Chapter 2

Ontology

A fundamental characteristic of the Web is the structural information it encodes, which has been obscured through there being a number of different ways to represent different kinds of structure. A key contribution of my work is providing a *uniform interface* to these different types of structure. This is done through the Squeal ontology, represented by a SQL database schema, which serves as the user's view of the Web, allowing the user to focus on structural information without needing to be aware of the representation.

2.1 Types of Information on the Web

The World-Wide Web consists of *pages* of data, addressed by *uniform resource locators (URLs).* We focus on pages written in *hypertext markup language (HTML).* HTML includes *tags* and *attributes* to specify intra-document information, such as a page's logical structure or how it should be visually rendered. HTML also incorporates inter-document information, such as what pages are connected by *hyperlinks.* In this section, I discuss what type of information about Web pages can be accessed through Squeal.

2.1.1 Uniform Resource Locators (URLs)

A URL consists of the following components:

- An access scheme, such as "http", which stands for "hypertext transfer protocol", followed by a colon (:).
- A machine name, preceded by two forward slashes (//).
- Optionally, a machine port number, preceded by a colon (:). The default port number is 80.
- A Unix-style (case-sensitive, slash-delimited) path name, including the file name.
- Optionally, a label preceded by a pound sign (#), specifying a reference within a document.

We concern ourself with the documents accessible via http. Some examples of URLs are:

- http://www.ai.mit.edu:80
- http://www.ai.mit.edu/projects/cs101/#course
- http://www.amazon.com/exec/obidos/ats-query/4304-6019362-659411

```
<H1>Massachusetts Institute of Technology</H1>

<H2>Spotlights</H2>

<UL>
<LI> Felice Frankel: The Power of Images
<LI> An Evening of Conversation with Noam Chomsky and Kathleen Cleaver
</UL>
```

Figure 2-1: HTML Specification of Internal Document Structure. The corresponding display is shown in Figure 2-2

Massachusetts Institute of Technology

Spotlights

- Felice Frankel: The Power of Images
- An Evening of Conversation with Noam Chomsky and Kathleen Cleaver

Figure 2-2: Appearance of HTML Internal Document Structure (Figure 2-1).

2.1.2 Hypertext Markup Language

All documents, except for some leaf nodes, are written in *hypertext markup language* (HTML) [32], which consists of ordinary text augmented with *tags* and *attributes*. Tag names are followed by zero or more attribute assignments, all enclosed in angle brackets. In some cases, subsequent text is affected until the appearance of a concluding tag, prefixed with a forward slash (/). This enclosed text is referred to as *anchor text*. For example, this piece of hypertext:

```
<FONT face="Times" color=red>fire</font>
```

contains the tag "FONT", which indicates that the anchor text ("fire") should appear in the font specified by the attributes, i.e., in the face "Times" and the color red. HTML is used to express information about a document's internal structure, appearance, hyperlinks, and meta-information. We will discuss these in the following separate sections, although a single tag can serve more than one of these purposes.

Internal structure

An HTML document can contain internal structure in the form of *headers* and *lists*. The tag "H1" is used to designate a top-level header, "H2" a second-level header, and so forth through "H6", a sixth-level header. Four tags are used to designate different types of simple lists: "OL" (ordered list), "UL" (unordered list), "MENU", and "DIR" (directory list). The "LI" (list item) tag appears before each item. Figure 2-1 shows some hypertext specifying header and list information. Figure 2-2 how it might be displayed to a user.

Appearance

Tags are used to express the font family (e.g., Times), style (e.g., italics), size, and alignment of text on a page. They are also used to include graphics, white space, and lines. Authors frequently misuse header tags for their effects on display, even though the commands are meant for structural information.

Hyperlinks

Tags are used to designate hyperlinks from the current page to another location on the Web, typically another page. A link is designated with the "A" ("anchor") tag, and the "HREF" attribute is used to express the destination. When displayed by a browser, clicking on the anchor text causes the destination to be loaded.

Meta-information

The "META" tag is used to express meta-information meant for Web tools rather than for browsers. The "NAME" and "CONTENT" attributes are used to specify the type of information being given and its value, respectively. Common values of the "NAME" attribute are "description" and "keywords", both used by search engines to categorize, rank, and display URLs. While broader standards for metadata, such as the Dublin Metadata Core Element Set [44], have been proposed, none are in wide use.

