Generació de codi amb WebRatio

Memòria del projecte

Estudiant: **Martí Manrique Farcy**

Enginyeria en Informàtica

Consultor: **Jordi Ceballos Villach**

Data Lliurament: **18/06/2007**

Resum

Darrerament, l'enginyeria d'aplicacions web estava enfocada únicament al desenvolupament d'aplicacions web que només permetien navegar per la informació emmagatzemada en les bases de dades. Darrerament, el llenguatges de modelatge web s'estan ampliant per també permetre la modificació de la informació. Un exemple és WebML, un llenguatge que permet modelar aplicacions web de forma conceptual i que consisteix en un model de dades o conceptual, un model de navegació que defineix la composició de les pàgines i els enllaços entre elles; i un model de presentació enfocat en l'estil de presentació de la informació.

Un dels inconvenients que presenten aquestes extensions del WebML és l'augment de la complexitat dels models. La seva definició pot introduir errors que afecten significativament a la qualitat ja que sovint la generació del codi de l'aplicació es realitza de forma automàtica a partir del models de navegació. Respecte a la qualitat d'un model navegació, hi han dues propietats importants: completesa i correctesa. Un model de navegació és complet si conté un camí d'execució per modificar cada dada inclosa en el model de dades. A més, un model també és correcte si permet un camí d'execució que mantingui les dades en un estat consistent segons les regles definides en el model de dades.

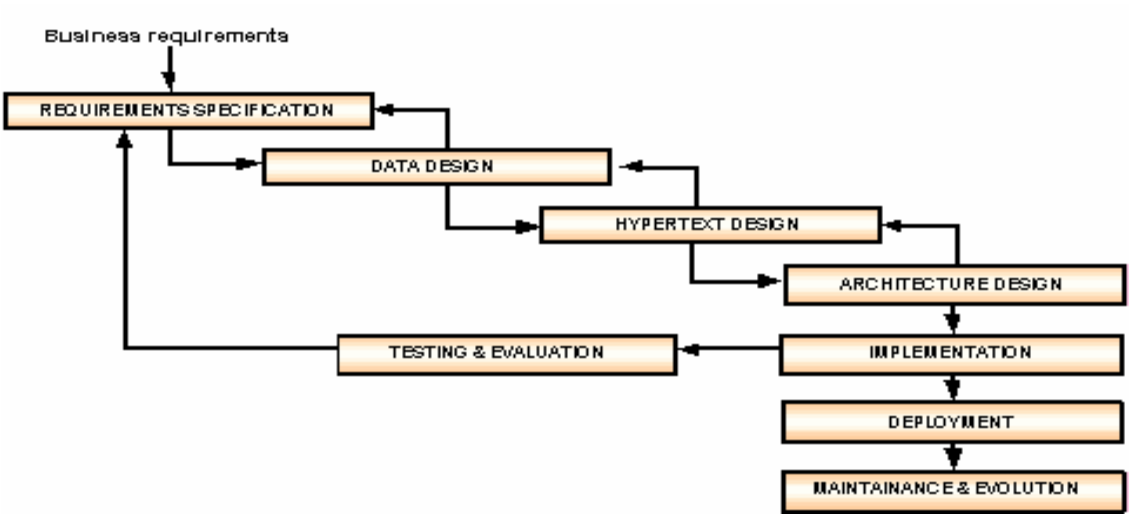
L'objectiu d'aquest PFC és la implementació d'una eina que s'encarregui de transformar un model de dades en un model de navegació complet i correcte. Per fer-ho, el programa WebRatio suporta completament el llenguatge WebML i s'utilitzarà com a dissenyador dels models de dades i també com a eina per comprovar els models de navegació generats.

Taula de continguts

Introducció

Actualment, les aplicacions web a gran escala són llocs web que normalment necessiten accedir a grans volums de dades emmagatzemades normalment en sistemes de gestió de base de dades (SGBD). La construcció d'aquestes aplicacions és una activitat que implica vàries tasques que van des del disseny de les estructures de dades per guardar el contingut, la definició de interfícies hypertext per navegar per la informació, el disseny de estils de presentació efectius, la integració entre arquitectures robustes i d'alt rendiment amb sistemes obsolets o serveis externs.

A finals dels anys noranta i principis del 2000 van començar a sorgir diferents metodologies i llenguatges per adreçar aquests problemes. Una d'aquestes solucions és WebML (Web Modelling Language), un llenguatge conceptual que suporta totes les activitats i perspectives del disseny de Web sites. WebML proporciona un conjunt d'especificacions gràfiques juntament amb un procés de disseny complet, que pot ser assistit amb eines de disseny visual com WebRatio. Tal com es mostra a la següent figura, el procés de disseny es desglossa en vàries fases que s'apliquen de forma incremental o iterativa.



Les fases de disseny de dades i disseny hypertext són les activitats que es veuen més afectades per l'adopció del WebML. Respecte el disseny de dades, WebML proporciona un model, anomenat model de dades, que és una adaptació dels models conceptuals utilitzats en altres disciplines com el disseny de bases de dades o la representació de coneixement. Els elements fonamentals d'aquest model de

dades són les entitats, definides com a contenidors de dades i relacions, definides com a connexions semàntiques entre entitats.

El disseny hypertext especifica la composició i la navegació del web site. La composició descriu quines pàgines i unitats componen la interfície de l'usuari. Les pàgines d'un web site són els contenidors d'informació que realment es mostren a l'usuari, mentre que les unitats són els elements que permeten connectar el model hypertext amb el model de dades i que s'utilitzen per consultar i mantenir la informació. La navegació del web site s'especifica a través d'enllaços. Un enllaç es pot definir entre dues unitats o pàgines.
Una de les extensions més rellevant del model de navegació WebML és la incorporació de unitats que permeten modificar el contingut de les dades. Aquestes unitats poden implementar-se com a operacions bàsiques (inserció/eliminació/modificació) o bé com a operacions més complexes.

Per definir la qualitat d'un model de navegació respecte a les operacions de modificació de la informació, cal assegurar que es compleixen les següents propietats:

- *Completesa*: un model de navegació és complet respecte al seu model de dades si l'usuari pot realitzar el manteniment de qualssevol dada per mitjà de les unitats incloses al model.
- *Correctesa*: un model de navegació és correcte quan existeix com a mínim un camí de navegació entre pàgines que deixa les dades modificades en un estat consistent.

La transformació d'un model de dades en un model de navegació que compleixi aquestes propietats de qualitat pot ser una tasca complexa ja que els models resultants acostumen a ser difícils de mantenir. El principal objectiu d'aquest projecte és l'automatització d'aquest procés, és a dir, la creació d'una eina que permeti convertir un model de dades en un model de navegació que sigui complet i correcte. Tot i que el model resultant sovint representa una versió preliminar del model definitiu, es redueix el temps de desenvolupament ja que no és necessari que el desenvolupador defineixi l'esquema des de zero.

Objectius

Tal i com s'ha comentat en l'anterior apartat, el principal objectiu d'aquest PFC és el desenvolupament d'una aplicació que permeti transformar de forma automàtica un

model de dades d'una aplicació web en un model de navegació que compleixi les següents propietats:

a) *Complet*: l'usuari pot modificar qualssevol dada a través de les operacions del model.
b) *Correcte*: per cada operació el model conté un camí de navegació que garanteix la consistència de les dades.

Tant el model de dades com el model de navegació estaran expressats en llenguatge WebML (format XML) però s'utilitzarà un programari comercial, WebRatio, per definir el model dades i per examinar el model de navegació resultant. També caldrà assegurar que el model de navegació sigui de fàcil manteniment per l'usuari amb l'eina WebRatio ja que sovint el model només representa una primera versió de l'aplicació.

Altres objectius més genèrics del PFC també són:

- Aprendre a desenvolupar una aplicació de tamany mig utilitzant totes les etapes del desenvolupament de programari: anàlisis, disseny, implementació i prova.
- Conèixer el llenguatge WebML i l'eina WebRatio.

Enfocament i mètode seguit

El desenvolupament de l'aplicació es portarà a terme en tres etapes que coincideixen amb les tres proves d'avaluació continuada (PAC) de l'assignatura:

a) *Primera fase*: l'objectiu de l'aplicació en aquesta primera fase és la lectura d'un model de dades contingut en un projecte WebRatio.

b) *Segona fase*: l'objectiu en aquesta segona fase és transformar el model de dades en un model de navegació representa en forma de graf que compleixi les propietats de completesa i correctesa.

c) *Tercera fase*: en aquesta tercera fase la tasca realitzar consistirà en traduir la representació en graf del model de navegació al llenguatge WebML.

Per cadascuna de les fases anteriors, s'aplicarà el cicle de vida clàssic o en cascada i que consta de les següents etapes: anàlisis dels requeriments, disseny, construcció i proves.

Planificació

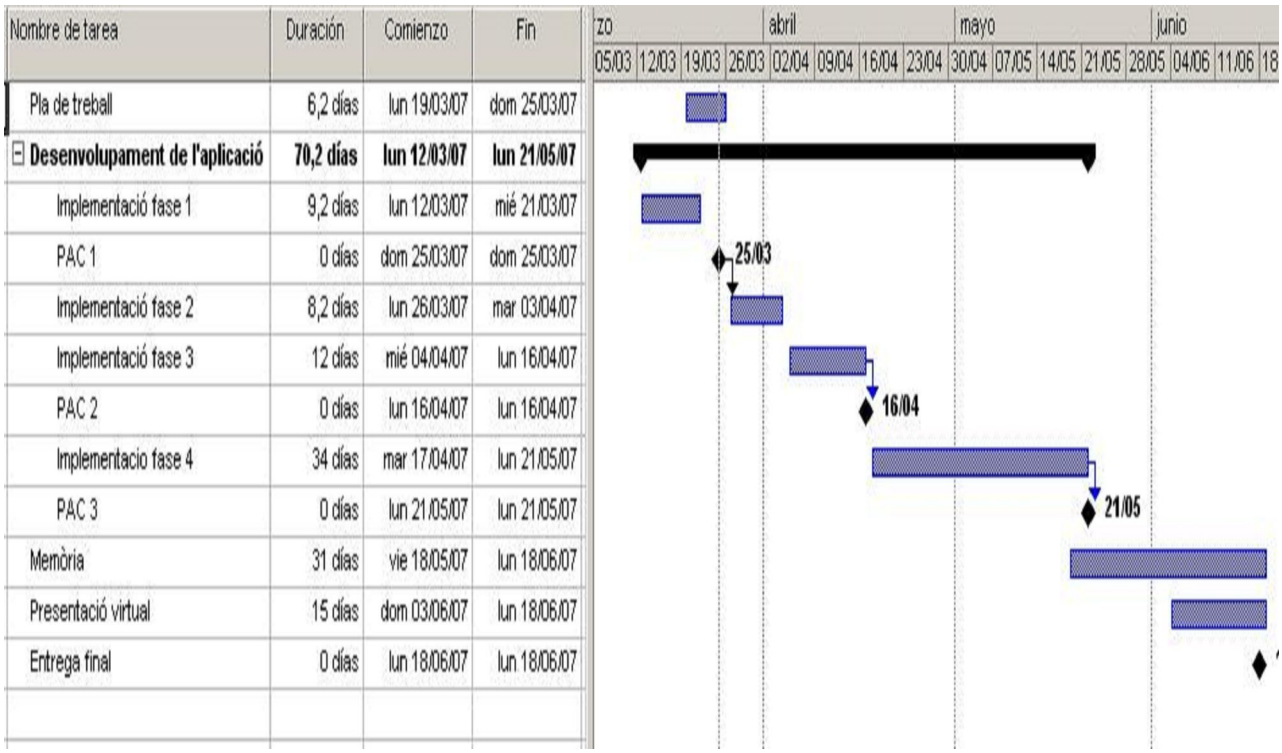
Per realitzar els objectius del projecte, s'estableixen les següents tasques i subtasques:

- *Pla de treball*: document on consten la definició del projecte i la planificació temporal.
- *Desenvolupament de l'aplicació (fase 1)*: Aquesta és la primera fase de l'aplicació i consisteix en el desenvolupament d'un programa que permeti llegir el fitxer XML d'un projecte WebRatio (SDSProject.xml). Es crearà un projecte exemple amb una entitat Client i el programa haurà de generar un site view que contingui una pàgina on es mostrin la llista de clients de la base de dades.
- *Desenvolupament de l'aplicació (fase 2)*: A partir del model conceptual llegit, l'aplicació haurà de generar automàticament un graf navegacional complet. Inicialment es considerarà un exemple reduït amb només les classes Customer i Sale, relacionades amb multiplicitat 1 a N.
- *Desenvolupament de l'aplicació (fase 3)*: A partir del graf anterior, crear un graf navegacional complet i correcte, tenint en consideració les dependències entre les entitats. Si es possible, en aquesta fase també es faran els refactorings necessaris per millorar el graf generat.
- *Desenvolupament de l'aplicació (fase 4)*: finalment, caldrà traduir el graf complet i correcte al format XML que llegeix WebRatio. Aquest és la darrera tasca de l'aplicació i el fitxer resultant haurà de permetre al WebRatio al generació del codi de l'aplicació web.
- *Memòria del projecte*: Recopilació de la informació elaborada durant les tasques anteriors i redacció dels altres capítols de la memòria (Introducció, Bibliografia, Glossari, etc.).
- *Presentació virtual*: Elaboració de la presentació virtual del projecte.

Segons la planificació de l'assignatura, les fites d'aquest projecte serien:

Data	*Fita*	*Productes a lliurar*
25/03/2007	PAC 1	Pla de treball i fase 1 de l'aplicació
16/04/2007	PAC 2	Fases 2 i 3 de l'aplicació (creació de graf complet i correcte)
21/05/2007	PAC 3	Fases 4 de l'aplicació (traducció del graf a XML)
18/06/2007	Entrega final	Memòria del projecte i presentació virtual

A continuació es mostra una diagrama de Gantt de les tasques:



Nombre de tarea	Duración	Comienzo	Fin
Pla de treball	6,2 días	lun 19/03/07	dom 25/03/07
Desenvolupament de l'aplicació	**70,2 días**	**lun 12/03/07**	**lun 21/05/07**
Implementació fase 1	9,2 días	lun 12/03/07	mié 21/03/07
PAC 1	0 días	dom 25/03/07	dom 25/03/07
Implementació fase 2	8,2 días	lun 26/03/07	mar 03/04/07
Implementació fase 3	12 días	mié 04/04/07	lun 16/04/07
PAC 2	0 días	lun 16/04/07	lun 16/04/07
Implementacio fase 4	34 días	mar 17/04/07	lun 21/05/07
PAC 3	0 días	lun 21/05/07	lun 21/05/07
Memòria	31 días	vie 18/05/07	lun 18/06/07
Presentació virtual	15 días	dom 03/06/07	lun 18/06/07
Entrega final	0 días	lun 18/06/07	lun 18/06/07

Descripció dels altres capítols

Els propers capítols tenen per objecte l'anàlisi detallat del objectius plantejats i un disseny de la solució plantejada:

- *Capítol Anàlisi dels requeriments*: en aquest capítol s'analitzarà detalladament les característiques d'un model de dades i d'un model de navegació del llenguatge WebML. A continuació, s'estudiarà exhaustivament les propietats completesa i correctesa del model navegacional.
- *Capítol Disseny*: es presenta una solució per transformar un model de dades en un model de navegació correcte i complet. En aquest capítol, es presenta un diagrama de classes i es comenten els principals atributs i mètodes de cada classe.
- *Capítol Proves*: en aquest capítol es detallen les proves realitzades sobre el programa i els resultats obtinguts.
- *Capítol Instal·lació*: en aquest capítol es detallen les proves realitzades sobre el programa i els resultats obtinguts.

En el darrer capítol de conclusions es comenten els resultats del projectes i s'exposen futures accions per tal de millorar el programa.

1. Anàlisi dels requeriments

La principal funció del programa és transformar un model de dades en un model navegacional que compleixi les propietats de completesa i correctesa citades anteriorment tal i com es pot observar a la següent figura.

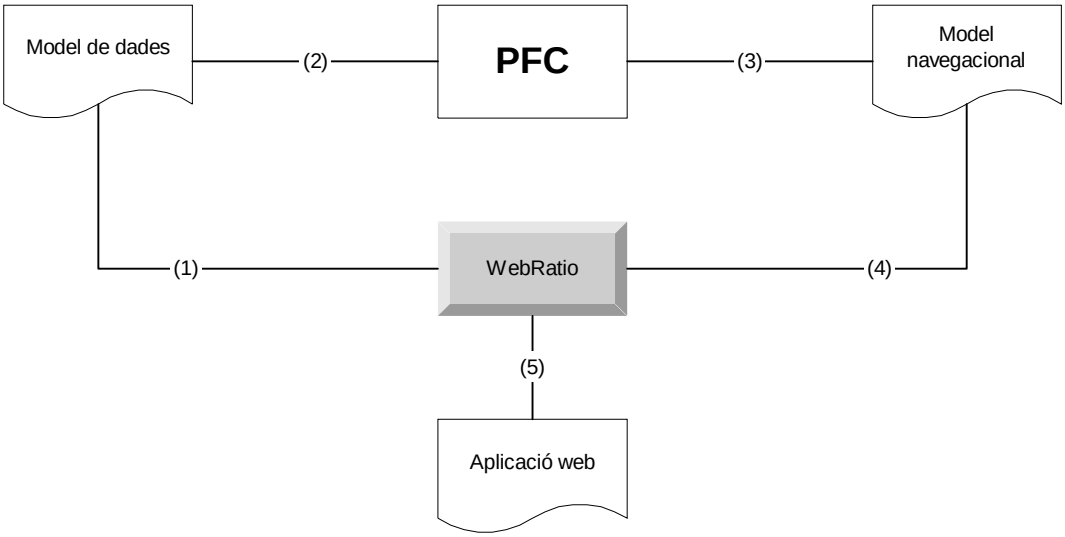


Figura 1. Context del programa

L'eina WebRatio ens ajudarà a dissenyar de forma visual una aplicació amb un determinat model de dades. Aquest projecte es guardarà en un fitxer en format WeBML que el PFC haurà de llegir. El PFC llegirà el projecte i el modificarà per afegir el model navegacional. A continuació, el fitxer resultant (amb el model de dades inicial i el model navegacional) es podrà importar de nou al WebRatio per tal de refinar-lo o ampliar-lo. Quan el desenvolupament de l'aplicació hagi finalitzat, la mateixa eina WebRatio ens permetrà generar el codi de l'aplicació web que es podrà executar en qualssevol navegador web.

A continuació, es descriurà de forma detallada cadascuna de les part del programa.

1.1. Model de dades

Un model de dades o model conceptual defineix el coneixement que una aplicació web ha de tenir present per portar a terme la seva funció. Per exemple, la figura 2. El model conté informació relacionada amb ventes, les línies d'una venta i els productes que pot tenir cada línia. Cada venta pot estar associada amb el client que fa la compra.

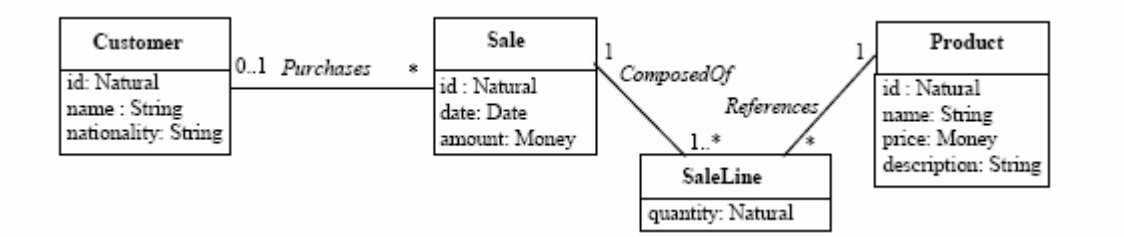


Figura 2. Model de dades d'una aplicació de comerç electrònic

Els elements fonamentals dels models de dades són les entitats, els atributs i les relacions o associacions.

Una entitat descriu les característiques comunes d'un conjunt d'objectes del domini del problema. Cada entitat conté un conjunt d'atributs i cada atribut té associat un determinat tipus de dada (sencer, text, data, etc.). Cada instància d'una entitat es pot identificar a partir de l'atribut OID (unique identifier). Per exemple, l'entitat Venta de la figura 2 es caracteritza pels atributs OID, date i amount.

