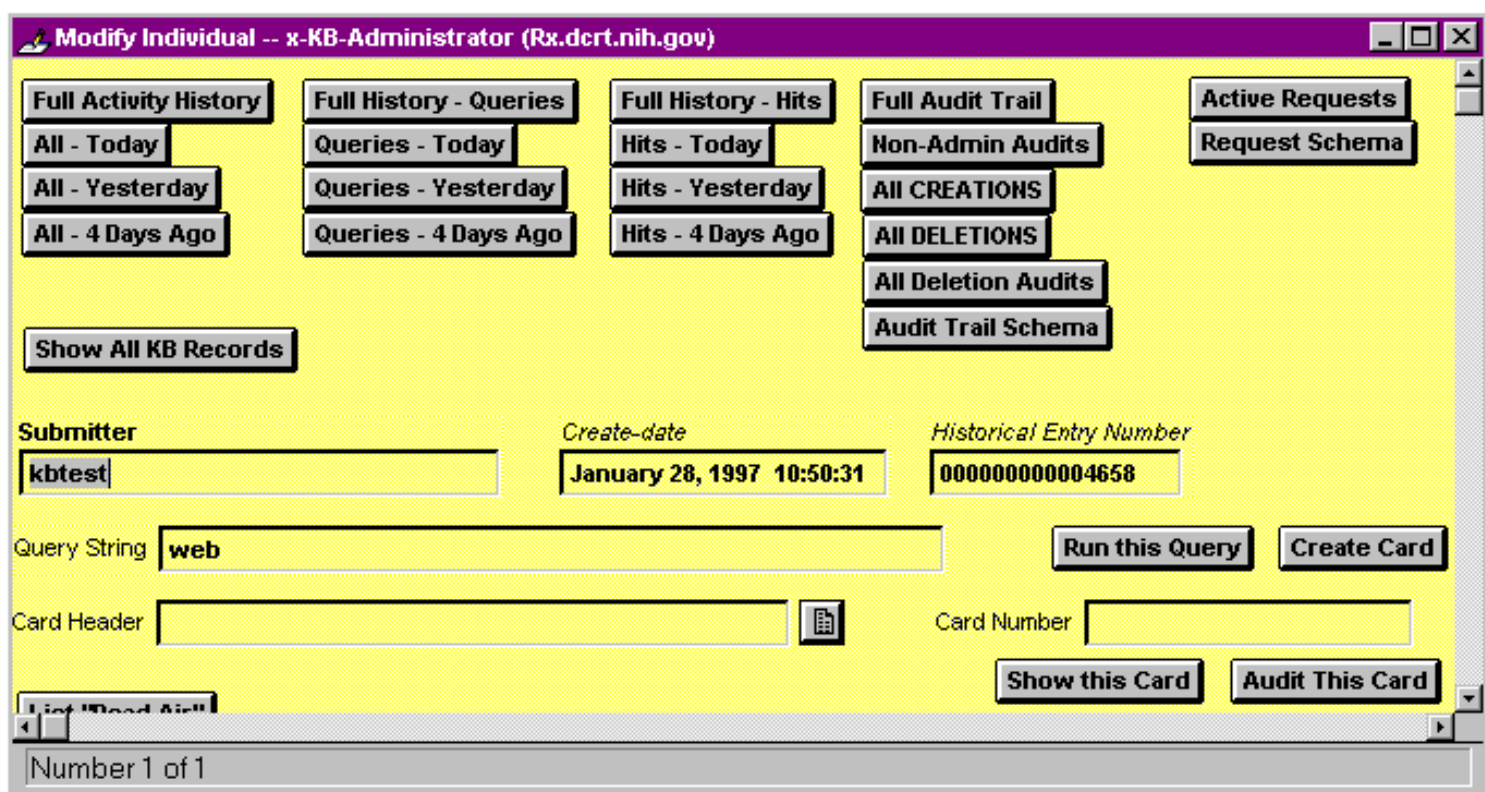


Figure 5: Administrator record showing a user query

And then, every time a user views one of the "hits" or possible solutions, another record is spawned into this schema.

