ThoughtWorthy Media, Inc. Sample Database Documentation v1.4

Introduction:

Welcome to the ThoughtWorthy Media, Inc. sample database documentation. The goal of this document is to introduce a web and database programmer to retrieving data from TWM's unique and powerful database through a design overview and example queries. If you are looking for instructions to enter data, please see the ThoughtWorthy Media Production Assistant iAdmin Manual.

This document contains confidential and proprietary materials.

THOUGHTWORTHY MEDIA, INC.
3927 OLD LEE HIGHWAY, SUITE 102-C
FAIRFAX, VIRGINIA 22030-2422
(T) 703.359.8450 (F) 703.359.8451

HTTP://WWW.THOUGHTWORTHY.COM/

Table of Contents:

Descriptors and Sub-Descriptors:

Tables:
`| descriptor_types`

Relational databases traditionally suffer from a strict rigidity in design. Because of this inherent limitation, a pure relational design is not appropriate for databases, such as that of ThoughtWorthy Media, which must add features constantly and share its data with partners.

Instead, TWM designed a unique “descriptor” and “sub-descriptor” database hierarchy. This database hierarchy lets us have main tables with an unlimited hierarchy of descriptors with descriptors to those descriptors and so on. This allows our design to be completely flexible and allows for a minimal amount of database changes that require partner implementation changes.

And, because we strive to make our descriptor titles completely human readable, there is less confusion for partners to implement our data, *especially for new features!*

While discussing new features, it is important to note that to increase the speed of queries joining on descriptors containing IDs, the descriptor databases will be implementing a new ID descriptor column in addition to the existing text descriptor column in 4Q 2007.

When using the descriptor tables, there are several different descriptor types as defined in the descriptor_type table. These descriptor types help determine what you do with each descriptor as described below in Table 1:

Descriptor Types	Typical Uses
Image	An image descriptor is a graphic that can be displayed.
Text	A text descriptor is plain-text that can be displayed.
contact_id	A contact_id descriptor indicates that a contact name and link should be displayed. For example, the Spouse descriptor may return a Text descriptor or a contact_id descriptor.
image_no_border	An image_no_border descriptor should be displayed without a visible border.
contact_id_w_logo	A contact_id_w_logo descriptor indicates that a contact logo should be displayed and linked to the contact. For example, a Record Label descriptor may return a contact_id_w_logo allowing for the display of the company logo rather than simply text.
wiki_text	A wiki_text descriptor indicates that the descriptor should be parsed prior to display. The parsing should be done either using MediaWiki (http://www.mediawiki.org/) or TWMWiki (http://www.thoughtworthy.com/sandbox.cgim).
thought_id	A thought_id descriptor indicates that a thought category name should be displayed and linked to the thought category.
general_id	A general_id descriptor indicates that a general category name should be displayed and linked to the general category.
product_id	A product_id descriptor indicates that a product name should be displayed and linked to the product.
html	An html descriptor indicates that the descriptor is formatted as HTML and can be displayed.
general_descriptor_image	A general_descriptor_image descriptor indicates that a general category logo should be displayed and linked to the general category.
general_descriptor_image_no_border	A general_descriptor_image_no_border descriptor indicates that a general category logo should be displayed without a visible border and linked to the general category.
thought_descriptor_image	A thought_descriptor_image descriptor indicates that a thought category logo should be displayed and linked to the thought category.
thought_descriptor_image_no_border	A thought_descriptor_image_no_border descriptor indicates that a thought category logo should be displayed without a visible border and linked to the thought category.
product_descriptor_image	A product_descriptor_image descriptor indicates that a product image should be displayed and linked to the product.
product_descriptor_image_no_border	A product_descriptor_image_no_border descriptor indicates that a product image should be displayed without a visible border and linked to the product.
contact_descriptor_image	A contact_descriptor_image descriptor indicates that a contact image should be displayed and linked to the contact.
contact_descriptor_image_no_border	A contact_descriptor_image_no_border descriptor indicates that a contact image should be displayed without a visible border and linked to the contact.

Table 1 - Descriptor Types and Typical Uses

Additionally, when using any of the descriptor tables, there are three basic rules to keep in mind:

1. Sub-descriptors are still descriptors. They use the same database design except that instead of linking to a main table row, they link to a descriptor table row.
2. There are Primary and Secondary descriptors. Primary descriptors have a Priority less than 1000 and Secondary descriptors have a priority of 1000 and above.
3. Hidden descriptors are used to control the look and feel of the output but are not designed to be directly viewed. For example, a gender descriptor of 'Male' could be used to output Actor instead of Actress but would not be appropriate to display as-is.

Main Tables:

The main tables are: Contacts, Products, Global Categories, General Categories and Thought Categories.