2.2 Database Relations

Rather than invent a new language to represent structural information, we turn to a representation designed for this purpose: relational databases. This section presents the relational database schema through which the Squeal user views the Web. Once in this representation, Structured Query Language (SQL)can be used to access it.

The Squeal relations are divided into four categories: (1) basic relations, (2) tag and attribute relations, (3) relations built on tags, and (4) relations for second-hand information. Relation and column names appear in **bold face** and function names appear in *slant* font. After a relation is defined, its use is demonstrated through an annotated transcript, in which user commands are shown in **boldface** and annotations in *italics*. To keep the query results readable, I sometimes do not show all of the results returned by the system.

2.2.1 Basic Relations

In order to provide useful information about the Web, it is necessary that strings, URLs, and Web pages can be represented. This is done through the four basic relations:

1. **valstring**, which maps an integer to a string
2. **urls**, which describes the set of URLs representing the same page
3. **parse**, which represents the components of a URL
4. **page**, which represents information corresponding to a single page

valstring

The **valstring** relation, shown in Table 2.1 is used to map an integer **value_id** to a string. The **value_id**s are frequently used in other relations to represent strings. The primary purpose of this indirection it to represent strings as cleanly and efficiently as possible. The **vcvalue** field is used for strings of less than 256 characters, the maximum length of a varchar (variable-length character array) object supported by Microsoft SQL Server. The unlimited **textvalue** field is only used when needed for larger strings, because it requires at least 2048 bytes of storage. For every legal row in the relation, one of **vcvalue** or **textvalue** is null.[1] The SQL definition for this and the other relations is shown in appendix A.

[1]Because of limitations on operations involving TEXT fields, Microsoft SQL Server cannot enforce this constraint.

valstring		
colname	*type*	*notes*
value_id	INT	
vcvalue	VARCHAR(255)	
textvalue	TEXT	

Table 2.1: The **valstring** Relation. Note that this is the SQL data structure representing the table, indicated by the double border.

The *value_id* function returns the **value_id** associated with a string, adding a new row to the relation if necessary. The *value* function converts a **value_id** into a string.

```
CREATE PROCEDURE value @id INT
AS
SELECT COALESCE(CONVERT(TEXT,vcvalue),textvalue)
FROM valstring
WHERE value_id = @id
```

urls

The **urls** relation, shown in Table 2.2 is used to map one or more **value_id**s representing URL strings to a unique **url_id**. If there were a one-to-one correspondence between URL strings and **url_id**s, it would not be needed, but multiple URL strings can reference the same page. For example, both "http://www.ai.mit.edu" and "http://www.ai.mit.edu/index.html" reference the same document, so they would have the same **url_id**, with different **variant** numbers.

The *url_id* function returns the **url_id** associated with a URL string, adding a new row to **urls** and **value_id** if necessary. The *url* function converts a **url_id** into a URL string. This can be defined in SQL:

```
CREATE PROCEDURE url @id INT
AS
SELECT COALESCE(CONVERT(TEXT,v.vcvalue),v.textvalue)
FROM valstring v, urls u
WHERE v.value_id = u.url_id AND u.url_id = @id
```

The **url_id** associated with a string can change because a **url_id** is initially assigned when the URL is encountered, although it cannot be determined whether two URLs correspond to the same page until their contents have been retrieved. It is impractical to retrieve entire pages when a **url_id** is defined, so the only other option is to allow a **url_id** to change. For this reason, the user should call the *url_id* function when a **url_id** is needed, rather than storing the result of calling *url_id*. If the user creates a table with a column of type **url_id**, the value in the table will change if the **url_id** changes.

Strictly speaking, there is no way to tell if two URLs reference the same file or merely two copies of a file. Squeal uses the heuristic that two URLs reference the same page and should have the same **url_id** if they retrieve identical contents and the host component of the URLs are the same strings (ignoring case differences). Corresponding pages at separate mirror sites would not have the same **url_id**.

parse

The **parse** relation, shown in Table 2.3 encodes a parsed version of a URL string. This is useful for determining if two URLs are on the same host machine or if one is above the other in

urls		
colname	*type*	*notes*
url_id	INT	
value_id	INT	unique
variant	INT	default: 0