Una relació té un nom que la descriu i dos participants. El participant és una entitat que té un determinat paper en la relació. Cada relació representa una connexió semàntica entre les dues entitats. Un participant d'una relació té una cardinalitat mínima i màxima. La cardinalitat mínima defineix que el número mínim d'entitats que s'han de relacionar amb l'altre participant. De forma similar, la cardinalitat màxima estableix un número màxim d'entitats que es poden relacionar amb una entitat de l'altre participant. Per exemple, la relació ComposedOf especifica que una venta consisteix en una o més línies, per tant, la cardinalitat mínima 1 i en aquest cas no hi ha una cardinalitat màxima.

Addicionalment, el model de dades pot incloure la definició d'operacions per modificar l'estat de les dades gestionades per l'aplicació web. Les operacions bàsiques que es consideren en aquest projectes són:

Operacions sobre les entitats

- *InsertET (x)*: inserció de l'objecte x dintre de l'entitat ET.
- *DeleteET (x)*: Eliminació de l'objecte x de l'entitat ET.
- *UpdateAiET (v, x)*: Actualització d'un o varis valors dels atributs de l'entitat ET a partir dels valors del vector v.

Operacions sobre les relacions

- *InsertRT(x1, x2):* inserció d'una instància de la relació RT entre els objecte x1 i x2.
- *DeleteRT (x1, x2):* Eliminació de la instància de la a relació RT entre els objectes x1 i x2.

1.2. Model navegacional

El model de navegació o model hypertext especifica l'organització de les interfícies d'usuari de l'aplicació web. La figura 3 mostra un possible model de navegació de l'aplicació presenta a la figura 2. El model presenta la interfície per crear noves ventes amb les línies de venta associades.

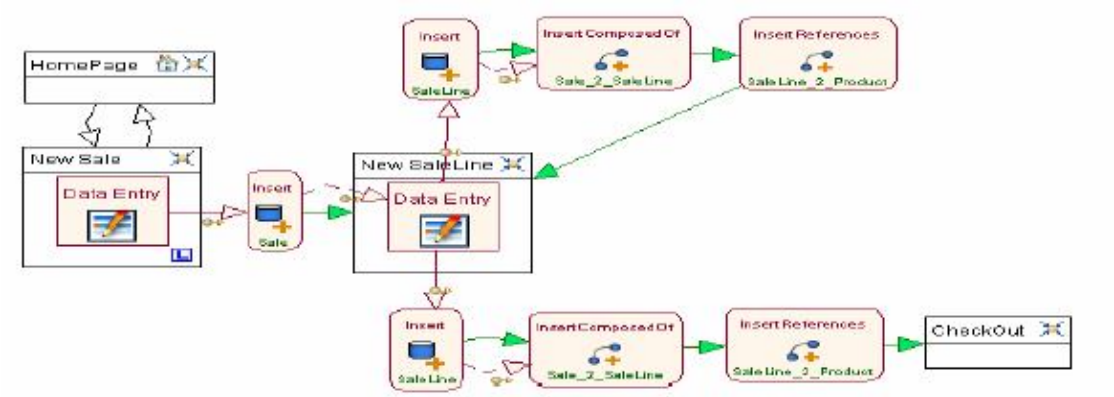


Figura 3. Model navegacional de l'aplicació de e-commerce

Els elements més importants d'un model navegacional són les pàgines i els enllaços. Les pàgines representen els contenidors de informació que realment es mostren a l'usuari. Cada pàgina pot contenir unitats que són els elements que permeten publicar la informació descrita en el model de dades. El llenguatge WebML té set tipus d'unitats predefinides: data, multi-data, index (amb les seves variants muti-selecció i jeràrquica), entry, scroller. Cada unitat està associada amb una entitat a partir de la qual es calcula el contingut.

La navegació d'una aplicació web s'especifica a través dels enllaços. Els enllaços es poden definir entre dues unitats de la mateixa pàgina, entre unitats de diferents pàgines o bé entre pàgines. La informació associada a un enllaç s'anomena context de navegació o simplement context.

El llenguatge WebML permet definir que les operacions associades a les unitats s'executin quan l'usuari activa un enllaç. Per exemple, a la figura 3 es pot observar que quan l'usuari navega a la pàgina New SaleLine des de la pàgina New Sale s'executa l'operació InsertSale (la nova venta és creada a partir de les dades indicades per l'usuari a la pàgina New Page).

En aquesta figura 3 es mostra el procés de crear una venta nova. Des de la pàgina d'inici (Home Page), l'usuari accedeix a la pàgina New Sale. Aquí, l'usuari pot desplaçar-se cap a la pàgina New Sale Line o bé tornar a la pàgina d'inici. Si es navega cap a New Sale Line, es crea una nova venta ja que l'operació Insert està associada a l'enllaç. En aquesta pàgina, l'usuari selecciona el producte que vol comprar i la quantitat. Aleshores, l'usuari pot navegar a la pàgina CheckOut (en aquest moment es crea una nova línia de venta i s'estableixen les connexions entre la nova venta i la nova línia) o comprar més productes seguint l'enllaç cap a la pàgina New Sale Line.

1.3. Propietats completesa i correctesa d'un model navegacional

Tal i com s'ha comentat anterior, l'objectiu del programa és generar un model de navegació que compleixi les següents propietats:

- Completesa: el model ha de tenir un camí de navegació que permeti modificar cadascuna de les dades del model conceptual.
- Correctesa: Tot camí ha de poder deixar les dades en un estat consistent.

Per comprovar si un model de navegació compleix aquestes propietats respecte a un determinat model de dades, aquest model es formalitzarà com un graf on:

- Cada pàgina del model és un node.
- Cada enllaç del model entre una pàgina X i una pàgina Y és un arc entre els nodes corresponents a les pàgines X i Y.
- L'etiqueta d'un arc a emmagatzema una seqüència ordenada d'operacions bàsiques associades a un enllaç.

La figura 4 mostra un graf corresponent al model navegacional de la figura 3.

Figura 4. Representació en graf del model navegacional

1.3.1. Definició de model de navegació complet

Un model de navegació és complet quan els usuaris poden realitzar totes les operacions bàsiques amb els elements del model de dades a través de la interacció d'un conjunt de pàgines del model de navegació. Pel contrari, un model incomplet generarà una aplicació on hi hauran parts de les dades que es podran mantenir.

El conjunt d'operacions no es llegiran del model de dades d'entrada sinó que el programa generarà automàticament les operacions simples descrites en el capítol de model de dades. Per tant, el model de navegació serà complet si existeix un arc dintre del graf que conté cadascuna de les operacions.

1.3.2. Definició de model de navegació correcte

Un model de navegació defineix els possibles camins de navegació possibles d'una aplicació web. Cada camí de navegació admet vàries execucions on cada execució representa un escenari de interacció entre l'usuari i l'aplicació.

Durant una execució, s'apliquen vàries operacions sobre les dades que, en determinades ocasions, poden portar les dades a un estat inconsistent, és a dir, un estat en que s'ha deixat de complir alguna regla d'integritat del model de dades.

També pot succeir que totes les possibles execucions del mateix camí navegacional portin a un estat inconsistent. En aquest cas, aquest camí no és útil i s'hauria d'eliminar de la aplicació per tal de millorar el rendiment i la usabilitat.

Els models de navegació correctes són aquells que no inclouen camins de navegació en els que sempre (per qualssevol execució) porten les dades a un estat inconsistent, independentment dels valors introduïts per l'usuari. Per tant, el model

de navegació generat pel programa serà correcte si tots els camins de navegació són correctes.

1.3.3. Camins de navegació d'un graf

En principi, tots els camins de navegació possibles són vàlids. Tanmateix, en general, no tots els possibles camins ho són perquè normalment l'usuari no acostuma a començar l'execució escollint una pàgina arbitrària sinó que comença des de la pàgina d'inici. De forma similar amb la pàgina de sortida, s'espera que l'usuari navegui fins a una determinada pàgina de sortida que pot ser diferent per cada camí.

Per tant, des d'aquest punt de vista, el programa només considerarà els camins que comencen en una pàgina d'inici i acaben a la mateixa pàgina d'inici després d'haver completat un determinat camí navegacional. En l'exemple, la seqüència de navegació {HomePage, NewSale, NewSaleLine, CheckOut} representen un camí vàlid mentre que {HomePage, NewSale} no s'acceptaria com a vàlid.

La correctesa d'un camí dependrà de les operacions associades als arcs del camí. La idea és que les operacions requereixen la presència d'altres operacions en un arc anterior o posterior per tal de deixar les dades en un estat consistent. Per exemple, un camí que inclou una operació InsertSale requereix que hi hagi una operació InsertComposedOf posterior. En cas contrari, cada execució del camí de navegació finalitzarà en un estat inconsistent degut a que la venta inserida no té una relació amb cap línia de venta.

Quan una operació op1 requereix la presència d'una altra operació op2 es diu que op1 depèn de op2. Les dependències d'una operació són diferents en funció del tipus d'operació (inserció, modificació, ...) i de les cardinalitats de les relacions definides al model de dades. Per tant, un camí de navegació serà correcte si es compleixen les dependències associades a cada operació executada.

A continuació, es definiran les dependències per cadascuna de les operacions suportades pel programa. Una dependència d'una operació es defineix com una tupla <direcció, operació> on operació és el nom de l'operació requerida i direcció indica si l'operació s'ha d'executar abans, després o és irrellevant.

Dependències de les operacions sobre les entitats:

- InsertET (inserció): hi ha una dependència depRT = <->, InsertRT> para cada relació RT on min(ET, RT) >= 1.
- DeleteET (eliminació): hi ha una dependència depRT = <<-, DeleteRT> per cada relació RT on min(ET, RT)>=1.
- UpdateAiET (modificació): No hi ha dependències ja que els canvis de valor dels atributs no afecten a la cardinalitat de les relacions.

Dependències de les operacions sobre les relacions:

- InsertRT (inserció): hi ha una dependència dep = <irrellevant, DeleteRT> si min(ET, RT) = max(ET, RT) per només un participant ETi) o <anterior, InsertET> per cada ET tal que min(ET, RT) = max(ET,RT)>=1.

- DeleteRT (eliminació): hi ha una dependència dep = <irrellevant, InsertRT> si min(ET, RT) = max(ET, RT) per només un participant ETi) o <posterior, DeleteET> per cada ET tal que min(ET, RT) = max(ET,RT)>=1.

