ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE DELL'INFORMAZIONE

PROGETTAZIONE ED IMPLEMENTAZIONE DI UN REDATTORE DI DIAGRAMMI E-R

Relazione finale in
Basi di Dati

Relatore
Prof. Dario Maio

Presentata da
Claudio Cantalupo

Correlatore
Dott.ssa Annalisa Franco

SESSIONE II
ANNO ACCADEMICO 2012/2013

SOMMARIO

Capitolo 1
INTRODUZIONE

Negli ultimi anni i sistemi informatici hanno subito una rapida accelerazione, legata soprattutto alla presenza sempre più forte della tecnologia nelle nostre vite e nella nostra società.

Con l'affermarsi dell'era di Internet, prima su PC e successivamente sui dispositivi portatili, stiamo tuttora assistendo ad un fenomeno globale la cui crescita pare inarrestabile e il cui futuro, non lontano, vedrà il nostro stesso corpo trasformato in un terminale di quello che oggi conosciamo come *Web*.

In un settore dove tutto è in costante e repentino cambiamento, il successo ottenuto dal modello relazionale e i suoi derivati non ha probabilmente eguali nella storia dell'informatica. Tale modello, mutuato, nei primi anni '70, dalla teoria matematica degli insiemi, è stato concepito da Edgar F. Codd per risolvere il problema di accedere e condividere enormi quantità di dati nel sistema bancario; il successo è stato immediato e i modelli fino ad allora utilizzati, gerarchico e reticolare, sono stati rimpiazzati, nella quasi totalità dei casi, nel giro di pochi anni. La visione di Codd, sin dai primi studi, è apparsa subito chiara.

> *The term relation is used here in its accepted mathematical sense […]. Users should not normally be burdened with remembering the domain ordering of any relation […]. Accordingly, we propose that users deal, not with relation which are domain-ordered, but with relationships which are their domain unordered counterparts. To accomplish this, domains must be uniquely identifiable at least within any given relation, without using position. [Cod70]*

Pochi anni dopo l'adozione del modello relazionale per la rappresentazione e l'accesso ai dati, un'ulteriore forte spinta è arrivata con la presentazione, avvenuta nel 1976 da parte di Peter Chen, del modello Entità-Relazione; tale modello è una versione migliorata di quello puramente relazionale ed è stato ottenuto tenendo in considerazione anche gli aspetti positivi di altri modelli quali il reticolare e l'entity-set.

> *The logical view of data has been an important issue in recent years. Three major data models have been proposed: the network model, the relational model and the entity set model. These models have their own strengths and weaknesses […]. This paper presents the entity-relationship model, which has most of the advantages of the above three models. The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. [Che76]*

Lo stesso Codd, inizialmente scettico sulla bontà del modello E-R, anni dopo ne ammise le evidenti qualità [Che02].
Parte del successo del modello E-R è sicuramente dovuta alla simbologia grafica che venne presentata, dallo stesso Chen, contestualmente al modello.

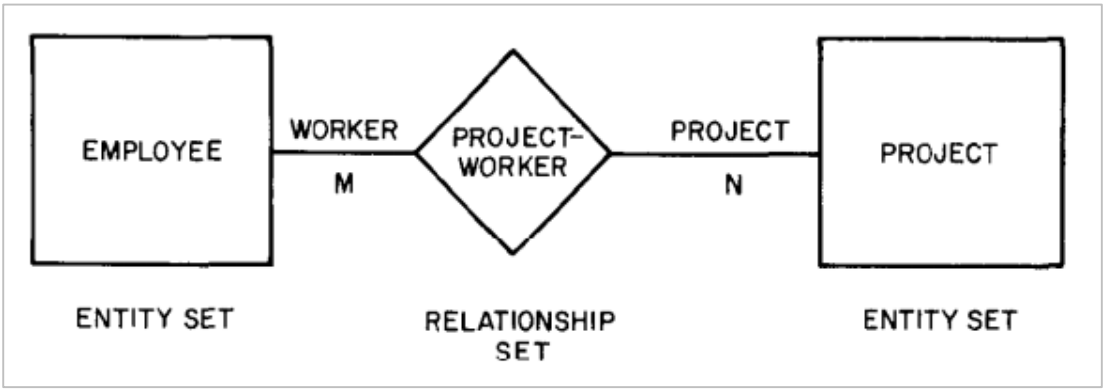


Figura 1.1: *primo diagramma E-R pubblicato da Chen*

La simbologia indicata in figura 1.1, a circa 40 anni di distanza, è ancora alla base di tutti gli strumenti di progettazione di database; nel corso del tempo è stata lievemente migliorata, spesso si fa riferimento alla versione modificata come *notazione estesa*, presentata per la prima volta a metà degli anni '80 [Tyf86] e, successivamente, ulteriormente rivista e ottimizzata.