Table 2.2: The **urls** relation

parse		
colname	*type*	*notes*
url_value_id	INT	
component	CHAR(5)	{"host", "port", "path", "ref"}
value	VARCHAR(255)	
depth	INT	null when component $\neq$ "path"

Table 2.3: The **parse** relation

the directory hierarchy. The **component** field contains "host", "port", "path", or "ref", and the **value** field shows the associated value. For the "path" component, the **depth** within the file hierarchy is also indicated, starting at the file name. Table 2.4 shows the **parse** relation for the URL "http://www.ai.mit.edu/people/index.html". Note that one **url_id** can have multiple parses if multiple URLs correspond to that page.

page

The **page** relation, shown in Table 2.5 caches pages so they are not unnecessarily retrieved from the Web. It contains information from the last time a page was retrieved, including the text of a page, its size, a time stamp and the results of a Message Digest 5 (MD5) hash function [34] on the page's text, providing a quick way to tell if two pages contain the same text. The **valid** field encodes the results of the last attempted retrieval of the page: "y" if it was successfully retrieved, "n" if it could not be retrieved, or "b" if the page was over the maximum page size limit (to be discussed in the next chapter).

Figure 2-3 includes a transcript demonstrating these relations.

2.2.2 Tag and attribute relations

Because they encode the structure of pages and hyperlinks, tags and attributes are fundamental to Squeal, allowing users to access this information.

url_value_id	**component**	**value**	**depth**
950	host	www.ai.mit.edu	0
950	port	80	0
950	path	index.html	1
950	path	people	2
950	ref	[null]	0

Table 2.4: Parse relation for **url_value_id** = *value_id*('http://www.ai.mit.edu/people/index.html')

What **value_id** *corresponds to the string "http://www.ai.mit.edu/index.html"?*
SELECT value_id
FROM valstring
WHERE vcvalue='http://www.ai.mit.edu/index.html';

value_id
2606

What **url_id** *corresponds to the url represented by "http://www.ai.mit.edu/index.html"?*
SELECT url_id
FROM urls
WHERE value_id = 2606;

url_id
817

The url_id operator is a shortcut for the above sequence:
PRINT *url_id*(**'http://www.ai.mit.edu/index.html');**
817

Show all the known URLs corresponding to this **url_id***.*
SELECT *
FROM urls
WHERE url_id = 817;

url_id	value_id	variant
817	2606	1
817	11128	2

Show all the known URLs for this page.
SELECT v.vcvalue
FROM valstring v, urls u
WHERE u.url_id = 817 AND v.value_id = u.value_id;

vcvalue
http://www.ai.mit.edu
http://www.ai.mit.edu/index.html

How many bytes long is this page?
SELECT bytes FROM page WHERE url_id = 817;

bytes
5618

What is the host component of the URL?
SELECT value
FROM parse
WHERE component='host' AND url_value_id = 817;

value
www.ai.mit.edu

Figure 2-3: Transcript Illustrating the Basic Relations

page		
colname	*type*	*notes*
url_id	INT	
contents	TEXT	
bytes	INT	
md5	CHAR(32)	
when	DATETIME	

Table 2.5: The **page** relation

tag		
colname	*type*	*notes*
url_id	INT	
tag_id	INT	unique
name	CHAR(15)	
startOffset	INT	
endOffset	INT	

Table 2.6: The **tag** relation

tag

The **tag** relation, shown in Table 2.6, contains one line for each tag or set of paired tags encountered. (Single tags include "<hr>" (horizontal rule); paired tags include "<title>" ... "</title>".) Information stored includes the **url_id**, tag name, start and end character offsets within the document, and a unique **tag_id**, which is used to reference attributes.

att

The **att** relation, shown in Table 2.7, contains one line for each attribute name and value in a document. The name is a short string and the value an index into **valstring**. I treat the anchor text enclosed by a pair of tags to be the value of the attribute "anchor".

Figure 2-4 has a transcript demonstrating the use of the **tag** and **attribute** relations.

