Data Integration Exercise

Technologies for Information Systems
November, 7th 2011

Politecnico di Milano
Dipartimento di Elettronica e Informazione

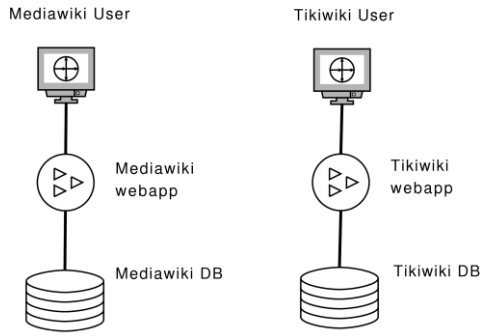
Problem setting

MediaWiki and TikiWiki are two wikilike content management systems with a DB backend.

At the University of Wasomao some of the professors use MediaWiki while some others use TikiWiki.

The university wants to allow the professor to use one of the two systems to access the pages of the other one, while keeping unaltered the original applications.

Current Situation



Original DB backends

MEDIAWIKI-DB

PAGE (pid, title, namespace)

REVISION (rid, page, text-id)

TEXT(tid, plain-text)

USER (uid, nickname, password);

PAGELINKS (from, title, namespace)

PAGE contains the metadata about a page

REVISION contains different versions of the same page

TEXT contains individual text of each revision.

USER contains the information about CMS users

PAGELINKS stores the links between pages as an id of the source page and the title and namespace of the target page.

Original DB backends (2)

TIKIWIKI-DB

PAGES (page-id, version, page-title, user-id, text)

COMMENTS (page-id, user-id, comment)

USER (user-id, real-name, username, email, pwd, path-to-picture)

LINKS (from-page, to-page)

PAGES contains the information about pages (their versions and the actual texts)

COMMENTS stores user comments to pages, *USER* contains information about users

LINKS represents the links between pages as id pairs of the source page and of the target page.

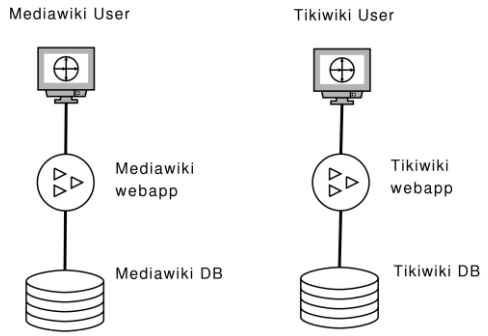
Exercise

1) Propose a data-integration solution based on the requirements above
2) Provide, for each input data source, the reverse engineering from the logical to conceptual schema (ER graph)
3) Design the mappings between the two sources
4) Consider the following query:

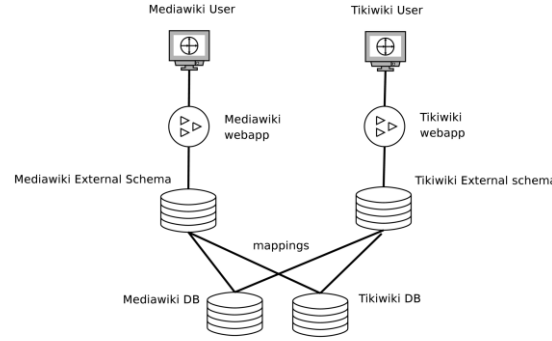
   Return all the pages pointing to a page titled "data integration"

   - Show how this query will be rewritten when posed on both systems

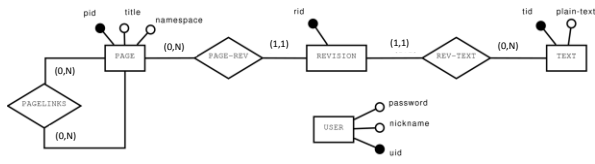
Current Situation



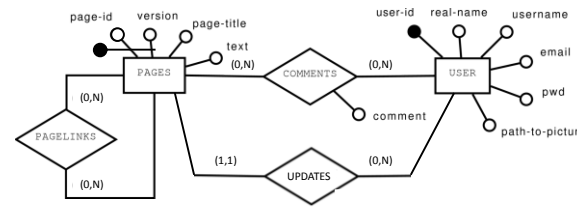
Proposed solution



Mediawiki reverse engineering



Tikiwiki reverse engineering



Mapping design

To ease the query processing we design two set of mappings:

- TIKITOMEDIA which considers the MEDIAWIKI schema as the target schema and the TIKIWIKI schema as the source schema.
- MEDIATOTIKI which does the opposite.