Considerando la progettazione concettuale pura, la notazione estesa è, probabilmente, la migliore che sia mai stata concepita per tale scopo.

In ambito accademico tale notazione è ancora largamente usata ma in ambiti più pratici è stata rimpiazzata da altre notazioni come *IDEF1X* [Bro93], *Bachman* [Bac69]*, Information Engineering* [Hal01] *e Barker* [1] [Bar90] in quanto più vicine alla struttura fisica del database finale, quindi più adeguate alla progettazione strutturale. Tali notazioni non introducono alcun nuovo concetto, semplicemente utilizzano delle simbologie grafiche che richiedono meno spazio e che facilitano lo sviluppo dei redattori di diagrammi, così come la realizzazione del diagramma stesso da parte dell'utente finale.

Tale semplificazione ha però un costo poiché esiste una differenza fondamentale tra l'approccio illustrato originariamente da Chen e quello che si è affermato negli anni a venire nella quasi totalità degli strumenti di progettazione di database. Nel modello originario l'entità e la relazione sono due elementi ben distinti a livello grafico, ognuno con il proprio gruppo di attributi. In tal modo anche gli identificatori sono di più facile comprensione poiché vengono resi evidenti, a livello visuale, in maniera decisamente più naturale rispetto ad una stringa inserita all'interno di una tabella. Ciò facilita una migliore progettazione concettuale perché graficamente rispecchia meglio il dominio reale in cui quelle entità "vivono" e sviluppano le loro relazioni.

[1] Le notazioni *Barker* e *Information Engineering* utilizzano a loro volta un'altra celebre notazione, conosciuta oggi come *crow's foot* (piede di corvo) ma inizialmente chiamata c*hicken feet* (piedi di pollo) oppure *fork* (forchetta) [Eve87].

L’applicativo realizzato per questa tesi, SharpER Diagram Editor (SER-DE da ora in poi), ha proprio l’obiettivo di reintrodurre, almeno in ambito accademico, uno strumento in grado di realizzare in maniera completa i diagrammi E-R così come mostrati nella teoria. Questa necessità è stata resa evidente dalla totale mancanza di strumenti efficaci e dedicati a questo specifico scopo. Prima della realizzazione di SER-DE era possibile realizzare un diagramma completo in notazione estesa con il solo ausilio di strumenti di disegno generici, che nulla hanno a che vedere con il modello relazionale e che rendono oltremodo complessa e foriera di errori la redazione di un diagramma Entità-Relazione.

Nei capitoli successivi analizzeremo per prima cosa quali sono gli strumenti ad oggi esistenti, quali sono le loro funzionalità e quali i limiti. Quindi vedremo quali sono stati i requisiti che hanno portato allo sviluppo di SER-DE e quali le funzionalità di conseguenza implementate.
Largo spazio verrà dato alla sezione dedicata ai dettagli implementativi, con diagrammi UML delle classi e spiegazioni dettagliate sui pattern architetturali scelti e relative motivazioni.
Infine verrà presentata una panoramica sui risultati ottenuti con SER-DE e verranno analizzati gli sviluppi futuri possibili di questo strumento.

Capitolo 2
STATO DELL'ARTE

Nel corso degli anni sono stati sviluppati una pletora di applicativi CASE[2] in grado di aiutare il progettista nella creazione di diagrammi E-R. Molti di questi, i più avanzati e ricchi di funzionalità, sono disponibili esclusivamente sotto licenza commerciale (a pagamento); tra questi ricordiamo, in particolare, ER/Studio [ERS], ERwin [ERW], Toad Data Modeler [TDM] e Sybase Power Designer [SPD].
Tali applicativi offrono funzionalità estremamente avanzate, utili sia in caso di analisi e reverse engineering di database già esistenti, così come per la creazione di nuovi (metodologie d'approccio top-down e bottom-up); alcune di queste funzionalità comprendono:

- supporto nativo a tutti i principali DBMS;
- documentazione automatica di sistemi già esistenti;
- migrazione da un DBMS ad un altro;
- modellazione logica e fisica;
- automatizzazione degli script di creazione e modifica;
- versioning.

Con questi strumenti è possibile seguire l'intero ciclo di vita di un database, dalla progettazione, alla creazione delle tabelle, così come il controllo dei dati in produzione e le eventuali modifiche strutturali inesorabilmente necessarie nel tempo.
Esistono fortunatamente anche altri ottimi applicativi, gratuiti e talvolta open source; in particolare nei prossimi paragrafi verranno analizzati tre di questi:

- DB-Main;

[2] Computer Aided Software Engineering

- MySQL Workbench;
- Oracle Data Modeler.