2.2.3 Relations built on tags

Information about the structure of headers and lists on a page and hyperlinks across pages can be inferred from the information in the **tag** and **att** tables, but we provide the user with specialized tables **header**, **list**, and **link** for greater convenience.

att		
colname	*type*	*notes*
tag_id	INT	
name	CHAR(15)	
value_id	INT	

Table 2.7: The **att** relation

How many times does each type of tag appear on "http://www.mit.edu"?

```
SELECT name, COUNT(*)
FROM tag
WHERE url_id = url_id('www.mit.edu')
GROUP BY name;
```

name	
!DOCTYPE	1
A	42
BODY	1
FNORD	15
HEAD	1
HTML	1
IMG	10
P	16
TITLE	1

What tags appear within the first hundred characters?

```
SELECT name, tag_id
FROM tag
WHERE url_id = url_id('www.mit.edu') AND startOffset < 100;
```

name	tag_id
!DOCTYPE	1087
HTML	1088
HEAD	1089
TITLE	1090

Show me the attributes of the "title" tag.

```
SELECT *
FROM att
WHERE tag_id = 1090;
```

tag_id	name	value_id
1090	anchor	1105

What is the text associated with the "title" tag?

```
SELECT vcvalue
FROM valstring
WHERE value_id = 1105;
```

vcvalue
SIPB WWW Server Home Page

What are the attributes associated with "img" (image) tags in the first 2000 characters?

```
SELECT a.name, v.vcvalue, t.tag_id
FROM valstring v, tag t, att a
WHERE t.url_id = url_id('www.mit.edu')
    AND t.startOffset < 2000 AND t.name='img'
    AND a.tag_id = t.tag_id AND v.value_id = a.value_id
```

name	vcvalue	tag_id
alt	MIT SIPB WWW Server	1098
src	http://www.mit.edu/gif/MITSIPB_cropped.GIF	1098
align	bottom	1126
src	http://www.mit.edu/gif/Icon_NEW.GIF	1126

Figure 2-4: Transcript Demonstrating the **tag** and **att** Relations

header		
colname	*type*	*notes*
url_id	INT	
struct	BINARY(6)	
ord	INT	
offset	INT	

Table 2.8: The **header** relation

list		
colname	*type*	*notes*
url_id	INT	
struct	BINARY(6)	
ord	INT	
offset	INT	

Table 2.9: The **list** relation

header

As discussed above, HTML supports six levels of headings (H1–H6). Through the **header** table, shown in Table 2.8, Squeal keeps track of the levels of headings at each point of change. The **struct** column consists of an array of six bytes, each of which corresponds to one level of heading. At the beginning of a page, all six bytes are initialized to zero. Whenever a <Hn> **tag** is encountered, byte $(n - 1)$ is incremented, and all bytes with higher offsets are zeroed. This can be seen in the above transcript. The **ord** element is an ordinal number incremented with every new database line. The following transcript provides an example:

Show the header structure of "www.ai.mit.edu/people/ellens/cv.html"
SELECT h.struct, t.name, h.offset
FROM header h, tag t
WHERE h.url_id=437 AND t.url_id = 437
AND h.offset = t.startOffset;

struct	**name**	**offset**
1 0 0 0 0 0	H1	129
1 1 0 0 0 0	H2	586
1 2 0 0 0 0	H2	1820
1 3 0 0 0 0	H2	3246
1 4 0 0 0 0	H2	4691
1 5 0 0 0 0	H2	5591
1 6 0 0 0 0	H2	6518
1 6 1 0 0 0	H3	6680
1 6 2 0 0 0	H3	10618
1 7 0 0 0 0	H2	13645
1 8 0 0 0 0	H2	16948

list

As discussed above, There are four types of lists supported by HTML: ordered lists, unordered lists, menus, and directories. In the **list** relation, they are not differentiated among. Lists can be arbitrarily nested in HTML; Squeal keeps track of up to six levels of nesting of lists, via the **list**

- Initialization
  1. Set each of the 6 bytes of **struct** to 0.
  2. Set each of the 6 bytes of internal structure **lmax** to 0.
- When a new list begins
  1. If all elements of **struct** are 0, set i to 0. Otherwise, set i to the highest index such that $\mathbf{struct}[i] \neq 0$.
  2. $\mathbf{lmax}[i] \leftarrow \mathbf{lmax}[i] + 1$
  3. $\mathbf{struct}[i] \leftarrow \mathbf{lmax}[i]$
  4. $\forall j \mid i < j < 6, \mathbf{lmax}[j] \leftarrow 0$
- When a list ends
  1. Set i to the highest index such that $\mathbf{struct}[i] \neq 0$.
  2. $\mathbf{struct}[i] \leftarrow 0$