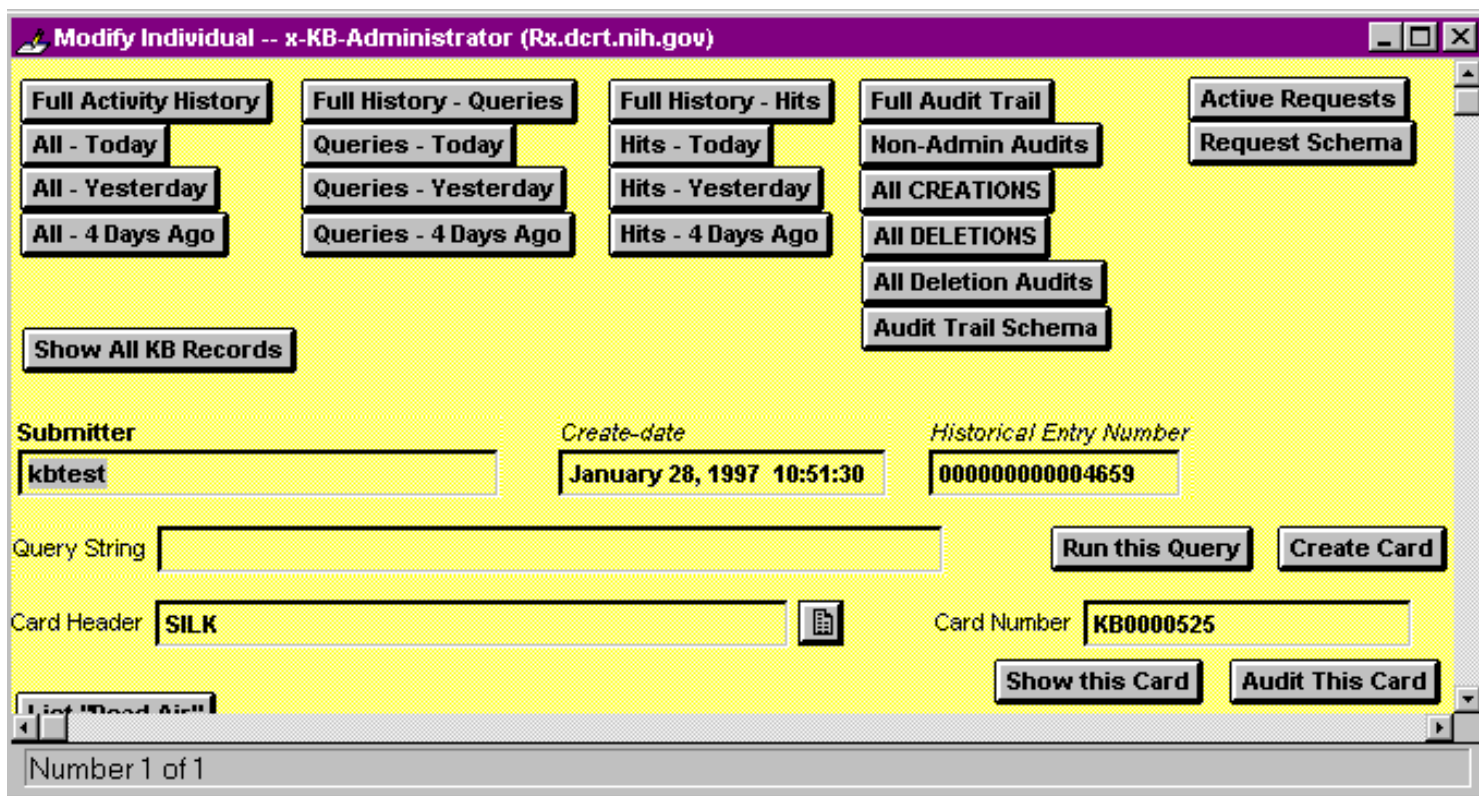


Figure 6: Administrator record showing a user "hit"

Entry-ID	Date/Time	User	Query	Record Header
00000000004667	01/28/97 11:29:	cohoon		BRMUG (Referral)
00000000004666	01/28/97 11:29:	cohoon	brmug	
00000000004665	01/28/97 11:15:	jimi		Electronic Forms Use
00000000004664	01/28/97 11:15:	jimi	forms	
00000000004663	01/28/97 11:11:	jimi		Electronic Forms Use
00000000004662	01/28/97 11:11:	jimi	forms	
00000000004661	01/28/97 10:52:	kellisoj		News Server (DCRT Se
00000000004660	01/28/97 10:52:	kellisoj	news	
00000000004659	01/28/97 10:51:	kbtest		SILK
00000000004658	01/28/97 10:50:	kbtest	web	
00000000004657	01/28/97 10:50:	kbtest	dcrtn	
00000000004656	01/28/97 10:49:	kbtest	nih	
00000000004655	01/28/97 10:49:	kbtest	eps	
00000000004654	01/28/97 10:49:	kbtest	dna	
00000000004653	01/28/97 09:44:	wrightm	leased	
00000000004652	01/28/97 09:41:	janowitr		BPA - Blanket Purcha
00000000004651	01/28/97 09:41:	janowitr	bpa	
00000000004650	01/28/97 09:41:	herbertb	KERMIT BREAK KEY	
00000000004649	01/28/97 09:40:	Ghebele		Parachute