Non verranno analizzati ulteriormente strumenti commerciali, sia per la mancanza delle licenze necessarie al loro utilizzo, sia perché in quelli gratuiti troviamo tutte le caratteristiche dei sistemi a pagamento, anche se talvolta in versione semplificata o limitata; cosa ancora più importante per il nostro scopo è che tutti gli strumenti disponibili, gratuiti e non, supportano le stesse tipologie di notazioni grafiche ma nessuno supporta in maniera completa quella originaria definita da Chen né quella estesa, supportata invece da SER-DE.

2.1 DB-Main

Questo applicativo [DBM], sviluppato presso l'*Università di Namur*, consente di creare differenti tipologie di diagrammi, in particolare quelli E-R e alcuni di quelli previsti dal linguaggio UML ossia quello delle classi, delle attività e dei casi d'uso. È bene ricordare che talvolta proprio il diagramma delle classi viene utilizzato in sostituzione dello stesso diagramma E-R durante la progettazione di un sistema, il che rappresenta una buona scelta in caso di analisi orientata agli oggetti.
La notazione adottata in DB-Main è quella che più si avvicina a quella estesa tra tutti gli strumenti analizzati.

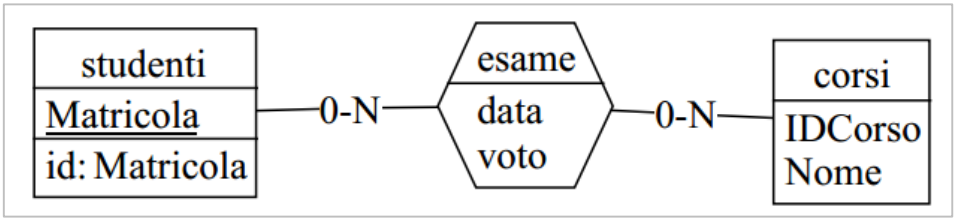


Figura 2.1: *porzione di diagramma.*

In figura 2.1 è rappresentata una porzione di diagramma E-R tramite cui è possibile illustrare alcune peculiarità della simbologia utilizzata. Notiamo ad

esempio come le entità siano rappresentate da rettangoli e le relazioni da esagoni, molto simili ai rombi della notazione di Chen. Per i vincoli di cardinalità è stato scelto l'utilizzo della simbologia (min-card..max-card) in grado d'illustrare in una sola etichetta di testo sia la partecipazione che la cardinalità. Per questioni di semplificazione grafica gli attributi non rispettano la notazione di Chen né quella estesa ma sono stati inseriti direttamente all'interno dell'entità o della relazione corrispondente; nel caso in cui l'attributo sia anche un identificatore questo viene semplicemente sottolineato.

D'altra parte, come già accennato, la simbologia introdotta da Chen per rappresentare gli attributi ha purtroppo trovato poco successo nei redattori di diagrammi; anche quella estesa, nonostante sia ottimizzata rispetto a quella originaria, ha avuto la stessa sorte. Questo per diversi motivi, in particolare perché la rappresentazione degli attributi in questo modo può richiedere molto spazio all'interno del canvas di disegno. Tuttavia, con la simbologia adottata da DB-Main, non è possibile rappresentare con la stessa chiarezza e la stessa immediatezza un diagramma E-R, soprattutto nel caso sia necessario ricorrere a identificatori esterni o misti.

Infine, anche per le gerarchie è stato scelto un simbolo diverso da quello utilizzato nella notazione estesa, nello specifico viene utilizzato un triangolo.

Pro:

- Adotta una notazione e un approccio simili a quelli della notazione estesa.
- Favorisce una progettazione concettuale avanzata.
- Supporta la fase di progettazione logica che può essere effettuata anche in modo automatico.
- Supporta la creazione di DDL[3] per i più importanti DBMS.
- Facilita la creazione di documentazione e report.

[3] Data Definition Language

Contro:

- Interfaccia utente piuttosto complessa e relativa curva d'apprendimento alta.
- Reverse engineering di sistemi esistenti molto limitato.
- Bassa qualità grafica e di personalizzazione dei diagrammi.
- Licenza non open source.
- Funzionalità di stampa molto limitata.
- Disponibile solo per sistemi operativi Windows.

2.2 MySQL Workbench

Altro ottimo applicativo [MSW], ben progettato e ricco di funzionalità, è stato realizzato inizialmente da *Sun Microsystems* e oggi è mantenuto e migliorato da una ricca community open source supportata da *Oracle*; è disponibile anche una versione a pagamento, con la quale viene fornita assistenza.