Contacts:

Tables:
```
| contacts
| contact_types
| contact_descriptors
| contact_subdescriptors
| contact_sub_subdescriptors
```

The contacts table includes any type of contact from Actors/Actresses to Restaurants to Websites. It has 3 important columns: id, name, and alphabetize_on.

To pull contact ID #22 from the database, the following query could be used:

```
SELECT id as contact_id, name, alphabetize_on
FROM contacts
WHERE id=22;
```

And the results of this query on the sample database:

contact_id	name	alphabetize_on
22	Greg J. Grande	GRA

To find out the type of contact, query the contact_type table:

```
SELECT contact_type
FROM contact_types
WHERE contact_id=22
ORDER BY priority;
```

This will return:

contact_type
Set Decorator

Important: Contacts can have multiple contact types! For example, Bill Murray (Contact ID: 14843), will return multiple rows including: Actor, Comic, Producer & Director.

Querying the contact_descriptors for the chosen contact_id will then provide you with the descriptors:

```
SELECT contact_descriptors.id, contact_descriptors.title, contact_descriptors.descriptor,
contact_descriptors.descriptor_type_id, contact_descriptors.priority,
contact_subdescriptors.id as has_subdescriptor
FROM contact_descriptors
LEFT JOIN contact_subdescriptors ON
contact_subdescriptors.contact_descriptor_id=contact_descriptors.id
WHERE contact_descriptors.contact_id = 14843
GROUP by contact_descriptors.id
ORDER by priority, title;
```

This will return the following descriptors:

id	title	descriptor	descriptor_type_id	priority	has_subdescriptor
41117	Disable tPods	Enabled	2 [->text]	0	*NULL*
41118	Birth Name	William James Murray	2 [->text]	100	*NULL*
41120	Date of Birth	1950-09-21	2 [->text]	200	*NULL*
41121	Place of Birth	Wilmette, Illinois (USA)	2 [->text]	300	*NULL*
41122	AKA	Billy	2 [->text]	350	*NULL*
41123	Spouse	Jennifer Butler	2 [->text]	400	*NULL*
41124	High School Alma Mater	Loyola Academy (Wilmette, Illinois, USA) - Graduated	2 [->text]	500	*NULL*
41125	University Alma Mater	Regis College (Denver, Colorado, USA) - Studies Deferred	2 [->text]	600	*NULL*
41126	Career Duration	1975-00-00 -	2 [->text]	700	*NULL*
41127	Biography	'''William James Murray''' (born [[September 21]], [[1950]]) is an [[Academy Awards|Academy Award]]-nominated [[United States|American]] [[comedian]], [[actor]], [[film producer|producer]], [[film director|director]] and [[poet]]. He is most famous for his comedic roles in "[[Groundhog Day (film)|Groundhog Day]]", "[[Caddyshack]]", "[[Ghostbusters]]", and "[[What About Bob?]]". He recently has gained prominence for more dramatic roles, such as in the 2003 film "[[Lost in Translation]]". **[NOTE: Descriptor truncated for demonstration purposes]**	13 [->wiki_text]	1100	4090
41119	Gender	Male	2 [->text]	10000	*NULL*

The example contact descriptor query above returns a has_descriptor column which can be used to trigger the pulling of sub-descriptors.

Digging further, you can retrieve the indicated sub-descriptors for the Biography descriptor for Bill Murray with the following query:

```
SELECT contact_subdescriptors.id, contact_subdescriptors.title,
contact_subdescriptors.descriptor, contact_subdescriptors.descriptor_type_id,
contact_subdescriptors.priority, contact_sub_subdescriptors.id as has_subdescriptor
FROM contact_subdescriptors
LEFT JOIN contact_sub_subdescriptors ON contact_sub_subdescriptors.contact_subdescriptor_id =
contact_subdescriptors.id
WHERE contact_subdescriptors.contact_descriptor_id = 41127;
```

This will return the following sub-descriptors:

id	title	descriptor	descriptor_type_id	priority	has_subdescriptor
4090	Wikipedia Link	http://en.wikipedia.org/wiki/Bill_Murray	2 `[->text]`	1100	*NULL*

To determine a complete and current list of all of the Contact Descriptors available, use DISTINCT SQL Queries, for example:

```
SELECT DISTINCT(title) FROM contact_descriptors;
SELECT DISTINCT(title) FROM contact_subdescriptors;
SELECT DISTINCT(title) FROM contact_sub_subdescriptors;
```

To determine a complete and current list of all Contact Sub-Descriptors and the Descriptors they describe, use the following query:

```
SELECT contact_subdescriptors.title as Sub_Descriptor_Title, contact_descriptors.title as
Descriptor_Title
FROM contact_subdescriptors
LEFT JOIN contact_descriptors on contact_subdescriptors.contact_descriptor_id =
contact_descriptors.id
GROUP BY contact_subdescriptors.title, contact_descriptors.title
ORDER BY contact_subdescriptors.title;
```