Per exemple, si considerem el camí navegacional de la figura 4 que consisteix en la seqüència: {HomePage, NewSale, NewSaleLine, CheckOut}. El conjunt d'operacions seria {InsertSale, InsertSaleLine, InsertComposedOf, InsertReferences}. Per comprovar si el camí és correcte cal definir les dependències de cada operació:

depInsertSale = <posterior, InsertComposedOf>
depInsertSaleLine = <posterior, InsertComposedOf> i <posterior, InsertReferences>
depInsertComposedOf = <anterior, InsertSaleLine> OR <irrellevant, DeleteComposedOf>
depInsertReferences> <anterior, InsertSaleLine> OR <irrellevant, DeleteReferences>

En aquest cas, les dependències de InsertSale es compleixen perquè hi ha l'operació InsertComposedOf després de l'operació InsertSale. Les dependències de InsertSaleLine també es compleixen perquè hi han les operacions InsertComposedOf i InsertReferences. De forma similar, també es compleixen les dependències de les dues darreres operacions, per tant, podem concloure que el camí de navegació és correcte.

2. Disseny

A partir de l'anàlisi anterior, podem establir el procés que el programa haurà de seguir els següents passos per construir el model de navegació:

Pas (1). Lectura d'un model de dades expressat en format WebML.

A partir d'un projecte WebRatio, cal extreure el model de dades en format WebML i carregar-lo a memòria.

Pas (2). Calcular el conjunt d'operacions bàsiques per cada entitat i relació del model.
A partir del model conceptual es crea conjunt d'operacions sobre cada entitat i relació de forma que l'aplicació web pugui realitzar un manteniment complet de la informació. L'eina inclou les següents operacions bàsiques :
Per les entitats es permetran les operacions: inserció, eliminació i modificació.
Per les relacions es permetran les operacions: inserció i eliminació.

Pas (3). Calcular les dependències de cada operació.
Quan ja tenim el conjunt d'operacions bàsiques sobre totes les entitats i relacions del model conceptual, es busquen les dependències entre les operacions en funció de les cardinalitats descrites en les relacions. Aquestes dependències s'utilitzaran en el següent pas per tal de garantir que el model navegacional generat és correcte i, per tant, l'aplicació resultant no ha de permetre executar operacions que portin a un estat inconsistent de la informació.

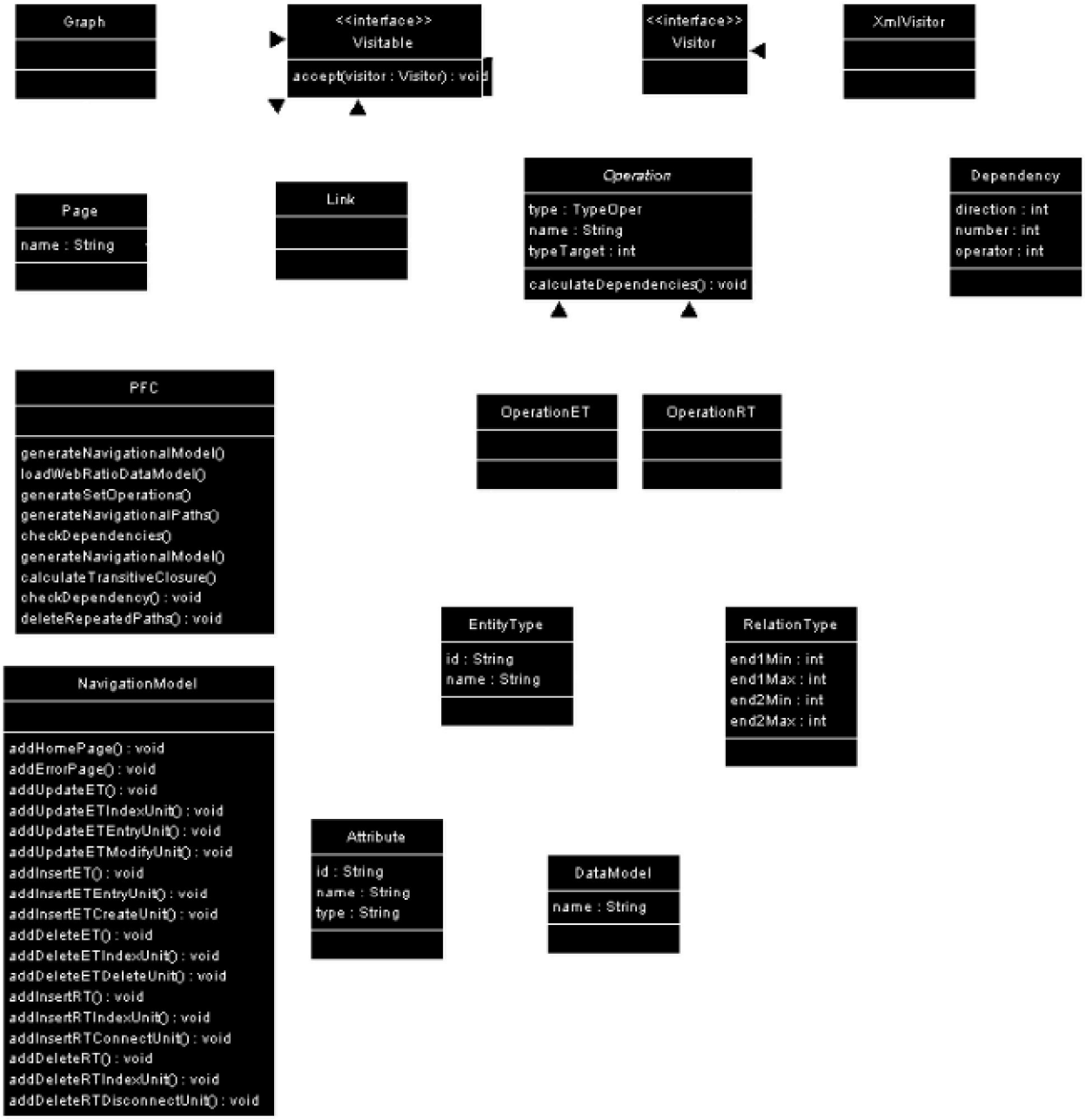
Pas (4). Generar un model navegacional complet i correcte.
En el quart pas, cal construir un model navegacional complet i correcte que està representat. Aquesta model està representat per un graf i té una pàgina d'inici (Home Page) des d'on surten i arriben tots els camins d'execució.
La generació d'un conjunt complet i correcte de camins d'execució passa inicialment per la creació d'una nova pàgina per cadascuna de les operacions del conjunt. Cada pàgina estarà doblement enllaçada a la pàgina d'inici, garantint que el graf és complet ja que cada operació existeix en almenys en un arc. Per assegurar que el graf també és correcte, és necessari que les operacions incloses en tots els camins d'execució compleixin les seves dependències amb altres operacions. Si un camí no és correcte, s'ha d'afegir les operacions necessaris per fer complir les dependències de totes les seves operacions.

Pas (5). Traducció a WebML.
Finalment, l'últim pas consisteix en traduir la representació del graf en un model navegacional en llenguatge WebML per tal que el WebRatio el pugui interpretar. Tot model navegacional ha de tenir dues pàgines bàsiques: la pàgina inicial de l'aplicació o Home Page des de la qual sortiran tots els camins d'execució de l'aplicació i la pàgina d'error que ha d'aparèixer quan una operació no s'ha pogut

realitzar correctament. Sempre que es produeix un error, el model mostrarà la pàgina d'error i anul·larà les operacions executades dintre del camí d'execució. Per realitzar la traducció d'un camí d'execució es farà un recorregut per les operacions incloses i cadascuna d'aquestes operacions es convertirà en una seqüència d'elements del llenguatge WebML que conjuntament implementen la operació sobre la base de dades. A continuació es mostren la implementació en llenguatge WebML de cada possible operació:

2.1. Disseny de classes

A continuació s'especifica un diagrama de classes del programa que implementen els passos anteriors.



A continuació es descriurà els mètodes principals de les classes:

Classes del model de dades:

- DataModel: model de dades.
- EntityType: representa una entitat del model de dades
- Attribute: conté un atribut d'una entitat.
- RelationType: relació entre dues entitats del model de dades.

Classes de les operacions:

- Operation: classe abstracte per una operació genèrica
- OperationET: operació sobre una entitat del model

- OperationRT: operació sobre una relació
- Dependency: dependència d'una operació sobre una altra operació.

Classes del graf:

- Graph: representació del graf complet.
- Page: pàgina d'un graf
- Link: enllaç entre dues pàgines.
- XmlVisitor: classe que implementa el patró de disseny Visitant per obtenir una representació del graf en format XML
- Visitor: interfície que ha de complir una implementació del patró visitor.

Nota: Aquestes classes només es van utilitzar a la PAC2 per estudiar els resultats generats. En l'entrega final es treballa directament amb la llista de camins complets i correctes.

Generació del graf complet i correcte:

- PFC: PFC és la classe principal del projecte i implementa el procés de transformació descrit en els capítols anteriors.

El mètode *generateNavigationalModel* és la única funció pública i té dos paràmetres: inFile que conté el nom del fitxer amb el model de dades en format WebRatio i outFile que tindrà tot el contingut del fitxer d'entrada més el model navegacional generat.

Els altres mètodes de la classe serien:

- **loadWebRatioDataModel**: a partir d'un fitxer de WebRatio carrega el model de dades en les classes descrites anteriorment.
- **GenerateSetOperations**: genera un conjunt bàsic d'operacions sobre un model de dades.
- **generateNavigationalPaths**: crea un graf complet i correcte a partir d'un determinat conjunt d'operacions. Per fer-ho, es criden a les següents funcions:
- **CalculateTransitiveClosure**: aplica de forma recursiva les correccions necessàries per tal de complir les dependències de les seves operacions.
- **CheckDependencies**: comprovar si totes les dependències d'una determinada operació es compleixen en un camí navegacional.
- **CheckDependency**: verificar que una determinada dependència es compleix en el camí navegacional.

- **DeleteRepeatedPaths**: esborra els camins de navegació repetits del graf.
- **generateNavigationalModel**: Guardar un graf en un determinat fitxer en format WebML.