TIKITOMEDIA mappings

CREATE VIEW **PAGE** (pid, title, namespace) AS

```
SELECT      *
FROM        MEDIA.PAGE

UNION

SELECT      DISTINCT TP.page-id, TP.page-title, 'main'
FROM        TIKI.PAGES as TP
```

CREATE VIEW **TEXT** (tid, plain-text) AS

```
SELECT      *
FROM        MEDIA.TEXT

UNION

SELECT      ENCRYPT_TDES(TP.text, 'abc'), TP.text
FROM        TIKI.PAGE as TP
```

TIKITOMEDIA mappings (2)

CREATE VIEW **REVISION** (rid, page, text-id) AS

```
SELECT      *
FROM        MEDIA.REVISION

UNION

SELECT      CONCAT(TP.page-id, '$', TP.version),
            TP. page-id, ENCRYPT_TDES(TP.text, 'abc')
FROM        FROM TIKI.PAGES AS TP
```

CREATE VIEW **USER** (uid, nickname, password) AS

```
SELECT      *
FROM        MEDIA.USER

UNION

SELECT      TU.user-id, TU.username, TU.pwd
FROM        TIKI.USER
```


TIKITOMEDIA mappings (3)

CREATE VIEW **PAGELINKS** (from, title, namespace) AS

```
SELECT      *
FROM        MEDIA.PAGELINKS

UNION

SELECT      TL.from-page, TP.page-title, 'main'
FROM        TIKI.LINKS AS TL, TIKI.PAGES as TP
WHERE       TP.page-id = TL.to-page
```


MEDIATOTIKI mappings

CREATE VIEW **PAGES** (page-id, version, page-title, user_id, text) AS

```
SELECT      *
FROM        TIKI.PAGES

UNION

SELECT      MP.pid, MR.rid, MP.title, null, MT.plain-text
FROM        MEDIA.PAGE as MP, MEDIA.REVISION as MR,
            MEDIA.TEXT as MT
WHERE       MP.pid = MR.page and MR.text-id = MT.tid
```

CREATE VIEW **USER** (user-id, real-name, username, email, pwd, path-to-picture) AS

```
SELECT      *
FROM        TIKI.USER

UNION

SELECT      MU.uid, null, MU.nickname, null, MU.password, null
FROM        MEDIA.USER as MU
```


MEDIATOTIKI mappings (2)

CREATE VIEW **LINKS** (from-page, to-page) AS

```
SELECT      *
FROM        TIKI.LINKS

UNION

SELECT      MPL.from, MP.p-id
FROM        MEDIA.PAGELINKS as MPL, MEDIA.PAGE as MP
WHERE       MPL.title = TP.title
            AND MPL.namespace = MP.namespace
```

CREATE VIEW **COMMENTS** (page-id, user-id, comment) AS

```
SELECT      *
FROM        TIKI.COMMENTS
```


Query

Return all the pages pointing to a page titled "data integration"

Query Rewriting – Tikiwiki external schema - GAV

Tikiwiki

Q(x) ← LINKS(x,y) ∧ PAGES(y,_,w,_,_) ∧ w='data integration'

where

v1: LINKS(x,y) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q)

v2: PAGES (x,y,z,j,v) ← PAGE(x,z,_) ∧ REVISION(y,x,s) ∧ TEXT (s,v)

Query Rewriting - Tikiwiki - GAV (2)

Tikiwiki

Q(x) ← LINKS(x,y) ∧ PAGES(y,_,w,_,_) ∧ w='data integration'

where

v1: LINKS(x,y) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q)

v2: PAGES (x,y,z,j,v) ← PAGE(x,z,_) ∧ REVISION(y,x,s) ∧ TEXT (s,v)

rewritten query (simple unfolding – GAV mappings)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ PAGE(y,w,_) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ w='data integration'

Query Rewriting - Tikiwiki - GAV (3)

Tikiwiki

Q(x) ← LINKS(x,y) ∧ PAGES(y,_,w,_,_) ∧ w='data integration'

where

v1: LINKS(x,y) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q)

v2: PAGES (x,y,z,j,v) ← PAGE(x,z,_) ∧ REVISION(y,x,s) ∧ TEXT (s,v)

rewritten query (simple unfolding – GAV mappings)

Unification possible because y is the primary key (w → z)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ PAGE(y,w,_) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ w='data integration'

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ z='data integration'

Query Rewriting - Tikiwiki - GAV (4)

Tikiwiki

Q(x) ← LINKS(x,y) ∧ PAGES(y,_,w,_,_) ∧ w='data integration'

where

v1: LINKS(x,y) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q)

v2: PAGES (x,y,z,j,v) ← PAGE(x,z,_) ∧ REVISION(y,x,s) ∧ TEXT (s,v)

rewritten query (simple unfolding – GAV mappings)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ PAGE(y,w,_) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ w='data integration'

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ z='data integration'

We know from the ER schema that there is a foreign key constraint from REVISION to TEXT. Therefore the TEXT(s,_) atom can be deleted, since it does not impose any extra condition (the variables in the atom are not used in any other atom apart the two atoms involved in the foreign key constraint), if we assume the data to be consistent with the constraints.