Finally, to determine a complete and current list of all Contact Sub-Sub-Descriptors and the Sub-Descriptors they describe, use the following query:

```
SELECT contact_sub_subdescriptors.title as Sub_Sub_Descriptor_Title,
contact_subdescriptors.title as Sub_Descriptor_Title
FROM contact_sub_subdescriptors
LEFT JOIN contact_subdescriptors on contact_sub_subdescriptors.contact_subdescriptor_id =
contact_subdescriptors.id
GROUP BY contact_sub_subdescriptors.title, contact_subdescriptors.title
ORDER BY contact_sub_subdescriptors.title;
```

Products:

Tables:
```
| products
| product_types
| product_subtypes
| product_descriptors
| product_subdescriptors
| product_sub_subdescriptors
```

The products table includes 3 important columns: id, name, and product_subtype_id.

To pull product ID #1224 from the database, the following query could be used:

```
SELECT id as product_id, name, product_subtype_id
FROM products
WHERE id=1224;
```

And the results of this query on the sample database:

product_id	name	product_subtype_id
1224	Rappa Ternt Sanga	247 [->Compact Discs (CDs)]

To find out the type of product, query the product_subtype & product_type table:

```
SELECT product_type, product_subtype
FROM product_types
JOIN product_subtypes ON product_subtypes.product_type_id = product_types.id
WHERE product_subtypes.id=247;
```

This will return:

product_type	product_subtype
Music	Compact Discs (CDs)

Querying the product_descriptors for the chosen product_id will then provide you with the descriptors:

```
SELECT product_descriptors.id, product_descriptors.title, product_descriptors.descriptor,
product_descriptors.descriptor_type_id, product_descriptors.priority,
product_subdescriptors.id as has_subdescriptor
FROM product_descriptors
LEFT JOIN product_subdescriptors ON
product_subdescriptors.product_descriptor_id=product_descriptors.id
WHERE product_descriptors.product_id = 1224
GROUP by product_descriptors.id
ORDER by priority, title;
```

This will return the following descriptors:

id	title	descriptor	descriptor_type_id	priority	has_subdescriptor
15950	Music Artist	12167	4 [->contact_id]	50	*NULL*
15948	Album Flag	Enhanced	2 [->text]	100	*NULL*
15949	Album Flag	Explicit Lyrics	2 [->text]	100	*NULL*
15951	Label	11841	4 [->contact_id]	300	*NULL*
15952	Label	29	4 [->contact_id]	300	*NULL*
15953	Music Genre	Rap / Hip-Hop	2 [->text]	400	*NULL*
15954	Original Release Date	2005-12-06	2 [->text]	500	*NULL*
15955	ASIN	B000BF0DJC	2 [->text]	600	*NULL*
15956	MSRP	USD 18.98	2 [->text]	700	*NULL*
15975	Image	*NULL*	1 [->image]	10000	*NULL*
15957	Track Number	1-1	2 [->text]	12001	16025
15958	Track Number	1-2	2 [->text]	12002	16027

[NOTE: Descriptors truncated for demonstration purposes]

id	title	descriptor	descriptor_type_id	priority	has_subdescriptor
15973	Track Number	1-17	2 [->text]	12017	16057
15974	Track Number	1-18	2 [->text]	12018	16059

The example product descriptor query above returns a has_descriptor column which can be used to trigger the pulling of sub-descriptors.

Digging further, you can retrieve the indicated sub-descriptors for the Track Number descriptor with the following query:

```
SELECT product_subdescriptors.id, product_subdescriptors.title,
product_subdescriptors.descriptor, product_subdescriptors.descriptor_type_id,
product_subdescriptors.priority, product_sub_subdescriptors.id as has_subdescriptor
FROM product_subdescriptors
LEFT JOIN product_sub_subdescriptors ON product_sub_subdescriptors.product_subdescriptor_id =
product_subdescriptors.id
WHERE product_subdescriptors.product_descriptor_id = 15974;
```

This will return the following sub-descriptors:

id	title	descriptor	descriptor_type_id	priority	has_subdescriptor
16059	Title	I'm Sprung 2	2 `[->text]`	10000	*NULL*
16060	Track Time	6:51	2 `[->text]`	10000	*NULL*

To determine a complete and current list of all of the Product Descriptors available, use DISTINCT SQL Queries, for example:

```
SELECT DISTINCT(title) FROM product_descriptors;
SELECT DISTINCT(title) FROM product_subdescriptors;
SELECT DISTINCT(title) FROM product_sub_subdescriptors;
```