Nell'analisi effettuata sui vari strumenti CASE, MySQL Workbench è risultato essere senza dubbio quello con il più ampio supporto alle differenti tipologie di notazioni grafiche; tra queste troviamo, in particolare:

- IDEF1X;
- Information Engineering;
- classica (una rivisitazione della notazione estesa ma comunque ben differente da quest'ultima);
- UML (diagramma delle classi).

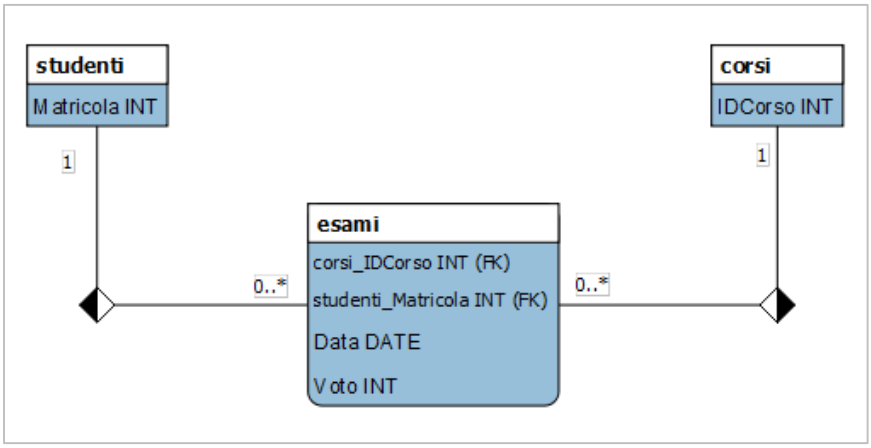


Figura 2.2: *porzione di diagramma con notazione classica.*

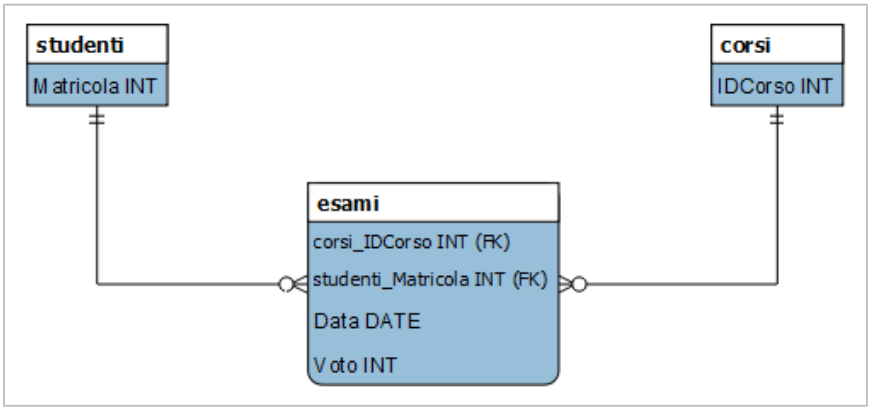


Figura 2.3: *porzione di diagramma con notazione Information Engineering.*

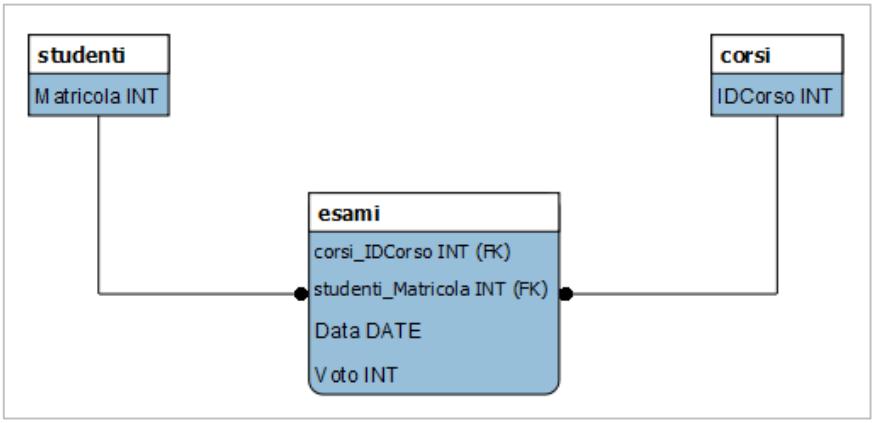


Figura 2.4: *porzione di diagramma con notazione IDEF1X.*

Come si nota nelle figure 2.2, 2.3 e 2.4 le differenti notazioni grafiche sono equivalenti in questo semplice caso; tuttavia in nessuna di queste è

possibile indicare dei vincoli di cardinalità specifici, ossia diversi dai canonici (0..1), (0..n), (1..1), (1..n). Ad esempio la relazione che intercorre tra figlio e genitori, esprimibile in notazione estesa con (2..2), non è rappresentabile graficamente in MySQL Workbench, ma potrà solo essere riportata nella documentazione accessoria.