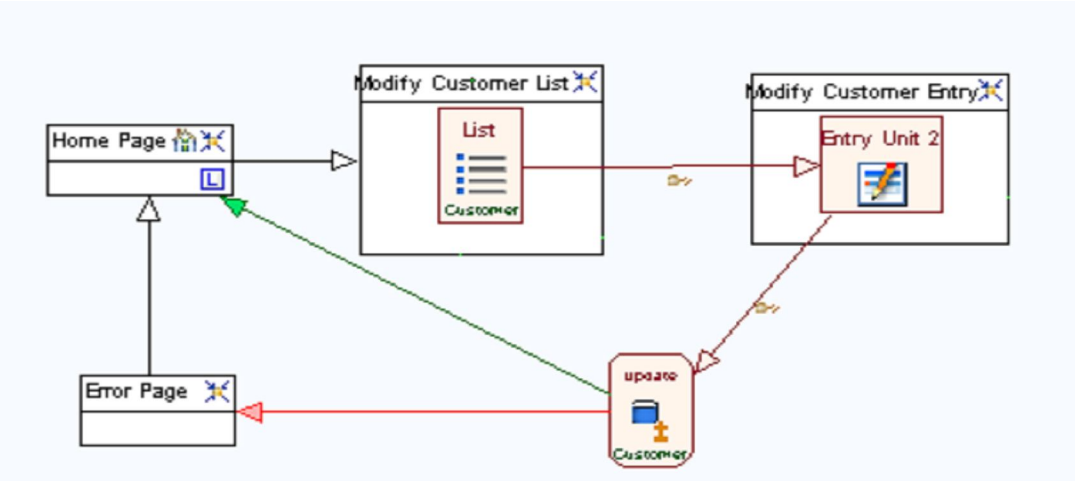
Traducció del graf a llenguatge WebML:

- NavigationModel: Aquesta classe s'encarrega de convertir la llista de camins en un siteview de WebRatio que representa el model navegacional. Si un camí conté més d'una operació, el programa agrupa totes les operacions en una transacció de base de dades per tal que es compleixin les regles d'integritat de la informació. Cal destacar també que la representació del model distribueix els elements en el siteview de forma que sigui visualment llegible quan es carrega en el WebRatio.

La classe inclou un únic servei públic, generate, que tradueix cada operació en unes primitives del llenguatge WebML que conjuntament implementen l'operació:

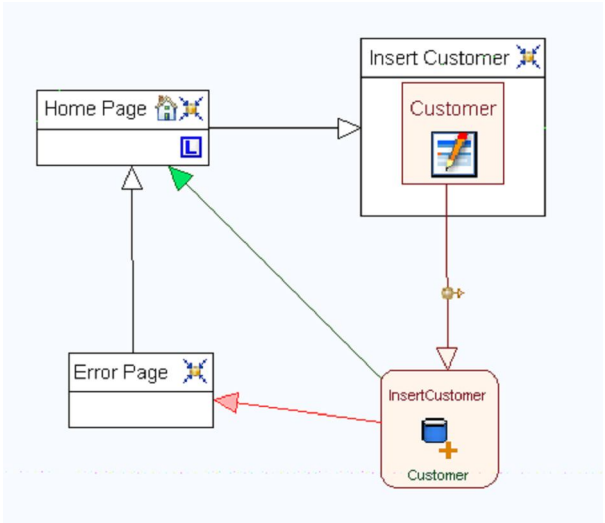
- **addHomePage**: afegeix la pàgina d'inici al siteview.
- **addErrorPage**: afegeix la pàgina d'error al siteview.

Creació d'una operació de modificació sobre una entitat

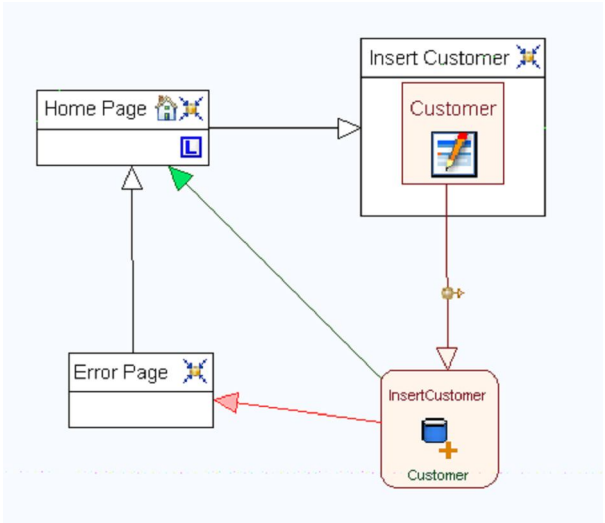


Mètodes implementats:

- *AddUpdateET*: genera una operació de modificació d'atributs mitjançant una crida a les següents subfuncions:
  - *addUpdateETIndexUnit*: genera la pàgina de selecció de la instància que es vol modificar.
  - *AddUpdateETEntryUnit*: genera la pàgina de modificació dels atributs.
  - *AddUpdateETModifyUnit*: genera l'operació de modificació a la base de dades.



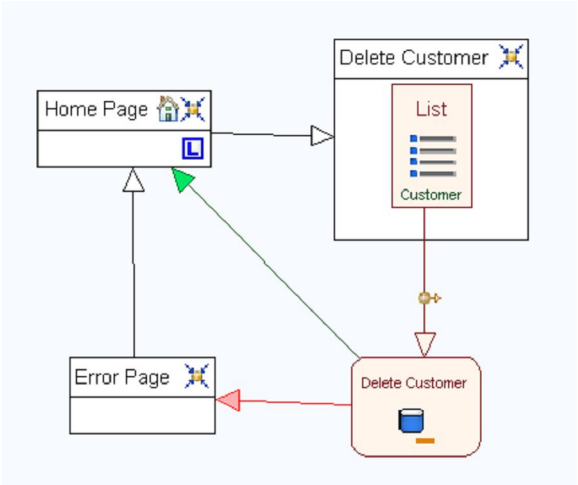
Creació d'una operació de inserció sobre una entitat



Mètodes implementats:

- *AddInsertET*: genera una operació de inserció d'una instància d'entitat i, per fer-ho es crida a les següents subfuncions:
  - *addInsertETEntryUnit*: genera la pàgina d'entrada de dades de la nova instància.
  - *addInsertETCreateUnit*: genera l'operació de inserció sobre la base de dades

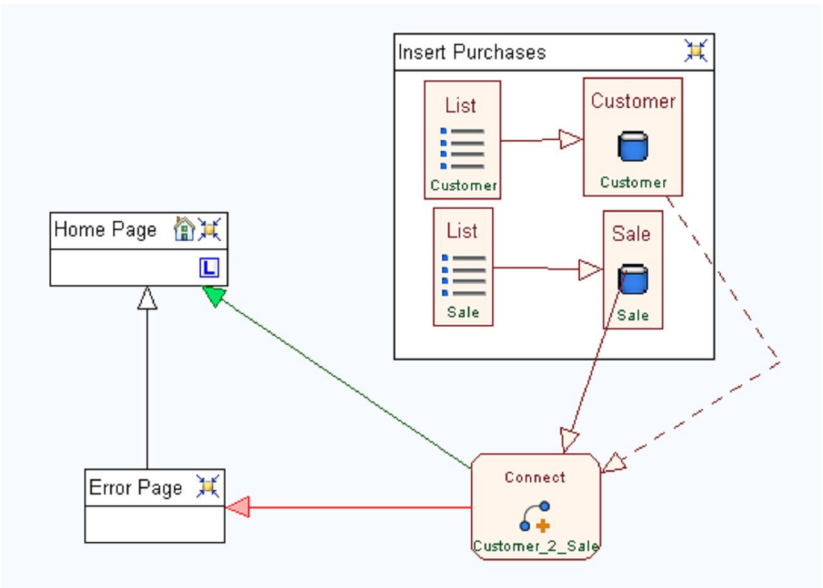
Creació d'una operació d'eliminació sobre una entitat



Mètodes implementats:

- *AddDeleteET*: genera una operació de eliminació d'una instància d'entitat i, per fer-ho es crida a les següents subfuncions:
  - *addDeleteETIndexUnit*: genera la pàgina de selecció de la instància que es vol esborrar.
  - *addDeleteETDeleteUnit*: genera l'operació d'eliminació sobre la base de dades.

Creació d'una operació de inserció sobre una relació

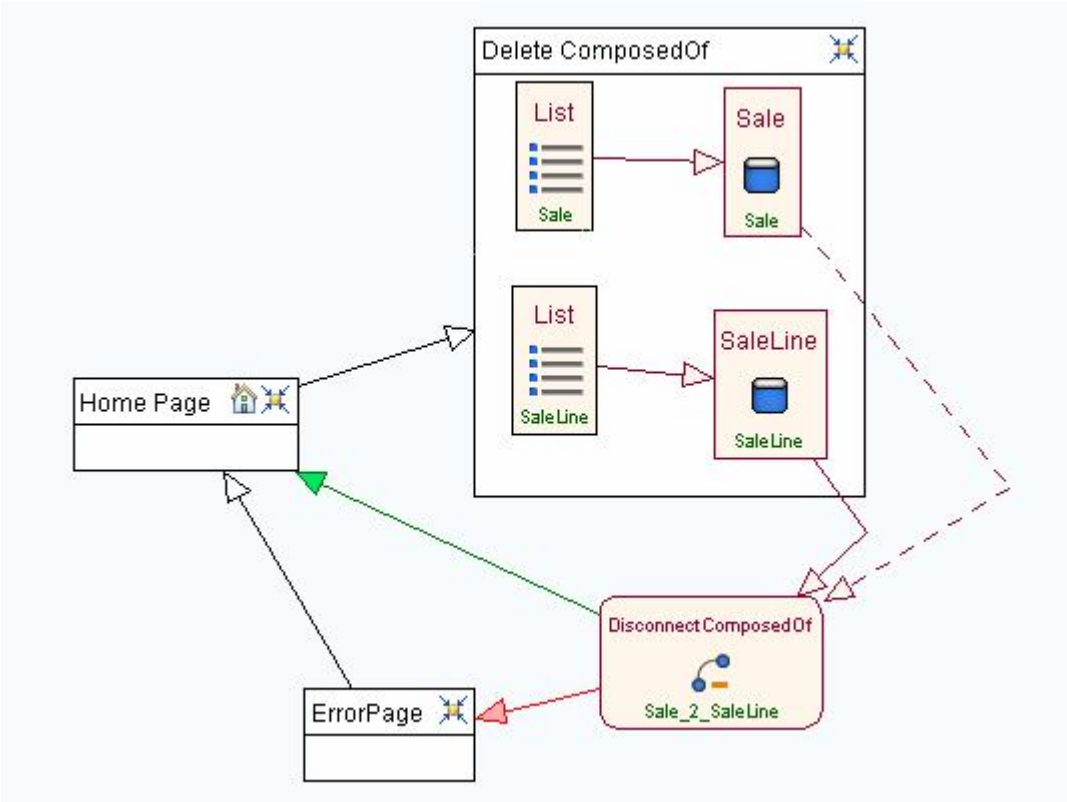


Mètodes implementats:

- *AddInsertRT*: genera una operació de creació d'una instància d'una relació executant les següents subfuncions:
  - *addInsertRTIndexUnit*: genera la pàgina per seleccionar les instàncies de cada entitat que es volen connectar.