Figure 7: Listing of a morning's KB activity in the Administrator's Database

When viewed on the screen, the data is a running log of who is looking for what and what they are finding, as shown in Figure 7. The listing, displayed in reverse chronological order, shows, from left to right, an entry-id (this may be ignored), a date and time stamp, and the user. The next column displays queries made by users, so anytime this column is filled in, the KB has been queried. The next column displays the header of any records that are being selected from a pick-list as a result of a

query.

So by correlating the the information in the listing, you can see clearly what folks are looking for and what they are finding (or not finding).

At the beginning of each day, I print out the "misses" and go back over them, adding new information, creating new links, and so on. Frequently, I will go back to the consultant and ask them what they remember about the user's question and how I can better structure the information in the Knowledge Base. This has been invaluable in growing and improving the KB, as the statistics quoted at the beginning of this section show.

Another benefit of this information is that it gives you some raw usage numbers through which you can track whether folks are using the database, who is using it, what's hot, and what's not. This can prove useful in deciding how the Gatekeeper needs to spend their time. If 75% of the queries are about Product X and 25% are about Product Y, this should tell the Gatekeeper where to focus her or his efforts.

III. Potayto or Potahto

One morning I sat down to do my daily review of what folks were looking for and I saw where one of my KB users had queried on the string "project control" while the person who sat just across from her had queried on "control" about 45 seconds later. I couldn't help but wince because I knew what had happened. The office in our organization that handles new user accounts, billing, and similar issues, goes by the name "Customer Accounts". However, many years ago, before we were restructured in the early 1990s, that office was known as "Project Control". Obviously, a long-time customer had called up the help desk and asked for the Project Control Officer. The woman who had taken the call was new and had no idea who this person was so she had turned to the KB. When the Knowledge Base came up dry, she turned to her friend across the way and asked her, who also queried the KB.

Luckily, the story had a happy ending, as I later found out, and the customer did end up with the right office. But the story illustrates the key point that any Knowledge Base is only as good as the search capability.

I once read a statistic that said there is only a 20% chance that two people will use the same terms to describe the same situation. ***That means that there is an 80% chance that they will not use the same terms.***

There are two ways to address this issue. One way is through the use of software that allows "fuzzy searches", though these packages tend to be expensive. Another is through the use of a lexicon or synonyms table/field. The first option is currently under investigation here, as we are reviewing the capabilities of some commercial Knowledge Base packages. The way we currently do it in our present implementation, is via option two.

The solution that we have used here, due to the limitations of our database package, is to create a field in our database called "Synonyms". Each Knowledge Base record has this field in it. Whenever a new record is created, the creator fills in this field with alternate names, alternate spellings, and likely misspellings. And, furthermore, this field is updated regularly as appropriate. It is important that when the Gatekeeper sees a new record created, they go in to this field and add synonyms to ensure the searchability of the database.

I cannot emphasize the importance of this functionality strongly enough. Think of your own organization, how many acronyms fly around on a regular basis, and how many different terms are used to describe the same thing. It is truly a jungle out there...

In the situation mentioned at the beginning of the section, I immediately pulled up the "Customer Accounts" record and added "project control", along with some likely variations (such as "project controll", and "control project"), a synonym. So next time that customer calls, even if they get a rookie who has only been manning the phones for an hour, they will end up in the right place.

IV. Only nothing is too little

Remember, if you know nothing about Product X, then anything you find in a Knowledge Base is more than you know.

The bottom line here is don't hold back due to insufficient information. You can always add more later. If you have anything at all, get it in there. The more your Knowledge Base users find things, even if it's minimal, the more they will use it. And, if they find a little, that may be enough to either answer a question or point them in the right direction. Then, hopefully they will come back later and supply you with what they couldn't find before.