Pro:

- Interfaccia molto intuitiva e personalizzabile.
- Forte accoppiamento e sincronizzazione con database reali.
- Progetto e community open source molto attivi.
- Gestione avanzata di stampa ed esportazione.
- Tante utility già sviluppate e disponibilità di una shell per eseguire comandi avanzati.
- Documentazione dell'applicativo completa e aggiornata.
- Disponibile per sistemi operativi Windows, Mac OS, Linux.

Contro:

- Forward/Reverse engineering disponibile solo verso/da il DBMS MySQL.
- Crash sporadici dell'applicativo utilizzandolo intensivamente.
- Progettazione concettuale pura non possibile.

2.3 Oracle Data Modeler

Recentemente entrato nel panorama dei redattori di diagrammi E-R, questo applicativo [ODM], sviluppato da *Oracle*, rappresenta un'interessante alternativa.
Supporta tre tipologie di notazioni e tra le varie alternative gratuite disponibili è quella che mette a disposizione le caratteristiche più avanzate, in particolare se si è scelto l'utilizzo del DBMS Oracle; tuttavia questo non rappresenta un grosso limite poiché non è l'unico ufficialmente supportato.

Per quanto concerne le notazioni utilizzabili abbiamo a disposizione:

- Bachman;
- Information Engineering;
- Barker.

Caratteristica peculiare di Oracle Data Modeler è la possibilità di passare agevolmente da una visione logica ad una relazionale, ossia più vicina alla struttura fisica del database, e viceversa.

Figura 2.5: *porzione di diagramma in modalità logica.*

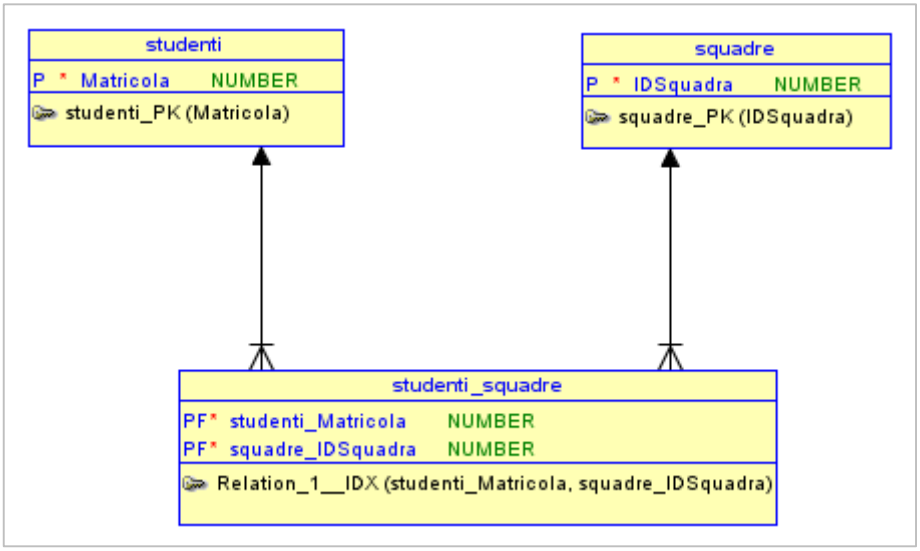


Figura 2.6: *porzione di diagramma in modalità relazionale.*

L'applicativo offre la possibilità di passare dalla modalità logica di figura 2.5, alla modalità relazionale di figura 2.6 e viceversa; questo senza perdere eventuali peculiarità esprimibili solo in una delle due modalità come, ad esempio, il tipo di dato che è possibile specificare solo nel diagramma relazionale.

Pro:

- Supporta la creazione di DDL per i più importanti DBMS.
- Possibilità di creare regole personalizzate per la denominazione automatica di campi e tabelle.
- Possibilità di aggiungere script personalizzati a quelli creati automaticamente a partire dal diagramma.
- Gestione avanzata del versioning.
- Interfaccia molto intuitiva e personalizzabile.
- Gestione avanzata di stampa ed esportazione.
- Individuazione automatica di errori concettuali, ad esempio la mancanza d'identificatori se necessari.
- Documentazione dell'applicativo completa e aggiornata.
- Importazione/Esportazione da/a applicativi di terze parti.
- Disponibile per sistemi operativi Windows, Mac OS, Linux.

Contro:

- Licenza non open source.
- Reverse engineering disponibile solo per il DBMS Oracle.
- Progettazione concettuale pura non possibile.