- *addInsertRTConnectUnit*: genera l'operació de connexió sobre la base de dades.

Creació d'una operació d'eliminació sobre una relació

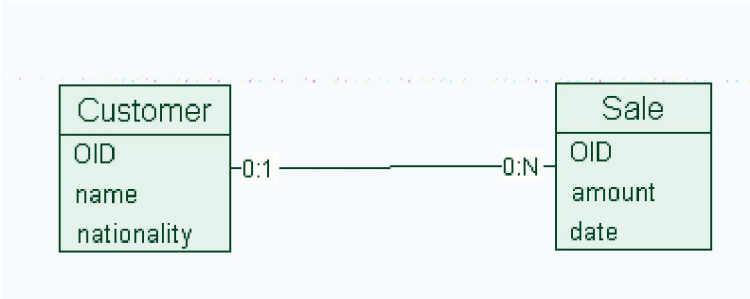


Mètodes implementats:

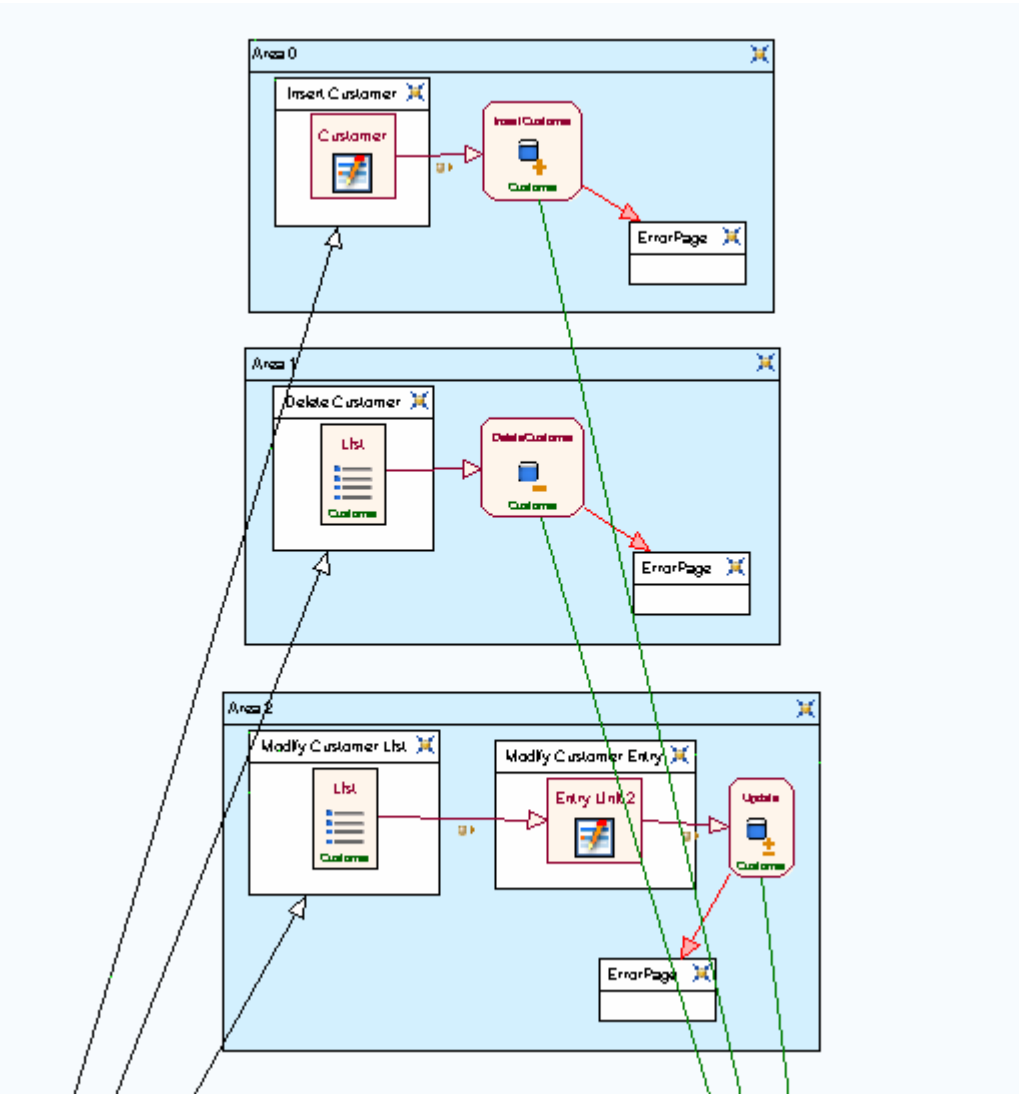
- *AddDeleteRT*: genera una operació d'eliminació d'una instància d'una relació executant les següents subfuncions:
  - *addDeleteRTIndexUnit*: genera la pàgina per seleccionar les instàncies de cada entitat que es volen desconnectar.
  - *addDeleteRTDisconnectUnit*: genera l'operació de desconnexió sobre la base de dades.

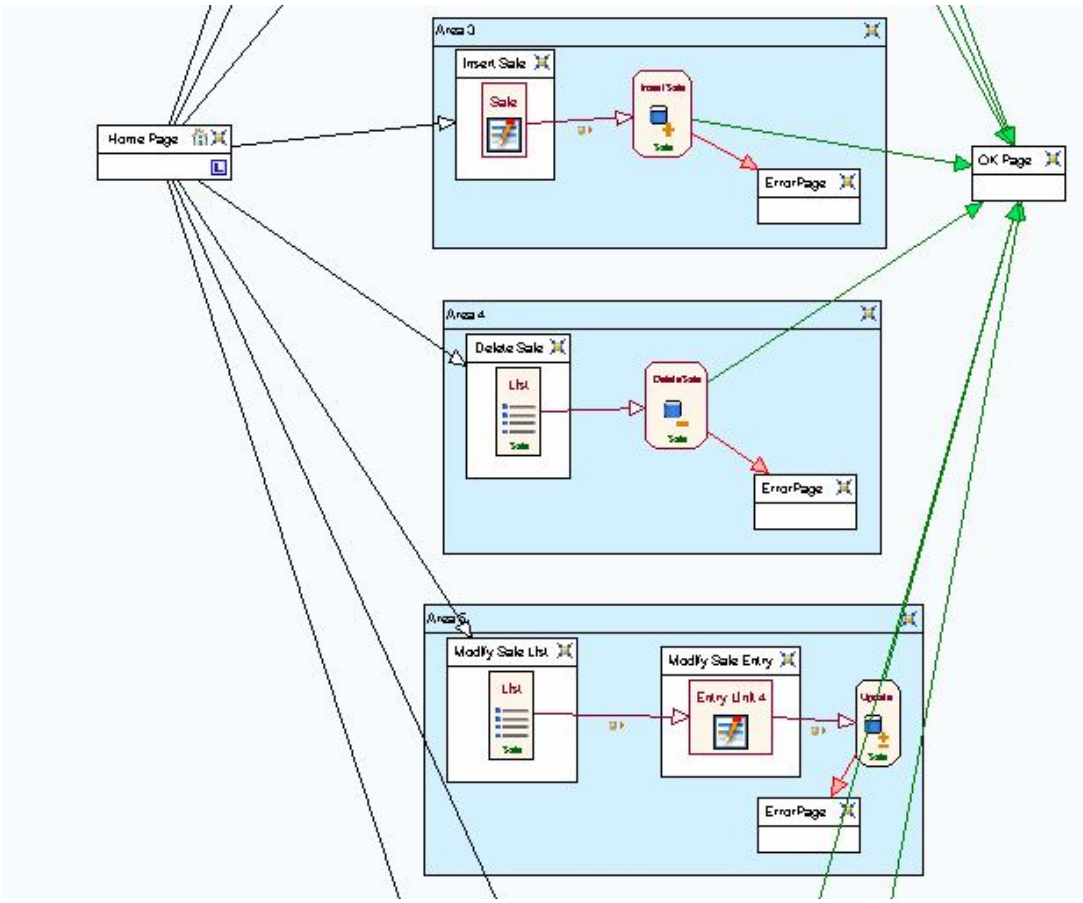
3. Proves

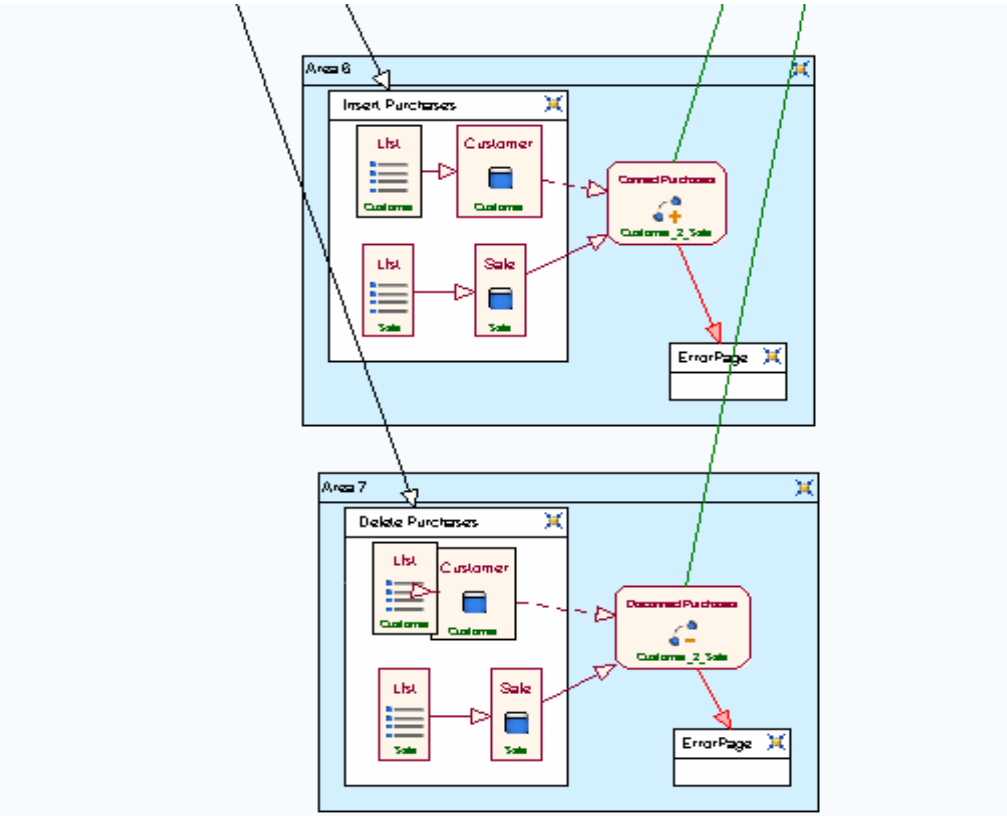
Les proves del PFC s'han realitzat principalment amb el següent model de dades:



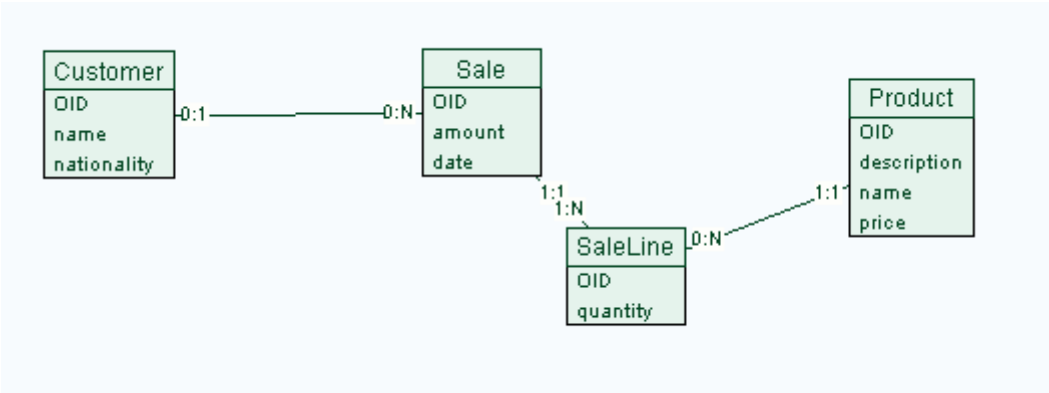
Per executar el programa amb aquest model, cal primerament definir un nou projecte al WebRatio i afegir-hi el model de dades. També és possible crear manualment el model de dades en WebML tot i que, en aquest cas, caldria afegir-lo en un projecte de WebRatio ja que el programa només admet un model de dades inclòs en un projecte de WebRatio. Després de guardar el projecte, s'ha guardat el fitxer SDSProject.xml per produir un model navegacional complet i correcte. En aquest cas, el resultat del programa és el que es mostra a continuació:




Area 3
Insert Sale
Sale
Insert Sale
Sale
ErrorPage
Home Page
OK Page
Area 4
Delete Sale
List
Sale
Delete Sale
Sale
ErrorPage
Area 5
Modify Sale List
List
Sale
Modify Sale Entry
Entry Unit 4
Update
Sale
ErrorPage



L'entrega també inclou un altre projecte d'exemple (fitxer exemple2.xml) que conté un model de dades més complet:



Tanmateix, cal dir que aquest model no s'ha pogut verificar completament amb el WebRatio ja que la versió d'evaluació té un límit de 25 pàgines per projecte i, en aquest cas, el programa generar un model navegacional que supera aquesta límit.

4. Instal·lació del programa

Per executar el projecte cal primerament descomprimir el fitxer mmanriquef_producte.zip una carpeta del disc. Aquesta acció extreu el present document i crea la següent estructura de subcarpetes:

1. *src*: conté tot el codi font del programa.
2. *bin*: aquesta carpeta consta dels fitxers binaris del programa, la llibrería JDOM i el fitxer exemple.xml que conté el projecte explicat anteriorment amb les entitats Customer i Sale.
3. *lib*: llibreries necessàries per l'execució del programa: JDOM (parser de fitxers XML) i el driver JDBC de MySQL (per l'accés a bases de dades MySQL des del WebRatio).
4. *test*: conté el projecte d'exemple (fitxer exemple.xml) que consta d'un model conceptual amb les entitats Customer i Sale. El fitxer SDSProject.xml és el resultat que dóna el programa quan s'executa amb el projecte d'exemple. També hi ha el fitxer exemple2.xml que inclou el model de dades més complet amb les entitats: Customer, Sale, SaleLine, Products.

Abans d'executar el programa, localitzat a la subcarpeta bin, cal cal assegurar primerament que el següent programari està instal·lat:

1. Java Runtime Environment (JRE) versió 1.5 o superior.
2. WebRatio versió 4.3 o superior.
3. Servidor de base de dades MySQL versió 5.0. Les proves del projecte s'han realitzat amb aquesta base de dades tot i que en principi el funcionament hauria de ser correcte en qualssevol altre servidor suportat per WebRatio (SQL Server, MS-Access, etc.).

A continuació es poden seguir el següents passos per provar el programa.

Pas 1: Execució del programa

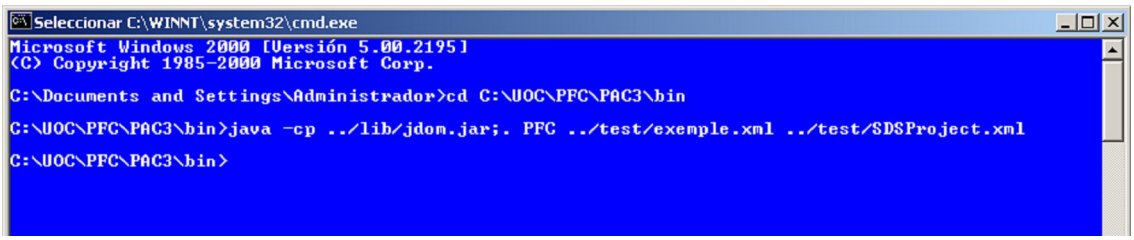
Per executar el programa, obrir una finestra de comandes (programa cmd.exe de Windows) i introduir les següents comandes:

Situar-se en la subcarpeta bin del projecte:

cd <Carpeta on s'ha descomprimit el fitxer .zip>\bin

Per executar el programa amb el projecte d'exemple cal introduir la següent instrucció:

java -cp ../lib/jdom.jar;. PFC ../test/exemple.xml ../test/SDSProject.xml



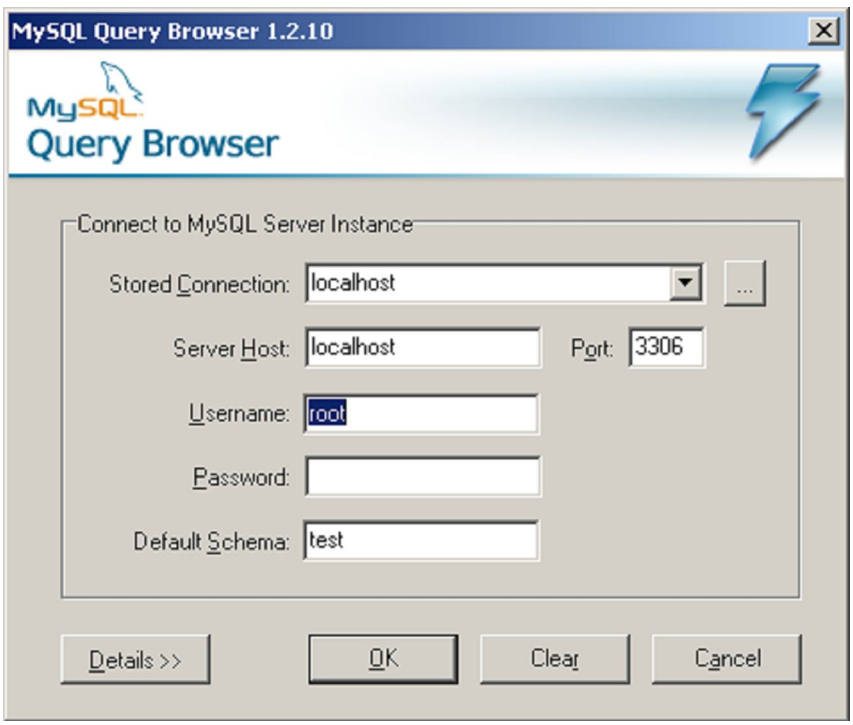
Si el funcionament és correcte, el programa no mostra cap missatge i genera el fitxer SDSProject.xml a la subcarpeta test.xml que tindrà el contingut del fitxer d'entrada més el model navegacional generat.

Nota: El programa no crea un siteview nou per guardar el model navegacional sinó que n'utilitza el primer siteview que troba al fitxer d'entrada. D'aquesta forma no cal recalcular el checksum que WebRatio inclou en qualssevol projecte. Per tant, si el projecte es vol provar amb un altre fitxer d'entrada caldrà que contingui almenys un siteview.

Pas 2: Preparació d'una base de dades MySQL

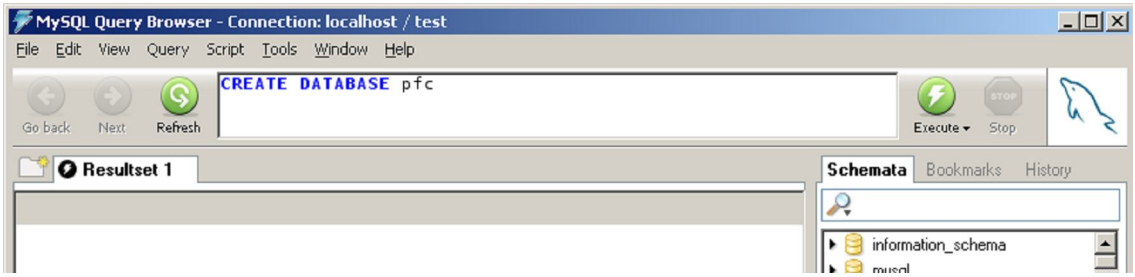
La connexió del WebRatio amb una base de dades de MySQL necessita que primerament que el driver JDBC estigui a la subcarpeta "SDS/lib/ext" de la carpeta d'instal·lació de WebRatio. Per tant, s'ha de copiar el fitxer mysql-connector-java-5.0.5-bin.jar de la subcarpeta "lib" del projecte en aquesta subcarpeta.

