

RMOUG SQL>UPDATE

The Magazine for the Rocky Mountain Oracle Users Group • Vol 61 • Winter 2010

Steven Feuerenstein
Guarantee Application Success

John Krahulec
Enterprise Social Networking

Tim Mishek
ASM - The Next Generation

Benjamin Wood
Four Things To Know About MySQL

Board Focus - Heidi Kuhn
Member Focus - Dan Hotka

Change Service Requested

Rocky Mountain Oracle Users Group
PO Box 621942
Littleton, CO 80162

Non-Profit
Organization
U.S. Postage Paid
San Dimas, CA
Permit No. 410

Enterprise Social Networking

It's Now Ready For The Workplace

by John Krahulec

The term Social Networking makes people think about sites like Facebook and the sometimes-inappropriate information that people post on it. Rarely does anyone think about how this new medium has enabled people to communicate in new ways. Enterprise Social Networking takes advantage of these communication methods to allow employees to share, collaborate, and stay in touch about business related items. Enterprise Social Networking connects People to People and People to Information. Social Networking connects People to People.

Enterprise Social Networking has been a holy grail for many years. It had many names: Enterprise Social Media, Enterprise 2.0, Gov 2.0, Enterprise Collaboration, and so on. Regardless of the name, the objective has always been to improve communication and collaboration within an organization. This concept has been fueled by the success of Social Networking sites like Facebook and Twitter. Many organizations have tried to embrace these existing sites for their own Enterprise use, but have quickly found that the nature of these open (and very public) sites does not meet the Enterprise requirements of information security, policy enforcement, and application integration, while maintaining meaningful collaboration among other things.

In this article, I will cover the merits of Social Networking, the merits of applying it in the Enterprise, and how to do it all from an Oracle database.

The Merits of Social Networking

Social apps aim to provide value in three ways. First, they can bridge geographical and organizational information divisions by moving conversations out of email and hallways and into shared online spaces such as walls, forums, blogs and wikis. This way, information becomes searchable, serendipitous connections can be made, and ideas pollinate in ways they couldn't before. Communities of interest spring up around subject matter rather than organizational hierarchies.

The second way social apps provide business value is by letting people add context to information stores, which helps others identify what's useful to them and can make search results more relevant. Social bookmarks, for example, let people share links to a content source – a Web page, blog, or white paper – that they found useful. Tags, social bookmarks, and other social networking tools help bring order to the avalanche of information that employees have to sift through.

Third, Enterprise Social Networking helps people find and connect to co-workers through user profiles, subject matter forums,

and social graphs that provide a visual map of an employee's connections with co-workers. No longer are employees just a cog in the machine. They are people again.

Now, let's do a quick review of the current social media elements that make up Social Networking:

- IM/Chat – AIM, Google Chat, Yahoo Messenger
- Blogs - WordPress, Drupal
- Microblogs - Twitter
- Collaborative Encyclopedia - Wikipedia
- Social Networks – Facebook, LinkedIn



Each has a specific purpose and offers different types of communication. You could certainly use all of them in the office, but that doesn't make it Enterprise. Each one has its own security authentication (no single sign-on) managed by some other unknown IT organization. True, there is some level of integration (e.g. mash-up) capability such as posting Tweets to your Facebook page or displaying your Facebook page on your blog site, but that isn't the only level of integration we want to achieve within the Enterprise. Also, these choices don't necessarily allow you to restrict information to specific groups or individuals. That would be a direct contradiction to the intent of current Social Networking: to make and share information with the world!

Now let's step back a moment. We like all the collaborative features afforded by social media, but it just won't work in an environment where marketing forecast reports need to be shared with the Marketing & Sales teams (and hidden from the Finance

department or the staff in the mail room). Also, you want these features integrated and within context to your corporate applications. Oh, and you want your information to be searchable, but yet again, only to those with the appropriate permission. And just to make it all easier to manage, let's keep it all on one unified platform... one that presumes, that as a reader of this publication, you are a user of the Oracle database. (That's not necessarily an overarching requirement of Enterprise Social Networking, but as you'll see, it makes our work much easier.) So what options are available to the Enterprise? The following table gives you a glimpse of the differences between consumer social networking sites and the type of social networking the Enterprise is looking for:

| Type of Social Networking Site | Need | | | |
|--------------------------------|--|--|---|---|
| Consumer Social Networking | Security User authentication at login, but security breaches commonly found | Integration No integration with other apps | Group Access Control Minor access control for groups Example: Facebook allows these choices: Everyone Friends Only Friends of Friends | Control of Features No messaging control Example: Everyone can view any message you post |
| Enterprise Social Networking | Multi-layered security: database level, application level, and use of access control lists Enforce security policies, regulatory compliance | Integrate with existing enterprise apps for real-time data updates across your suite of applications | Granular access control based on defined groups | Messaging control can be enforced based on groups |