2.4 Comparativa funzionalità

Anzitutto è bene ricordare che tutti gli strumenti commerciali elencati in precedenza sono preferibili in caso di utilizzo intensivo di funzionalità avanzate quali il reverse engineering spinto o le migrazioni di DBMS.
Tra gli strumenti gratuiti analizzati quello più completo è risultato essere senza dubbio Oracle Data Modeler; questo consente una buona analisi e un'ottima modellazione del database, nella quale è possibile specificare peculiarità molto intime del DBMS stesso (in particolare, ovviamente, rimanendo nell'ecosistema offerto da *Oracle*). Inoltre, la gestione integrata

del versioning tramite SVN[4] lo rende adeguato per un utilizzo professionale da parte di un gruppo di sviluppatori che hanno necessità di collaborare sullo stesso progetto.

Al fine di collocare meglio gli applicativi analizzati, di seguito troviamo una breve analisi comparativa suddivisa per funzionalità:

- Interfaccia utente: MySQL Workbench è senza dubbio lo strumento che offre l'interfaccia più fluida e personalizzabile.
  Oracle Data Modeler non fornisce la stessa possibilità di personalizzazione, offre comunque un'interfaccia comoda e di facile apprendimento.
  L'interfaccia di DB-Main non brilla per chiarezza, rapidità e facilità d'uso e questo è purtroppo un grosso limite al suo utilizzo.
- Progettazione concettuale: come già accennato DB-Main è l'applicativo che più convince per una progettazione concettuale pura, questo grazie all'utilizzo di una notazione e di una simbologia non utilizzati da altri redattori.
  Oracle Data Modeler e MySQL Workbench offrono caratteristiche simili, con un punto a favore del software di *Oracle*, la cui funzionalità di progettazione logica risulta essere molto comoda.
- Progettazione fisica: Oracle Data Modeler è uno strumento molto valido, il quale consente una modellazione ricca di dettagli avanzati, decisamente più completa rispetto agli altri applicativi analizzati.
  MySQL Workbench offre un buon numero di funzionalità ma purtroppo valide solo per il DBMS MySQL.
  DB-Main invece non ha limiti di applicazione ad un solo DBMS, tuttavia le opzioni offerte sono basilari se paragonate a quelle degli altri applicativi.
- Disponibilità notazioni: MySQL Workbench è il redattore con il supporto più esteso alla pletora di differenti notazioni esistenti.

[4] Apache Subversion

Oracle Data Modeler consente l'utilizzo delle sole notazioni ad oggi più utilizzate mentre DB-Main risulta essere il software meno duttile da questo punto di vista.

- Reverse engineering: Oracle Data Modeler consente un'analisi avanzata di database già esistenti; ciò è possibile in differenti modalità, ad esempio tramite collegamento diretto al database via JDBC[5], oppure importando un file contenente la sintassi DDL della struttura del sistema da analizzare.

  MySQL Workbench è paragonabile a Oracle Data Modeler come caratteristiche ma, come già scritto, valide solo per MySQL.

  DB-Main offre un reverse engineering basilare, funzionante però su diversi DBMS.
- Forward engineering: ancora una volta lo strumento che si è rivelato più completo è Oracle Data Modeler poiché consente una creazione molto precisa di DDL relativi ai principali DBMS presenti sul mercato.

  MySQL Workbench è in grado di creare dei DDL completi ma, al solito, per il solo DBMS omonimo.

  DB-Main consente la creazione di DDL per svariati DBMS, senza spingersi tuttavia in dettagli troppo specifici.
- Creazione documentazione: DB-Main è in grado di creare automaticamente una documentazione ben strutturata, inoltre, la possibilità di definire alcuni importanti diagrammi del linguaggio UML, consente di avere una documentazione completa dei flussi operativi e dei casi d'uso.

  Oracle Data Modeler non è in grado di creare automaticamente alcun tipo di documentazione ma è possibile definire i flussi di lavoro, gli scenari d'uso e i ruoli degli attori coinvolti, facendo uso di funzionalità e diagrammi proprietari.

[5] Java Database Connectivity

MySQL Workbench non prevede alcun tipo di documentazione strutturata; è possibile solo inserire dei commenti a corredo di tabelle e relazioni.

- Importazione/Esportazione diagrammi: Oracle Data Modeler consente di lavorare con parecchi formati, sia di applicativi sviluppati da *Oracle,* sia di terze parti.

  DB-Main lavora agevolmente con file XML e questo, in futuro, sarà l'unico formato supportato. Al momento supporta anche il formato testuale ISL[6].

  MySQL Workbench è l'unico tra gli applicativi analizzati in grado di lavorare solo con un formato proprietario, oltretutto binario, il che lo rende davvero poco versatile da questo punto vista.
- Stampa: Oracle Data Modeler e MySQL Workbench offrono funzionalità di stampa avanzate, ed è possibile effettuare le varie configurazioni utilizzando una comoda anteprima; sono supportate anche modalità virtuali di stampa, come quella su PDF, su PostScript o in vari formati immagine.