To determine a complete and current list of all Product Sub-Descriptors and the Descriptors they describe, use the following query:

```
SELECT product_subdescriptors.title as Sub_Descriptor_Title, product_descriptors.title as
Descriptor_Title
FROM product_subdescriptors
LEFT JOIN product_descriptors on product_subdescriptors.product_descriptor_id =
product_descriptors.id
GROUP BY product_subdescriptors.title, product_descriptors.title
ORDER BY product_subdescriptors.title;
```

Finally, to determine a complete and current list of all Product Sub-Sub-Descriptors and the Sub-Descriptors they describe, use the following query:

```
SELECT product_sub_subdescriptors.title as Sub_Sub_Descriptor_Title,
product_subdescriptors.title as Sub_Descriptor_Title
FROM product_sub_subdescriptors
LEFT JOIN product_subdescriptors on product_sub_subdescriptors.product_subdescriptor_id =
product_subdescriptors.id
GROUP BY product_sub_subdescriptors.title, product_subdescriptors.title
ORDER BY product_sub_subdescriptors.title;
```

Global Categories, General Categories and Thought Categories:

Tables:

```
| general_categories
| general_category_descriptors
| general_category_subdescriptors

| global_categories
| global_category_descriptors
| global_category_subdescriptors

| thought_categories
| thought_category_descriptors
| thought_category_sub_subdescriptors
| thought_category_subdescriptors
```

Global Categories are the highest level of category in the database. They consist of the utmost global categories: TV Shows, Commercials, Movies, etc.

General Categories are mid-level categories which define groups under Global. For example, Years for Movies (2005); Brands for Commercials (Geico), or a specific TV Show (Scrubs).

Thought Categories are the individual elements of General Categories, such as a specific Movie (King Kong), Commercial (Caveman at the Airport), or a specific Episode (My Musical).

These three categories, when used together, allow you to query the database on a global, general and/or thought level. For example, the following query will show you the global, general and thought categories for Thought Category ID # 3379:

```
SELECT global_category, general_category, general_categories.alphabetize_on as general_category_alphabetize_on, thought_category,
thought_categories.alphabetize_on as thought_category_alphabetize_on
FROM thought_categories
LEFT JOIN general_categories ON thought_categories.general_category_id = general_categories.id
LEFT JOIN global_categories ON general_categories.global_category_id = global_categories.id
WHERE thought_categories.id=3379
```

Which gives the following result:

global_category	general_category	general_category_alphabetize_on	thought_category	thought_category_alphabetize_on
TV Shows	Desperate Housewives	DES	Remember (Part I)	REM

The alphabetize_on columns are crucial in determining if a name starts with an article, such as The or A, that you may wish to ignore in an alphabetical ordering. For example, *"A Crying Shame"*, Thought Category ID # 858.

Awards:

Tables:
| awards
| award_categories
| award_groups

Awards can be entered both in pure RDBMS and as Text entries. For example, to allow for ease of entry and consistency, there are award groups, categories and reference types. However, to allow for awards, contacts, products, etc. that are not already in the TWM database, the text-based column entry can always be used.

Awards can also apply to a contact, thought category, general category, a product and/or a mixture of all of the above.

To pull the awards for Greg J. Grande (Contact ID: 22), the following query could be used:

```
SELECT award_groups.award_group, award_categories.award_category, award_category_text,
award_status, award_year, thought_category_id, general_category_id, contact_text,
thought_category_text, general_category_text
FROM awards
LEFT JOIN award_groups ON award_groups.id = awards.award_group_id
LEFT JOIN award_categories ON award_categories.id = awards.award_category_id
WHERE contact_id = 22
ORDER BY award_status desc, award_year desc, award_groups.award_group;
```

And the results of this query on the sample database:

award_group	award_category	award_category_text	award_status	award_year	thought_category_id	general_category_id	contact_text	thought_category_text	general_category_text
Emmy Awards (Creative Arts)	Outstanding Art Direction For A Multi-Camera Series	*NULL*	nominated	2003	0 [->Link not found]	2 [->Friends]	*NULL*	The One In Barbados (Parts I & II)	*NULL*
Emmy Awards (Creative Arts)	Outstanding Art Direction For A Multi-Camera Series	*NULL*	nominated	2002	0 [->Link not found]	2 [->Friends]	*NULL*	The One Where Rachel Has A Baby (Parts I & II)	*NULL*
Emmy Awards (Creative Arts)	Outstanding Art Direction For A Multi-Camera Series	*NULL*	nominated	2001	0 [->Link not found]	2 [->Friends]	*NULL*	Monica And Chandler's Wedding	*NULL*
NULL	*NULL*	Outstanding Individual Achievement In Art Direction For A Series	nominated	1995	0 [->Link not found]	2 [->Friends]	*NULL*	The One Where Rachel Finds Out	*NULL*