Bringing the Merits of Social Networking to the Enterprise

Why do we need an Enterprise social network?

Some of the merits of social networking are obvious, in an Enterprise context: Use Status Updates to let others know that you'll be in a meeting down the hall. Search the online employee directory to find people with common skill sets in other locations of your company. Use embedded Chat windows to discuss marketing data in real-time, and be able to search and retrieve that information later. Dynamically create social groups based on project teams, functional areas, and/or common interests, and have it include features such as group calendars, file sharing, and news boards. Use 'Search' features to access information according to access control lists. The ways you can apply social networking to the Enterprise seem limitless.

We are starting to see some new Enterprise Social Networking products come to market (e.g. Salesforce Chatter) or old products being retooled (or just re-marketed) to have a social networking flair (e.g. SharePoint). As promising as these seem, each has its own inherent problems. Salesforce Chatter works great for those employees in the Sales organization that already have access to Salesforce, but it requires everyone else in the company to now login to yet another site to participate in the "chatter". But, if there is no compelling reason for them to log in, then only a small portion of

your company is participating. This hardly makes it an Enterprise Social Network.

As far as SharePoint is concerned, it started as a document management solution, then was pushed beyond its original intent when developers tried using it to create applications, and then Microsoft created a mish-mash by bolting on some wiki and blog features so they could jump on the Enterprise Social Networking bandwagon. SharePoint is trying to be everything without being able to do anything well. I don't think the mish-mash can grow.

Road Blocks to Enterprise Adoption

So, it's clear that while Enterprise Social Networking has

merits, there is a lack of understanding of the Enterprise requirements in the current line of product offerings. In addition, you will find some other roadblocks to Enterprise adoption of social networking.

Perceived Tomfoolery: First and foremost, there's the bad rap these apps get from consumer-focused cousins such as Facebook and Twitter, which are perceived as outlets for sophomoric self-aggrandizement. Not much more to say here.

ROI: What's the ROI? Seems like a standard question for every new IT initiative. However, though email has been around for quite a while, I would have loved to hear the ROI justification for it. It's probably hard to explain, but we know we need it. Just as with e-mail and Internet access 15 years ago, it is difficult to quantify social media's utility. CIOs know that social networking will find its way into their companies if IT doesn't provide it. Beyond Facebook and Twitter are scores of wiki, blog, and collaboration sites that are free, or cheap enough that a product manager can buy a half-dozen accounts with a credit card and upload next quarter's product road map in five minutes. Therefore, even if the ROI is never clear cut, IT managers may be compelled to invest in these apps to keep employees from adopting public systems that are outside company control.

There are some other metrics that can serve to answer part of the ROI question. One metric is user adoption. A large company can have about 10,000 wiki documents and 3,000 blogs across 150 topical communities on its Enterprise Social Network. Of its 10,000 global employees, the platform may have about 1,000 active users and another 1,500 lurkers who are watching conversations or consuming content but haven't posted anything themselves. Adoption doesn't necessarily tie in to ROI; however, you should be able to point to a decrease in the volume of emails and attachments across the company's network, and the resulting reduction in storage costs. Instead of a 20-MByte PowerPoint being sent back and forth among

10 co-workers, it can sit in a common workspace where people can post comments about it.

Another way to measure the value of social networking tools is to relate them to process improvements rather than time savings or loose efficiency gains. Some process improvement areas are consolidation/integration, policy/content control, policy questions around content control, compliance, and moderating employee behavior.

Why Not Just Bring a Facebook Platform Into the Enterprise?