  DB-Main consente una semplice stampa dell'intero diagramma rappresentato sul canvas con la sola possibilità di definire alcune caratteristiche basilari come orientamento e dimensione del foglio.

[6] Information System Specification Language

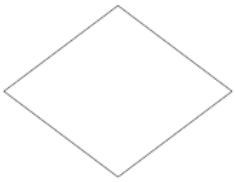
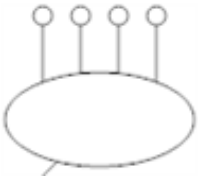
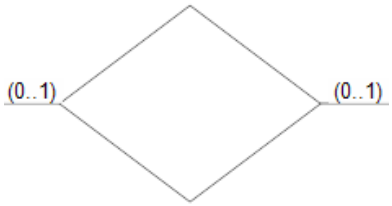
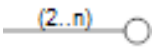
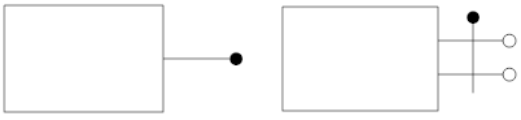
Capitolo 3
REQUISITI E FUNZIONALITÀ

Nel capitolo 2 sono stati analizzati alcuni dei più importanti redattori di diagrammi E-R ad oggi presenti sul mercato. È evidente come nessuno degli applicativi presi in esame rispecchi fedelmente la visione di Chen, che è alla base della sua celebre notazione. La questione non è semplicemente di simbologia ma è quasi filosofica; la cultura orientale, cui Chen appartiene, lo ha portato a modellare gli oggetti del mondo reale prima sotto rigorosa forma matematica e, successivamente, sotto forma di diagramma. È lui stesso a ricordare come il suo lavoro sia stato influenzato dalla cultura cinese, nella quale i caratteri e la loro composizione, altro non sono che una modellazione grafica degli oggetti del mondo reale [Che02].
L'obiettivo di SER-DE, nonché requisito fondamentale, è quello di facilitare la realizzazione di diagrammi che rispecchiano tale filosofia.
A tal fine sono state prese in esame non solo la più celebre notazione di Chen ma anche e soprattutto le notazioni derivate, le quali, nel corso degli anni, sono state sviluppate, migliorate ed accettate in ambito accademico. È molto importante ricordare che tali notazioni hanno introdotto solo delle piccole ottimizzazioni grafiche e non snaturano la visione originaria.

3.1 Simbologia adottata

Nella tabella 3.1 sono indicati tutti i costrutti necessari alla realizzazione di un diagramma E-R e la relativa rappresentazione grafica che è stata scelta ed implementata all'interno di SER-DE. Per chiarezza non sono state inserite in tabella le etichette di testo associate ai relativi oggetti, ma è

comunque possibile farlo, utilizzando SER-DE, per entità, relazioni ed attributi (semplici e composti).

Costrutto	Rappresentazione Grafica
Entità	
Relazione	
Attributo semplice	
Attributo composto	
Cardinalità	
Cardinalità di un attributo	
Identificatore interno	

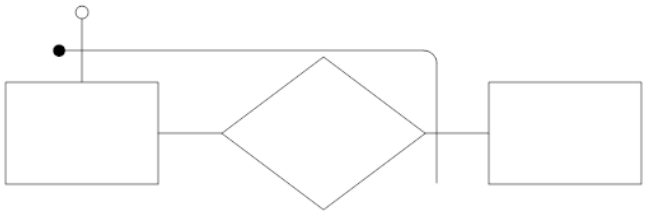
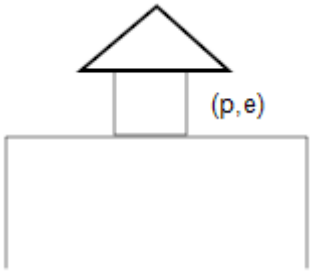
Identificatore esterno[7]	
Generalizzazione	
Subset	

Tabella 3.1: *costrutti di base e loro rappresentazione.*

Gli oggetti grafici illustrati in tabella 3.1 possono essere collegati tra loro utilizzando l'oggetto connessione, una semplice linea il cui significato, le opzioni e i rispettivi vincoli variano in funzione del tipo di collegamento; ad esempio una linea di connessione tra entità e relazione è profondamente diversa da una linea di connessione tra entità e generalizzazione o entità e subset, seppur graficamente appaiano identiche.

[7] È possibile rappresentare in SER-DE qualunque tipologia d'identificatore interno o esterno, compresi quelli misti.