Figure 2-5: Rules for Managing List Structure

link		
colname	*type*	*notes*
source_url_id	INT	
anchor_value_id	INT	
dest_url_id	INT	
hstruct	BINARY(6)	
lstruct	BINARY(6)	

Table 2.10: The **link** relation

relation, shown in Table 2.9. The **struct** column contains an array of six bytes. The rules for managing the bytes are shown in Figure 2-5, and an example is shown in Figure 2-6.

link

The **link** relation, shown in Table 2.10, contains one line for each hyperlink encountered, storing the source **url_id**, anchor text **value_id**, destination **url_id**, and the **list** structure and **header** structure at which the hyperlink occurs. Because all of the information in the **link** table exists in other tables, it can be defined as a SQL view (virtual table), as shown in Figure 2-7 (although for practical reasons, it is implemented as an ordinary table).
Figure 2-8 contains a transcript demonstrating the **link** relation.

2.2.4 Relations for second-hand information

In addition to direct information about the text and hyperlinks of Web pages, indirect information can be obtained from Web tools. It is important however to distinguish between verified and second-hand information, hence the **rcontains** table, for *reported contents* of pages, and the **rlink** table, for *reported links*.

0	1	2	3	4	5	
-	-	-	-	-	-	
0	0	0	0	0	0	[start of document]
1	0	0	0	0	0	List1 begins
1	1	0	0	0	0	Sublist 1a begins
1	0	0	0	0	0	Sublist 1a ends
0	0	0	0	0	0	List 1 ends
2	0	0	0	0	0	List 2 begins
2	1	0	0	0	0	Sublist 2a begins
2	0	0	0	0	0	Sublist 2a ends
2	2	0	0	0	0	Sublist 2b begins
2	0	0	0	0	0	Sublist 2b ends
0	0	0	0	0	0	List 2 ends

Figure 2-6: Example of **struct** values of **list** at list tags

```
CREATE VIEW link
AS
SELECT DISTINCT t.url_id AS source_url_id,
    aAnchor.value_id AS anchor_value_id,
    u.url_id AS dest_url_id,
    h1.struct AS hstruct,
    l1.struct AS lstruct
FROM tag t, att aAnchor, att aHref, urls u,
    header h1, header h2, list l1, list l2
WHERE t.name = ‘a’ AND t.tag_id = aAnchor.tag_id AND
    aAnchor.name = ‘anchor’ AND t.tag_id = aHref.tag_id AND
    aHref.name = ‘href’ AND u.value_id = aHref.value_id AND
    h1.url_id = t.url_id AND h2.url_id = t.url_id AND
    h1.offset < t.startOffset AND h2.offset > t.startOffset AND
    h1.ord+1 = h2.ord AND l1.url_id = t.url_id AND
    l2.url_id = t.url_id AND l1.offset < t.startOffset AND
    l2.offset > t.startOffset AND l1.ord+1 = l2.ord
```

Figure 2-7: A SQL definition of **link** in terms of other relations

Show the value_id and anchor text of all links from "www.ai.mit.edu/projects/haystack/"
SELECT l.anchor_value_id, v.vcvalue
FROM link l, valstring v
WHERE l.source_url_id = 503 AND l.anchor_value_id = v.value_id;

anchor_value_id	**vcvalue**
115	Publications
120	People
1686	Introduction
1687	Download
1688	Installation
1689	Online Manual

Give me the url_id and URL of some pages that point to "www.ai.mit.edu"
SELECT l.dest_url_id AS url_id, v.vcvalue AS value
FROM link l, valstring v, urls u
WHERE l.dest_url_id = 1 AND u.url_id = l.source_url_id
AND v.value_id = u.value_id;

url_id	**value**
3	http://www.tns.lcs.mit.edu/
4	http://www-mtl.mit.edu/
5	http://www.csg.lcs.mit.edu:8001/
6	http://farnsworth.mit.edu/other_servers.html
7	http://www.frob.com/
9	http://cag-www.lcs.mit.edu/