Seguidament es crearà una base de dades de MySQL mitjançant l'eina "MySQL Query browser" de MySQL. Aquesta eina demana prèviament els paràmetres de connexió al servidor MySQL que ha d'estar arrancat:



I després d'establir la connexió amb el servidor, introduint la següent comanda de SQL per crear una base de dades buida:

```
CREATE DATABASE pfc
```

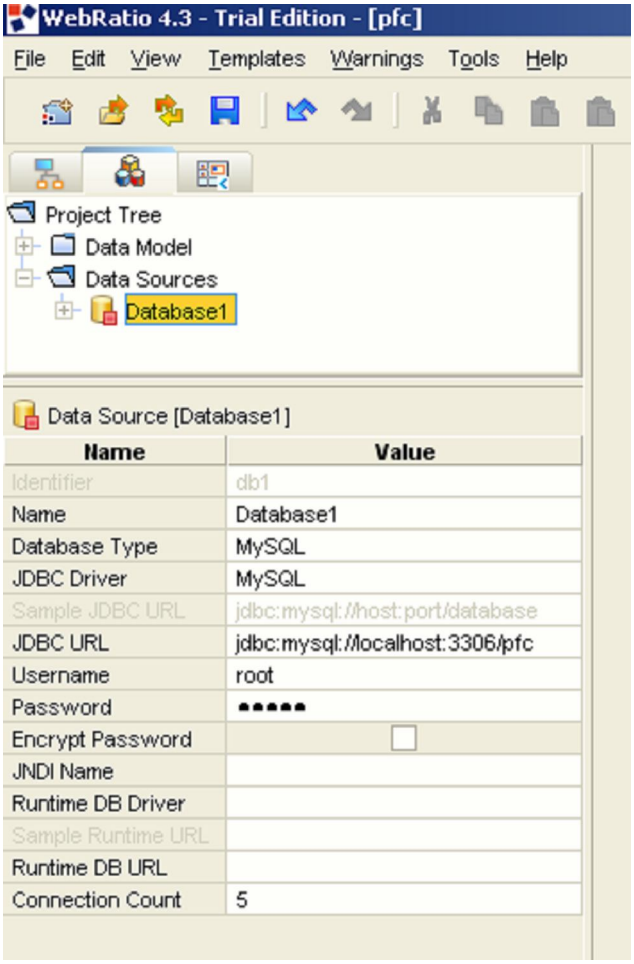


Pas 3: Creació d'un projecte de WebRatio

A continuació, crear un nou projecte de WebRatio seguint aquests passos:
Obrir amb l'Explorador de Windows la carpeta on hi ha instal·lat el WebRatio i crear una carpeta nova (per exemple “pfc”) dintre de la subcarpeta DataRepository.
Copiar el fitxer SDSProject.xml de la carpeta “test” del projecte a la nova carpeta.
Després d'aquests passos, ja podem executar l'eina WebRatio i obrir el nou projecte mitjançant l'opció de menú “File|Open” i seleccionant el projecte que coincideix amb el nom de la nova carpeta. Per veure el model navegacional generat, cal activar la fitxa “Site View 1” situada a la part central del WebRatio.

Pas 4: Establiment d'un enllaç entre el projecte i la base de dades

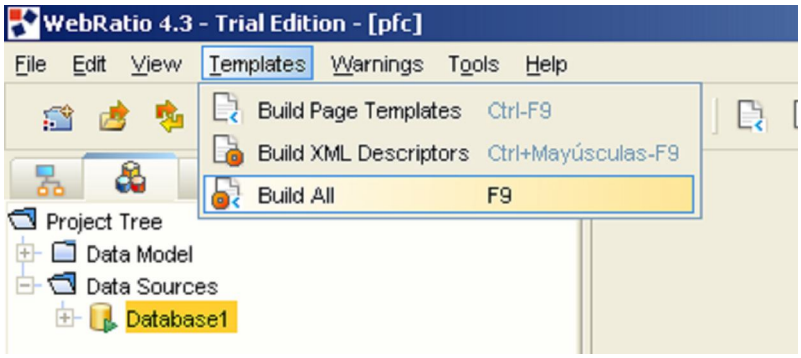
Aquest pas serveix per establir la connexió entre el projecte i la base de dades que s'ha creat anteriorment. Per fer-ho, situar-se a la vista Mapping View del WebRatio i seleccionar el node Project Tree/Data Sources/Database 1 de l'arbre.



Seguidament, modificar la propietat Password indicant la mateixa contrasenya que s'ha introduït a l'eina "MySQL Query Browser". Per comprovar que tots els paràmetres de connexió són correctes, fer clic amb el botó dret del ratolí sobre el node "Database1" de l'arbre i seleccionar l'opció de menú "Refresh". Després, executar l'opció "Create Filled Data Mapping" del mateix menú contextual per crear les taules i carregar-les de informació aleatòria.

Pas 5: Generació del codi de l'aplicació

L'últim pas abans de l'execució de l'aplicació és que el WebRatio generi de forma automàtica el codi de l'aplicació web a partir del model navegacional. El projecte d'exemple té assignat per defecte la carpeta de generació "C:\WebRatio-4.3\tomcat\webapps\pfc" però si el WebRatio s'ha instal·lat en una altra carpeta es pot canviar amb l'opció del menú "Tools|Project Preferences" i modificar la dade "Deploy Path" de la fitxa "Deployment Properties". Per començar la generació de codi, executar l'opció de menú Templates|Build All.

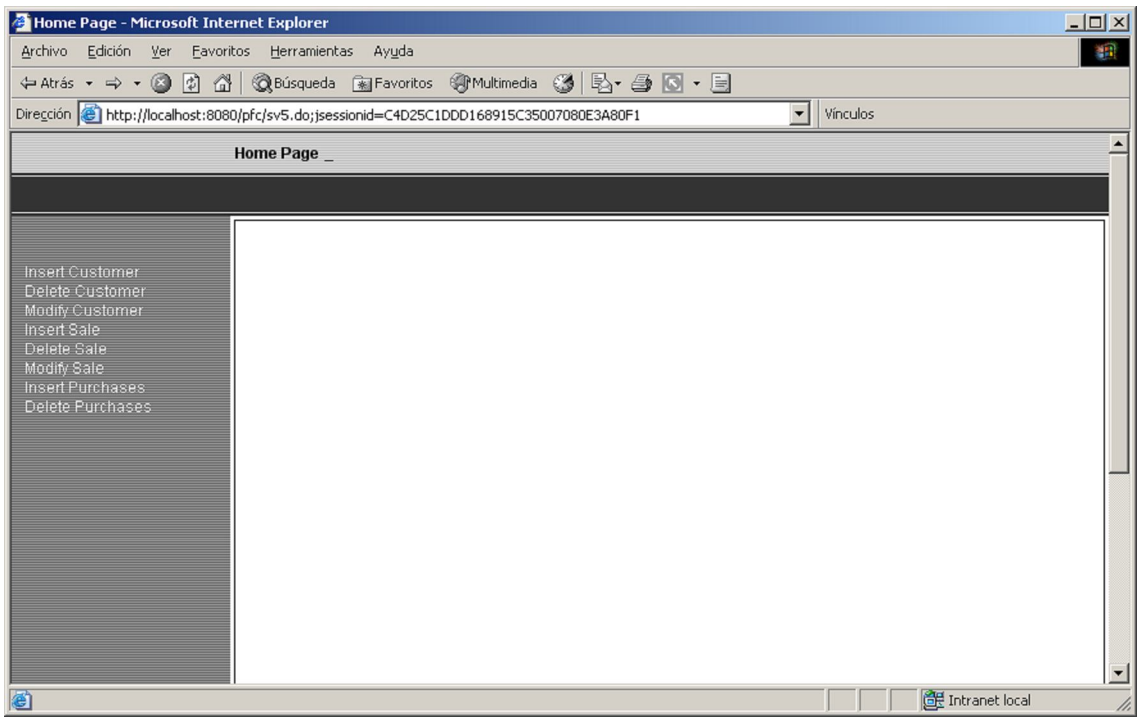


Per últim, el servidor Tomcat necessita establir la connexió amb la base de dades MySQL i, per tant, és necessari copiar novament el driver JDBC (fitxer mysql-connector-java-5.0.5-bin.jar) de la subcarpeta "lib" del projecte a la subcarpeta WEB-INF/lib de l'aplicació (C:\WebRatio-4.3\tomcat\webapps\pfc).

Pas 6: Execució de l'aplicació

Finalment, l'aplicació ja es pot executar a través de qualssevol navegador web. Així doncs, només cal arrancar el servidor de Tomcat subministrat per WebRatio i obrir un navegador web com Internet Explorer o Mozilla Firefox posant la següent adreça URL:

http://localhost:8080/pfc/



Conclusions

La realització del projecte ha estat una bona experiència per varis motius:

- El principal objectiu s'han complert, és a dir, el programa permet generar un model navegacional que compleix les propietats de completesa i correctesa.
- M'ha permès aplicar conceptes de diferents assignatures cursades durants els darrers anys com són programació orientada a objectes, bases de dades, el llenguatge de programació Java.

Tanmateix, també crec que m'ha faltat temps per aplicar millores al programa:

- Aplicar refactorings o optimitzacions sobre el graf generat com per exemple: Head-Merging o Tail-Merging.
- Permetre inserir arcs sobre la mateixa pàgina de forma que, per exemple, un usuari pugui afegir vàries línies d'una mateixa venta sense haver de tornar a la pàgina d'inici.
- Alguns models de dades també suporten les relacions de tipus Generalització tot i que en aquest projecte no s'han contemplat.

Llista de referències

[1] J. Cabot, J. Ceballos, C. Gómez: On the Quality of Navigation Models with Context-Modification Operations. Material proporcionat pel consultor de l'assignatura.

[2] WebML. [Data de consulta: 15/06/2007]
http://www.webml.org

[3] WebRatio. [Data de consulta: 01/03/2007]
http://www.webratio.com