This has been done. Jive Software. Lotus Connections by IBM. Eureka Streams (www.eurekastreams.org) by Lockheed Martin. However, the problem is that it is still another website to log into. It is not integrated with the other corporate applications used on a daily basis. It loses Enterprise context if the social network is not integrated with the applications.

The Easy Answer

The easy answer is to use a platform that allows for custom application development, and has built in features for Enterprise Social Networking, document management and collaboration, a calendar that supports individual and group events, etc., to create a truly Rich Web Application.

Applying Social Networking to the Enterprise with the Oracle Database

So, you're convinced of the merits of social networking, and you want to apply it to your Enterprise. You have seen consumer social networking sites fail to meet Enterprise demands. However, you have your Enterprise Social Networking strategy planned: you know what types of dynamic groups you'd want within your Enterprise, what security policies you'll need to adhere to, the regulations, if any, your Enterprise must follow, and so on. Now, you want to see what the best, most efficient way is to roll out your strategy.

In other words, the question is: How can you reap the benefits of social networking, meet stringent Enterprise requirements, and leverage your existing infrastructure? Let's consider these basic factors to evolving your strategy:

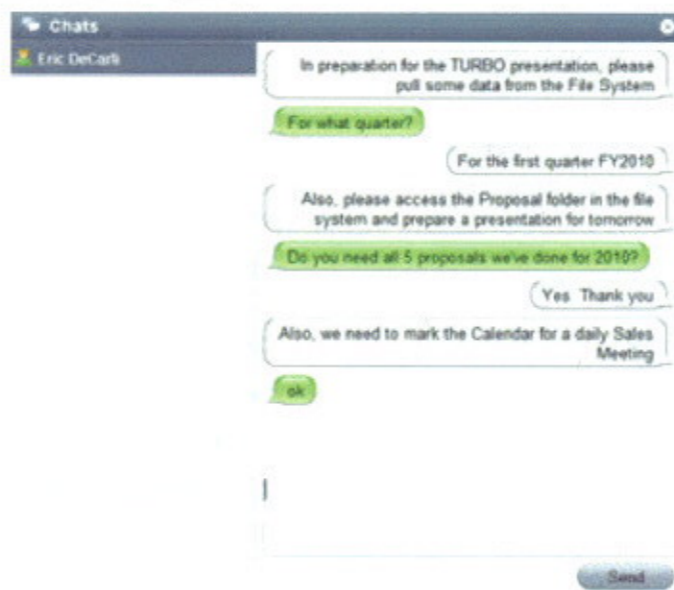
- How will you reconcile existing workplace policies with the new virtual environment?
- How will you maintain your Enterprise Social Networking tool of choice in a separate database?
- What resources will you allocate to maintain data, independent databases, and software?
- How will you monitor the stringent security policies your Enterprise demands?
- How will you maintain control of information while encouraging communication?

Rather than look around for expensive, stand-alone Commercial Off The Shelf (COTS) products, you can leverage your existing Oracle database to bring together all the great elements of Social Networking to improve how your Enterprise works. Just like Enterprise Social Networking is a new way to streamline and leverage the same vast pools of corporate information, using your Oracle database to develop your Enterprise Social Networking tool is a new way to leverage the same vast potential your Oracle database offers.

Let's take it one step further. What could be the easiest way to leverage your existing Oracle infrastructure, resources, training, man-power, and maintenance, without compromising on cutting-edge web technologies like HTML 5, JSON, AJAX, and FLASH that makes those consumer social networking sites so jazzy?

By using PL/SQL. With PL/SQL, you can use an unlimited number of Application Programming Interfaces (APIs) that will work with your application and maximize PL/SQL's top performance for database-intensive applications. For example, integrating the instant messaging feature allows you to leverage all your database features as opposed to using a stand-alone product that will require building APIs to retrieve 'chat' logs from your Oracle database.