See what other links from "www.tns.lcs.mit.edu" are in the same list as the link to "www.ai.mit.edu"
SELECT DISTINCT l.dest_url_id, v.vcvalue
FROM link l, link lAI, valstring v, urls u
WHERE l.source_url_id = 3 AND lAI.source_url_id = 3
AND lAI.dest_url_id = 1 AND l.lstruct = lAI.lstruct
AND l.dest_url_id = u.url_id AND v.value_id = u.value_id;

dest_url_id	**vcvalue**
1	http://www.ai.mit.edu/
2	http://www.lcs.mit.edu/
34	http://web.mit.edu/
78	http://farnsworth.mit.edu/
79	http://www.mit.edu/

Figure 2-8: Transcript Demonstrating the **link** Relation

rcontains		
colname	*type*	*notes*
url_id	INT	
value	VARCHAR(255)	
helper	CHAR(16)	

Table 2.11: The **rcontains** relation

rlink		
colname	*type*	*notes*
source_url_id	INT	
anchor	VARCHAR(255)	
dest_url_id	INT	
helper	CHAR(16)	

Table 2.12: The **rlink** relation

rcontains

The **rcontains** relation, shown in Table 2.11, indicates the claim by the search engine specified in the **helper** column that a certain page contains a certain piece of text. In the current implementation, legal values for the **helper** column are "altavista" and "lycos". (Details on how many such pages the **rcontains** table holds will be discussed in Section 3.6.) The SQL definition of **rcontains** is:

rlink

The **rlink** relation, shown in Table 2.12, contains search engine claims about hyperlinks among pages. Because of how the underlying search engine (AltaVista) works, the **source_url_id** field will always have a non-null value, as will the **helper** field, but exactly one of **anchor** and **dest_url_id** will be null. In other words, a line can indicate the reported source and destination of a hyperlink, or it can indicate a page and anchor text that reportedly appears in a hyperlink. The missing information, assuming the reported link actually exists, can be extracted from the **link** relation. The transcript in Figure 2-9 demonstrates the use of the **rlink** and **rcontains** relations.

2.3 Summary

We have shown the Squeal ontology, i.e., what information on the Web is available to the Squeal user and the abstraction through which it is accessed. Specifically, the user has access to the raw and parsed version of each URL (**urls**, **parse**), the raw text, size, and hash value of each page (**page**), including all of the **tag** and attribute (**att**) values. This information can be accessed directly or through the **header**, **list**, and **link** relations. The user can also ask what pages reportedly contain a specified piece of text (**rcontains**) or a described hyperlink (**rlink**). In the next chapter, we provide more detail on the syntax of user queries and how Squeal provides the information.

Find pages that reportedly contain "MIT AI Lab".

```
SELECT r.url_id, v.vcvalue
FROM rcontains r, valstring v, urls u
WHERE r.value LIKE '%MIT AI Lab%'
   AND u.url_id = r.url_id AND v.value_id = u.value_id;
```

url_id	**vcvalue**
829	http://www.ai.mit.edu/weather/weather.html
830	http://www.mic.atr.co.jp/ hajiri/html/AI.html
828	http://www.neurop2.ruhr-uni-bochum.de/MIT_lit.html
837	http://ymiseja.eenet.ee/IT/maillist/msg9.html
838	http://www.pmms.cam.ac.uk/ gjm11/jargon/jargappA.html
839	http://www.csa.runnet.ru/AI/faqs/lisp_6.htm

Find pages that reportedly point to "www.ai.mit.edu/people/ellens/gender.html".

```
SELECT r.source_url_id, v.vcvalue
FROM rlink r, valstring v, urls u
WHERE r.dest_url_id = 591 AND u.url_id = r.source_url_id
AND v.value_id = u.value_id;
```

source_url_id	**vcvalue**
542	http://www.cs.vassar.edu/
846	http://www.mcs.salford.ac.uk/TOUR/022.htm
847	http://www.cmu.edu/adm/uri/oncampus/opps.html
851	http://yaron.clever.net/sheizaf3.shtml

Find pages that reportedly contain "computer science" as anchor text.

```
SELECT r.source_url_id, v.vcvalue
FROM rlink r, valstring v, urls u
WHERE r.anchor = 'computer science'
AND u.url_id = r.source_url_id and v.value_id = u.value_id;
```

source_url_id	**vcvalue**
868	http://longwood.cs.ucf.edu/
869	http://emmy.smith.edu/
870	http://www.icsi.berkeley.edu/
871	http://www.engr.uvic.ca/

Figure 2-9: Transcript Demonstrating the **rlink** and **rcontains** Relations