Database and Information Systems

Homework 5 Solutions

Suppose you are starting a new website called Nile.biz, a clone of Amazon.com. You have three suppliers, two of whom supply music (and whose schemas were actually excerpted from CIS 550 Homework 4 submissions), and the third of which supplies CDs, movies, and books. You would like to have a unified, integrated view of this data so you can support your website. The three sources have the following schemas:

1. CD(key, title, publisherName, genre, artistXref: refs Artist)
   CDsong(key: refs CD, song)
   Participates(artistKey: refs Artist, publisherName)
   Artist(artistKey, name, street, city, state, zip, country)
   ArtistFormerName(artistKey: refs Artist, name)

2. CD(cdkey, artistKey, title, genre, year)
   Track(cdkey: refs CD, song)
   CDsubtitle(cdkey: refs CD, subtitle)
   CDlabel(cdkey: refs CD, label)
   Artist(artistKey, group, website, street address, box, city, state, country, zip)
   KnownAs(artistKey, name)
   Plays(artistKey: refs Artist, instrumentType)

3. Item(iKey, classification, title, category, year)
   SubItem(iKey, title)
   Publishes(iKey: refs Item, pKey: refs Publisher)
   Publisher(pKey, name, street, city, state, zip)
   DevelopedBy(iKey: refs Item, aKey: refs AuthorOrContributor, role)
   AuthorOrContributor(aKey, name, nickname, website, street, city, state, zip, country)
   Stock(iKey, count, location)

1 Mediated schema

There are many ways of representing the mediated schema. One possibility is to adopt a format similar to Source 3, with generic Item and SubItem relations. However, we chose
to create a separate relation for each of the different types of items, as well as for the various people involved in creating a book, CD, or movie.

Movie(mid, title, year, genre)
Book(isbn, title, year, genre)
Chapter(isbn, title)
CD(cid, title, year, genre)
Song(cid: refs CD, title)
StockMovie(mid: refs Movie, count, location)
StockCD(cid: refs CD, count, location)
StockBook(isbn: refs Book, count, location)
Author(aKey: refs Person, isbn: refs Book)
Artist(aKey: refs Person, cid: refs CD)
Star(aKey: refs Person, mid: refs Movie)
Director(aKey: refs Person, mid: refs Movie)
Person(aKey, name, website, group, street, city, state, zip, country)
FormerName(aKey: refs Person, name)
Plays(aKey: refs Person, instrument)
Publishes(item, pid: refs Publisher)
Publisher(pid, name, street, city, zip, country)

We will strictly use local-as-view mappings from this schema.

2 Mappings from Source 1

s1:CD(k, t, p, g, a) :- CD(k, t, _, g), Artist(a, k), Publishes(k, pk), Publisher(pk, p)
s1:CDsong(k, s) :- Song(k, s)
s1:Participates(a, pk) :- Artist(a, k), Publishes(k, pk), Publisher(pk, p)
s1:Artist(a, n, s, ci, st, z, c) :- Person(a, n, _, s, ci, st, z, c, _)
s1:ArtistFormerName(a, n) :-FormerName(a, n)

3 Mappings from Source 2

s2:CD(k, a, t, g, y) :- CD(k, t, y, g), Artist(a, k)
s2:Track(k, s) :- Song(k, s)
s2:CDlabel(k, l) :- Publishes(k, pk), Publisher(pk, p)
s2:Artist(a, g, w, s, c, st, co, z) :- Person(a, w, g, s, c, st, z, co, _)
s2:KnownAs(a, n) :- Person(a, n, _, _, _, _, _, _)
s2:KnownAs(a, n) :-FormerName(a, n)
s2:Plays(a, i) :- Plays(a, i)
We have chosen to drop CDsubtitle, just to demonstrate that some relations may not be of general interest in the mediated schema. We could equally well have created a mediated relation for subtitle.

4 Mappings from Source 3

\[
\begin{align*}
\text{s3:Item} & (i, \text{“movie”}, t, g, y) :- \text{Movie} (i, t, y, g) \\
\text{s3:Item} & (i, \text{“CD”}, t, g, y) :- \text{CD} (i, t, y, g) \\
\text{s3:Item} & (i, \text{“book”}, t, g, y) :- \text{Book} (i, t, y, g) \\
\text{s3:SubItem} & (i, t) :- \text{Song} (i, t) \\
\text{s3:SubItem} & (i, t) :- \text{CD} (i, t) \\
\text{s3:SubItem} & (i, t) :- \text{Book} (i, t) \\
\text{s3:Publishes} & (i, t) :- \text{Publishes} (i, p) \\
\text{s3:Publisher} & (p, n, s, ci, st, z) :- \text{Publisher} (p, n, s, ci, st, z, \text{“USA”}) \\
\text{s3:DevelopedBy} & (i, a, \text{“artist”}) :- \text{Artist} (a, i) \\
\text{s3:DevelopedBy} & (i, a, \text{“author”}) :- \text{Author} (a, i) \\
\text{s3:DevelopedBy} & (i, a, \text{“star”}) :- \text{Star} (a, i) \\
\text{s3:DevelopedBy} & (i, a, \text{“director”}) :- \text{Director} (a, i) \\
\text{s3:AuthorOrContributor} & (i, n, null, w, s, c, st, z, co) :- \text{Person} (i, n, w, g, s, c, st, z, co) \\
\text{s3:Stock} & (i, c, l) :- \text{StockMovie} (i, c, l) \\
\text{s3:Stock} & (i, c, l) :- \text{StockCD} (i, c, l) \\
\text{s3:Stock} & (i, c, l) :- \text{StockBook} (i, c, l)
\end{align*}
\]

Note that the mappings to \text{s3:SubItem} don't properly preserve information about what type of subitem was there: if we try to invert the mapping, we won't know which tuples should be inserted from \text{s3:SubItem} into \text{Song}, \text{CD}, or \text{Book}. Ditto for \text{Stock} and its mappings. This information can only be captured with a mapping expressed in the reverse direction — a global-as-view mapping. In the general case, local-as-view, used here, is more powerful and precise, but there are many cases where global-as-view is preferable. Some people have tried to combine the two mapping styles.