Let's look at how you can build an instant messaging feature integrated with your Oracle database as part of your social Enterprise networking strategy, eliminate the traditional 'middle-tier', secure your data with multi-level security, pare down development efforts, and provide the same rich user experience of consumer social networking sites:



Set up tables within your database to store chats, chat archives, buddies, chat sessions, and chat user information.

| COLUMN_NAME | DATA_TYPE | NULLABLE | DATA_DEFAULT | COLUMN_ID | COMMENTS |
|--------------|----------------------|----------|--------------|-----------|----------|
| MESSAGE_ID | NUMBER | No | (null) | 1 (null) | |
| TX_USER_ID | NUMBER | Yes | (null) | 2 (null) | |
| RX_USER_ID | NUMBER | Yes | (null) | 3 (null) | |
| TX_TIMESTAMP | TIMESTAMP (6) | Yes | (null) | 4 (null) | |
| MESSAGE | VARCHAR2 (4000 BYTE) | Yes | (null) | 5 (null) | |

Use JSON to manage data to your chat's client components: the Buddy List and the Chat window input fields used for sending messages. For example, clicking on the 'Submit' button to send a message activates AJAX which transmits the message and the name of the recipient directly to an Oracle stored procedure.

Use PL/SQL to transmit/receive the message and recipient variables, and insert the data into your database tables.

To fetch data, use a Polling System to make AJAX requests to the database. The Polling System invokes a procedure that checks the status of a user's buddies, for example, or looks for messages to be received. The JSON code block is sent to the client.

The following sample code shows how to write in-bound messages to the database:

```
PROCEDURE p_chat_write
(i_rx          IN VARCHAR2,
i_message     IN turbo_chat.message%TYPE)
IS
BEGIN
INSERT INTO turbo_chat
(message_id, tx_user_id, rx_user_id, tx_timestamp, message)
VALUES
(seq_chat_message.NEXTVAL, te_security.g_security_rec.user_id,
te_crypto.f_decrypt(i_rx), CURRENT_TIMESTAMP, i_message);

COMMIT;

EXCEPTION
WHEN others THEN
te_logging.p_error(SQLCODE, SQLERRM,
te_logging.f_get_procedure, NULL);

END p_chat_write;
```

Write chat message to the chat inbound message table

Construct JSON object to send back to client with all messages to display

Delete message from inbound message table so it is not sent again

The following sample code shows how you can update the JSON object with messages to send to the client:

```
PROCEDURE p_chat_read
(i_json       IN INTEGER)
IS
CURSOR messages IS
SELECT message_id, te_utils.f_get_full_name(tx_user_id) tx_name,
tx_user_id,
TO_CHAR(tx_timestamp, 'MM/DD/YYYY HH24:MI:SS') tx_time,
message
FROM turbo_chat

WHERE rx_user_id = te_security.g_security_rec.user_id
ORDER BY message_id;
BEGIN
MERGE INTO turbo_chat_users a
USING (
SELECT te_security.g_security_rec.user_id user_id
FROM dual) t
ON (a.user_id = t.user_id)
WHEN MATCHED THEN
UPDATE
SET
last_ping = SYSDATE
WHEN NOT MATCHED THEN
INSERT
(user_id, last_ping)
VALUES
(te_security.g_security_rec.user_id, SYSDATE);

te_json.p_object_open(i_json, 'chat');
te_json.p_array_open(i_json, 'messages');
```

Select messages from inbound message table to send to the user

Update the chat 'ping' table for online awareness

```
FOR x IN messages LOOP
te_json.p_object_open(i_json);
te_json.p_add_name_value(i_json, 'tx', te_crypto.f_encrypt(x.tx_user_id));
te_json.p_add_name_value(i_json, 'nm', x.tx_name);
te_json.p_add_name_value(i_json, 'text', x.message, FALSE);
te_json.p_object_close(i_json);

DELETE <your_app_name>.chat
WHERE message_id = x.message_id;

END LOOP;

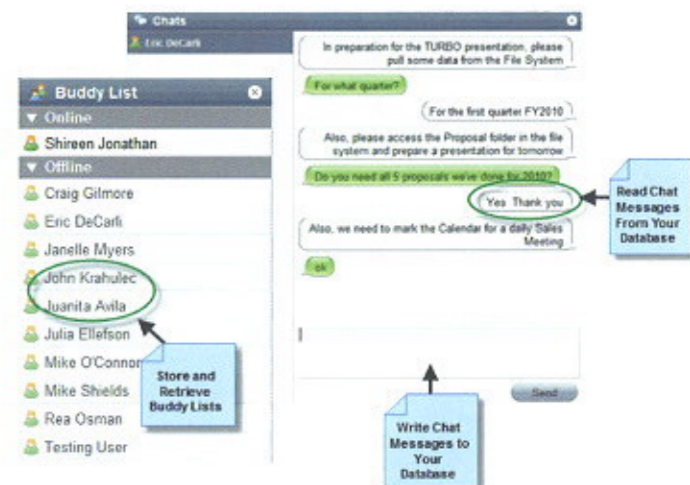
te_json.p_trim_comma(i_json);
te_json.p_array_close(i_json);
p_buddy_list_update(i_json);
te_json.p_trim_comma(i_json);
te_json.p_object_close(i_json);

COMMIT;

EXCEPTION
WHEN others THEN
NULL;

END p_chat_read;
```