Nothing is more discouraging to a KB user than coming up totally dry.

V. Tracking vs. Resolving

In a fast-paced environment like most help desks, it is difficult for your technical consultant to find time to enter detailed responses about what they did to resolve a customer's problem. The pressures of taking care of one customer so that they can hurry up and get on to the next one tends to create a culture in which many problem reports are closed with entries like "Fixed" or "Done" or "Assisted customer in resolving problem."

Thus, it is my experience that the problem tracking database is of limited use in resolving problems in our environment.

That is not to say that it can not be done. I know of one organization which has created a fine Knowledge Database using entries put in to a problem database by consultants. However, in that organization, the "Problem Tracking" system is divorced from the "Call Recording" system much in the way that our Knowledge Base is divorced from our Service Ticket. And, in that organization there is a strong culture and a strong management directive regarding the use of the problem tracking system.

An alternative to this approach, though it becomes a little boring at times, is to have the Gatekeeper regularly review the Service Tickets from the previous day. This can be extremely helpful, though endlessly tedious.

So the question naturally arises, how can you get consultants involved in maintaining a Knowledge Database or how can you use a tracking system as a KB?

If you intend to use a tracking system as a Knowledge Base, you must have a culture that can support a management mandate that the consultants enter a technically valid and concisely written response to customer calls. If you can mandate this and if your consultants have time for this, then you are well on your way.

On the other hand, if either the culture won't support it or your consultants just don't have time, then read on.

VI. Setting the Example

Before you let the consultants enter or update information in the Knowledge Base, you must first let them know what is expected. And I don't mean by mandating it, I mean by example.

In our first, failed, attempt at setting up our KB, discussed earlier in this paper, as soon as the Knowledge Base was there, everyone had the opportunity to add and update. As I mentioned, that failed very quickly.

In our second, more successful and still running KB, I feel that we made an error in the opposite direction. The Knowledge Base had been in place for nearly nine months before we allowed anyone except the Gatekeeper to update it. Today, about three months after we opened it up, updates by anyone but the Gatekeeper are still not the norm. On average, about 5% to 10% of all updates to the KB are done by consultants or technical experts (and the updates are, almost across the board, excellent). About 90% to 95% are still done by the Gatekeeper. I feel this is because people have just become too comfortable with the idea of not doing it themselves but rather leaving it to the "expert" who would do it right. And there is probably a certain amount of fear involved as well; folks are scared of "messing it up".

The Knowledge Base, once established, should run "untouched" for a period of three to four months without anyone but the Gatekeeper entering data into it. This will allow the users to get a feel for how entries in the database should look and taste. Then, after a few months, open it up.

Understand that there will be some failures. The Gatekeeper will spend a lot of time cleaning up after the users, indexing records, adding synonyms, and whatnot. But, over time, as the folks inputting the data start to get a feel for it, things should improve.

An interesting trend that I have noticed in this area involves the newer members of our help desk team. Most of the folks who have been here a long time tend to be more hesitant about updating the KB, though that is slowly changing. They are the ones who are more likely to send an email message to the Gatekeeper (me) with the update rather than going in and updating it themselves. On the other hand, many of the newer consultants, who "grew up" using the KB and are more comfortable, are much less nervous and more willing to go in make updates and corrections.

Admittedly this is a double-edged sword. The updates made by the newer folks are more likely to need to be "retouched" than those made by the more experienced people. But, on the other hand, their updates also frequently challenge some of the assumptions that we are making about the nature and the structure of the data, thereby stretching the bounds and limits into new areas that we hadn't considered.

To me, at least, the benefits are outweighing the dangers. But just remember, when you open up the Knowledge Base so anyone can update it, you are going to have to watch those updates very carefully.

It is *because* there are likely to be mistakes, and *because* the Gatekeeper will need to occasionally go in and clean up errors made by users, that an Audit Trail, the subject of the next section, is a must.

VII. Big Brother is Watching

In our Knowledge Base, any time anyone goes in and updates the information in an entry, a record of that update is kept in a separate table. And not just a rollback or recovery database, but in a table viewable by the Gatekeeper that shows who made the update, what record they updated, what field they updated, and the old and revised information in that field.

This has been invaluable to us for several reasons.

First, on those occasions when someone has made a mistake, you can go back and recover the data.