Query Rewriting - Tikiwiki - GAV (5)

Tikiwiki

Q(x) ← LINKS(x,y) ∧ PAGES(y,_,w,_,_) ∧ w='data integration'

where

v1: LINKS(x,y) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q)

v2: PAGES (x,y,z,j,v) ← PAGE(x,z,_) ∧ REVISION(y,x,s) ∧ TEXT (s,v)

rewritten query (simple unfolding – GAV mappings)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ PAGE(y,w,_) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ w='data integration'

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧
REVISION(_,y,s) ∧ TEXT(s,_)∧ z='data integration'

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧
REVISION(_,y,s) ∧ z='data integration'

Query Rewriting - Tikiwiki - GAV (6)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ REVISION(_,y,s) ∧ z='data integration'

Similarly, we remove the REVISION(_,y,s) atom since there must be a revision related to each page (as we can see from the ER schema).

Query Rewriting - Tikiwiki - GAV (7)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ REVISION(_,y,s) ∧ z='data integration'

Similarly, we remove the REVISION(_,y,s) atom since there must be a revision related to each page (as we can see from the ER schema).

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ z='data integration'

Query Rewriting – Tikiwiki – GAV (8)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ REVISION(_,y,s) ∧ z='data integration'

Similarly, we remove the REVISION(_,y,s) atom since there must be a revision related to each page (as we can see from the ER schema).

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ z='data integration'

Although this is not in the ER schema, it might be reasonable to assume that the following inclusion dependency holds:
PAGELINKS[title, namespaces] ⊆ PAGE[title, namespaces]

Query Rewriting – Tikiwiki – GAV (9)

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ REVISION(_,y,s) ∧ z='data integration'

Similarly, we remove the REVISION(_,y,s) atom since there must be a revision related to each page (as we can see from the ER schema).

Q(x) ← PAGELINKS(x,z,q) ∧ PAGE(y,z,q) ∧ z='data integration'

Although this is not in the ER schema, it might be reasonable to assume that the following inclusion dependency holds:
PAGELINKS[title, namespaces] ⊆ PAGE[title, namespaces]
If so, for the same reasons as before, we can simplify the query as follows:

Q(x) ← PAGELINKS(x,z,q) ∧ z='data integration'

Query Rewriting – Mediawiki external schema-GAV

Mediawiki

Q(x) ← PAGELINKS(x,y,_) ∧ y='data integration'

where

v1: PAGELINKS(x,y,'main') ← LINKS(x,w) ∧ PAGES(w,_,y,_,_)

Query Rewriting – Mediawiki-GAV(1)

Mediawiki

Q(x) ← PAGELINKS(x,y,_) ∧ y='data integration'

where

v1: PAGELINKS(x,y,'main') ← LINKS(x,w) ∧ PAGES(w,_,y,_,_)

We rewrite the view in such a way that the head only contains distinct variables:

v1: PAGELINKS(x,y,z) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ z = 'main'

Query Rewriting – Mediawiki-GAV(2)

Mediawiki

Q(x) ← PAGELINKS(x,y,_) ∧ y='data integration'

where

v1: PAGELINKS(x,y,'main') ← LINKS(x,w) ∧ PAGES(w,_,y,_,_)

We rewrite the view in such a way that the head only contains distinct variables:

v1: PAGELINKS(x,y,z) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ z = 'main'

rewritten query (simple unfolding – GAV mappings)

Q(x) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ y='data integration'∧ _ = 'main'

Query Rewriting – Mediawiki-GAV(2)

Mediawiki

Q(x) ← PAGELINKS(x,y,_) ∧ y='data integration'

where

v1: PAGELINKS(x,y,'main') ← LINKS(x,w) ∧ PAGES(w,_,y,_,_)

We rewrite the view in such a way that the head only contains distinct variables:

v1: PAGELINKS(x,y,z) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ z = 'main'

rewritten query (simple unfolding – GAV mappings)

Q(x) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ y='data integration'∧ _ = 'main'

In the last atom there is a variable (_) that is not used elsewhere, so we can safely remove it:

Query Rewriting – Mediawiki-GAV(2)

Mediawiki

Q(x) ← PAGELINKS(x,y,_) ∧ y='data integration'

where

v1: PAGELINKS(x,y,'main') ← LINKS(x,w) ∧ PAGES(w,_,y,_,_)

We rewrite the view in such a way that the head only contains distinct variables:

v1: PAGELINKS(x,y,z) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ z = 'main'

rewritten query (simple unfolding – GAV mappings)

Q(x) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ y='data integration'∧ _ = 'main'

In the last atom there is a variable (_) that is not used elsewhere, so we can safely remove it:

Q(x) ← LINKS(x,w) ∧ PAGES(w,_,y,_,_) ∧ y='data integration'