Here is a sample chat window and buddy list that effectively utilizes the database:



You can similarly add additional Enterprise Social Networking capabilities such as an integrated calendar, a personal wall, and a file sharing and management system, to provide seamless information flow. Let's take a quick look at how you can build a wall.

Set up tables within your database to store your wall posts, post images, make entries, and post comments for a personalized wall. The following is a sample table for blog entries:

| COLUMN_NAME | DATA_TYPE | NULLABLE | DATA_DEFAULT | COLUMN_ID | COMMENTS |
|----------------|----------------------|----------|--------------|-----------|----------|
| BLOG_ID | NUMBER | Yes | (null) | 1 (null) | |
| ENTRY_ID | NUMBER | No | (null) | 2 (null) | |
| BLOG_TEXT | VARCHAR2 (4000 BYTE) | Yes | (null) | 3 (null) | |
| LOAD_USER_ID | NUMBER | Yes | (null) | 4 (null) | |
| LOAD_DT | DATE | Yes | (null) | 5 (null) | |
| UPDATE_USER_ID | NUMBER | Yes | (null) | 6 (null) | |
| UPDATE_DT | DATE | Yes | (null) | 7 (null) | |

Each time an entry is posted on the wall, the client-side sends it to your database using Javascript to save the data. You store the data in a table using PL/SQL. The database then creates and sends back a JSON object, whose contents are rendered by the client for display on the wall. The following sample code shows how you can write and read data:

```

Write/store new post:
FUNCTION f_post_entry_merge
(i_rowtype IN te_types.rt_turbo_blog_entries)
RETURN turbo_blog_entries.entry_id%TYPE
IS
l_rowtype te_types.rt_turbo_blog_entries := i_rowtype;
BEGIN
-- Create entry_id new posts
IF l_rowtype.entry_id IS NULL THEN
-- Get new blog id
SELECT seq_turbo_blog_entries.NEXTVAL
INTO l_rowtype.entry_id
FROM dual;
END IF;
MERGE INTO turbo_blog_entries a
USING (
SELECT l_rowtype.blog_id blog_id,
l_rowtype.entry_id entry_id
FROM dual) t
ON (a.blog_id = t.blog_id
AND a.entry_id = t.entry_id)
WHEN MATCHED THEN
UPDATE
SET
blog_text = l_rowtype.blog_text,
update_user_id = te_security.g_security_rec.user_id,
update_dt = SYSDATE
WHEN NOT MATCHED THEN
INSERT
(blog_id, entry_id, blog_text, load_user_id, load_dt)
VALUES
(l_rowtype.blog_id, l_rowtype.entry_id, l_rowtype.blog_text,
te_security.g_security_rec.user_id, SYSDATE);
RETURN l_rowtype.entry_id;
EXCEPTION
WHEN OTHERS THEN
te_logging.p_error(SQLCODE, SQLERRM, te_logging.f_get_
procedure, NULL);
RETURN NULL;
END f_blog_entry_merge;