Second, because, as the Gatekeeper, you want to monitor everything that is going on in the database.

Third, because it allows you to see who is updating and to reward the participators and urge the non-participators to update the data themselves.

Finally, it just allows you to see how many updates are being performed on a regular basis.

We have found this to be an essential function in the KB.

Submitter	Action-date	Audit Id		
Geoff	January 20, 1997 1:13:12	KA0000000002299	Display this Card	Audit this Card
Card Number	Card Header			
KB0000112	SUDAAN, SESUDAAN			
Modified Field				
Card Header				
Original Data		Modified Data		
SUDAAN		SUDAAN, SESUDAAN		

Number 1 of 1

Figure 8: Audit Trail record showing an update to the database.

VIII. Document your processes!

Technical documentation for the systems you develop.

Need I say more?

Active Initiatives

Up until now, I have pretty much addressed what we have already done. Here are a few of the initiatives that are on the table currently.

Levelling or Tiering of information: One of the directions we would like to steer towards, as already mentioned, is an Answer Base function. Though no final decision has been made on implementation details, right now we are leaning towards a tiered approach. By this, I mean all the data resides in one database, except there are different levels of record. If a user queries on the search string "macintosh" they would get the main or "first level" record that gives the basic information on the product. Then, they would have the ability to pull up the "second level" records on this subject. These would be the questions and answers, the how-tos, etc. My thinking in this area is not completely clear yet, and it is still on the drawing board, but after several months of thought and experimentation, I think that this will be the approach I take.

AI Software: We have been performing reviews of some Artificial Intelligence software and trying it out in our help desk environment. So far, most of our consultants have remained happy with the KB as it is implemented in Remedy, and they have given thumbs down to other software we have brought in to test.

Improved Cross Referencing: Currently in the KB, many records contain information that cross-references to other records in the database. At this time we have not implemented an acceptable method for managing this, though several possibilities are under investigation. At one point we had implemented a "Related Records" function based on keywords, but as the KB grew, the usefulness of this functionality declined.

This just in...

As I have stated several times already, this paper is not meant to be the last word on the subject, but rather merely a statement of where we are in an evolving process. Therefore, it was necessary to drive a stake into the ground at a specific point in time and describe the project as it was at that moment. Since the process of writing a paper such as this takes some time, the Knowledge Base has evolved some since I started writing. Here is a quick look at what enhancements we have brought forth in the interim.

Most importantly, we have modified the Administrator's database so as to record not just queries and hits, but all activities and functions that users perform against the Knowledge Database. We now capture such basic information as who prints records, who emails records, and who closes Service Tickets to Knowledge Base entries.

Partly, the philosophy behind the admin schema has changed a little. Now we look at this schema as *recording any action against the KB database which does not alter auditable data*. And, of course, all changes to auditable information are recorded in the Audit Trail.

The other change we are making now, and this is admittedly minor, is a change to what we search when a user queries the KB. Soon, a query will not just search the three fields outlined earlier, but will also search contact information for matches. This is a request from several members of our help desk staff.

Closing

Knowledge Databases are a technology whose time is rapidly approaching. What I have attempted to present here is a picture of an evolving project at a point in time. We have a lot of work left ahead of us if we are to fully realize the potentials of this technology.

However, I hope that this outline of what we have accomplished so far may help others in achieving what we have achieved in less time that it took us. Additionally, I hope that it fosters debate and discussion in the technology community. In presenting this information, I am asking the readers to share their experiences as well in the hope that we can all learn from this process.

Trademark Acknowledgements

Remedy, Action Request System, and AR System are trademarks of Remedy Corporation. MacIntosh is a registered trademark of Apple Computer, Inc. IBM is a registered trademark of International Business Machines Corporation. Windows is a trademark of Microsoft Corporation. All other trademarks, service marks, etc. are hereby acknowledged and are property of their respective owners.

General Acknowledgements

Tara Riley, for creating the CardFile which launched the project.

Lee Freeman, for his assistance and patience while I learned Remedy Administration.

Knowledge Base White Paper

Dale Spangenberg, for his support of the project and for his review of the draft.

Star Kline, Jim Barrett, Joe Gannon, Jill Kellison, Mike Wright, Scott Collins, Rick Duhn, Steve Noe and Lee Freeman for reviewing the draft of this paper and supplying many insightful comments.

All CSB consultants who have contributed their thoughts and ideas, as well as their queries.