```

Generate post ID if this is a new wall post

Store/update wall post in database table

Update wall post if record exists

Insert new wall post if record does not exist

```

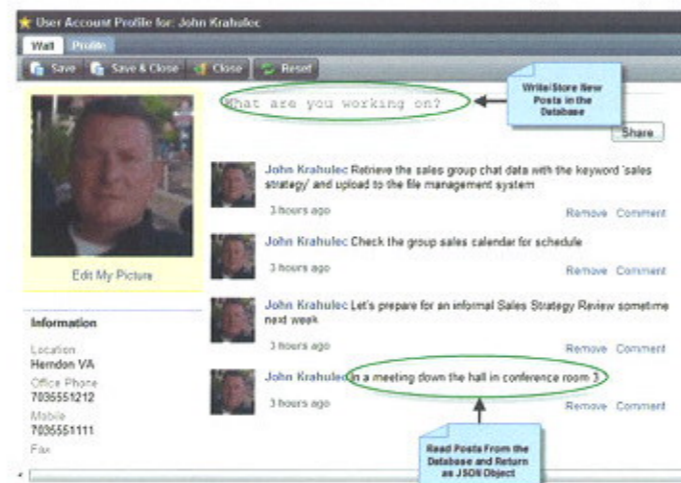
BEGIN
p_json_buffer('');
FOR x IN blogs LOOP
p_json_buffer('');
p_json_buffer('id:' || te_crypto.f_encrypt(x.blog_id) || ',';);
p_json_buffer('entry_id:' || te_crypto.f_encrypt(x.entry_id) || ',';);
p_json_buffer('author:' || te_utils.f_get_full_name(x.load_user_id) || ',';);
p_json_buffer('date:' || f_date_format(x.load_dt) || ',';);
p_json_buffer('text:' || REPLACE(REPLACE(x.blog_text, '\', '\\'), Chr(10), '<br>') || ',';);
p_json_buffer('comments':{});
p_blog_comment_json(x.blog_id, x.entry_id);
p_json_buffer('');
p_json_buffer('');
END LOOP;
g_json_arr(g_json_idx) := RTRIM(g_json_arr(g_json_idx), ',');
p_json_buffer('');
owa_util.mime_header('text/javascript', FALSE, 'ISO-8859-1');
htp.p('Cache-Control: no-cache');
htp.p('Pragma: no-cache');
owa_util.http_header_close;
FOR i IN 1..g_json_arr.COUNT LOOP
htp.prn(g_json_arr(i));
END LOOP;
EXCEPTION
WHEN OTHERS THEN
te_logging.p_error(SQLCODE, SQLERRM, te_logging.f_get_
procedure, NULL);
END p_blog_json;

```

Build wall into JSON object

Pass JSON to the browser

Here is a sample personal Wall:



Read post/wall data and return as JSON object to be rendered by the client:

```

PROCEDURE p_blog_json
(i_blog_id IN VARCHAR2)
IS
CURSOR blogs IS
SELECT *
FROM turbo_blog_entries
WHERE blog_id = te_crypto.f_decrypt(i_blog_id)
ORDER BY load_dt desc;

```

Get user's entire wall from table

Bringing It All Together

Social networking is coming to the Enterprise as surely as email did in the '80s and '90s. However, the challenges and roadblocks to Enterprise adoption only seem to have fueled a mish-mash



of products that all claim to be the 'holy grail' that will make social networking work for the Enterprise. The result is an avalanche of half-baked products that have been thrown together hoping to fill the very present need for a robust Enterprise Social Networking solution, without addressing the requirements of the Enterprise: information security, policy enforcement, and application integration, while connecting people to people and people to information. Instead of installing yet another overly complex, burdensome set of applications on already overloaded employees, try an Enterprise Social Networking solution that is already connected to your Enterprise data. Use your PL/SQL resources and your already stable Oracle-based infrastructure to get started now.

John Krahulec is Executive Vice President and COO for TURBO Enterprise, LLC and also for ConceptSolutions, LLC, a Deloitte Technology Fast 500 IT and management consulting firm. John has led development of TURBO Enterprise, a Rich Internet framework for PL/SQL developers, that delivers Enterprise-class applications from the Oracle Database. He has spent the past several years focused on fusing Web 2.0 technologies (HTML5, AJAX, FLASH, JSON, CSS, XML, etc.) and Enterprise-class social networking features with Enterprise back-end data to produce Rich Web Applications (RWA) in corporate, government, and military environments. With over nineteen years of Enterprise IT experience, John brings a unique perspective to RWA development, having developed databases for both desktop and Web applications.



The Rich Internet Framework for PL/SQL

**PL/SQL SDK, Framework & Application Platform
HTML5, AJAX, JSON, CSS3, FLASH & JavaScript**

Integrated Enterprise Social Networking

11600 Sunrise Valley Dr, Suite 300
Reston, VA 20191

For Sales or Information: 703-889-8499
info@TURBO-Enterprise.com