3.2 Funzionalità implementate

3.2.1 Interfaccia utente

Come mostrato nell'esempio di figura 3.1 la caratteristica base dell'interfaccia applicativa consiste nella gestione a schede dei diagrammi sui quali si sta lavorando.

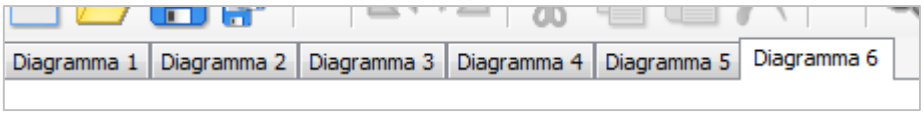


Figura 3.1: *schede di diagrammi aperti contemporaneamente.*

Tale modalità di lavoro è stata scelta per la fluidità e la praticità che mette a disposizione; in casi reali è frequente suddividere il lavoro in blocchi, soprattutto quando è necessario modellare database molto complessi. Per questo è preferibile realizzare un nuovo diagramma per ogni blocco necessario e l'interfaccia a schede è, senza alcun dubbio, la scelta migliore in tale prospettiva.

Altra caratteristica importante dell'applicativo realizzato, così come di ogni applicativo, è il modo in cui si rendono accessibili le varie opzioni e le funzionalità all'utente finale; in questo caso è stato scelto un approccio multiplo, pertanto è possibile accedere ad alcune opzioni ricorrenti in differenti sezioni e menu del programma. Gran parte delle opzioni offerte da SER-DE si trovano nel menu principale (figura 3.2).

Figura 3.2: *menu principale.*

La barra degli strumenti (figura 3.3) contiene le funzioni più comuni.

Figura 3.3: *barra degli strumenti.*

Per facilitare e velocizzare le operazioni di disegno l'utente ha a disposizione un pannello mobile (figura 3.4) contenente gli oggetti e gli strumenti di disegno (selezione, entità, relazione, connessione, generalizzazione o subset); in questo modo si ha sempre a portata di mano, in qualunque punto dell'area di disegno, la possibilità di cambiare velocemente oggetto.

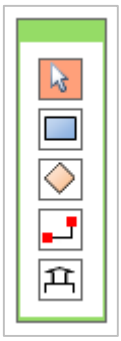


Figura 3.4: *pannello mobile di disegno.*

Ultima caratteristica rilevante dell'interfaccia utente è la barra di stato (figura 3.5), posizionata nella parte inferiore della finestra dell'applicativo.

Figura 3.5: *barra di stato.*

Tale sezione dell'applicativo contiene informazioni dinamiche molto utili; queste possono cambiare, infatti, in funzione dell'oggetto selezionato o dell'operazione corrente. Inoltre qui vengono indicate la dimensione del canvas di disegno e la posizione del cursore del mouse all'interno di tale canvas, oltre al numero di oggetti totali e quelli eventualmente selezionati. In alcuni scenari particolari, come quello di memoria di sistema in esaurimento, la barra di stato viene anche utilizzata per notificare degli avvertimenti all'utente.

È possibile nascondere la barra degli strumenti, il pannello di disegno e la barra di stato, ognuno di questi in maniera indipendente dagli altri.
Ultima cosa importante da tenere in considerazione circa l'interfaccia utente è che, al momento, questa è disponibile solo in lingua inglese, così come il manuale utente allegato all'applicativo.

3.2.2 Funzionalità specifiche

Gran parte delle caratteristiche illustrate nel paragrafo 3.2.1 sono comuni a qualunque editor grafico avanzato; le funzionalità specifiche illustrate in questo paragrafo, invece, sono strettamente legate ai diagrammi relazionali o comunque agli oggetti che è possibile disegnare con SER-DE. Anzitutto, per ogni oggetto disegnabile, sono disponibili alcune funzionalità di base (figura 3.6); a queste si accede premendo il tasto destro del mouse, una volta selezionato[8] l'oggetto.

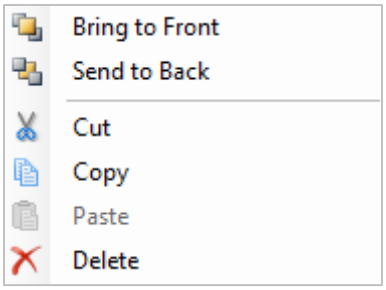


Figura 3.6: *funzionalità comuni degli oggetti disegnabili.*

L'opzione *bring to front* (porta in primo piano), indicata in figura 3.6, consente di visualizzare uno specifico oggetto sopra ad altri nel caso in cui questi si sovrappongano. In maniera opposta, la funzionalità *send to back* (manda in secondo piano), nasconde le porzioni di uno specifico oggetto che si sovrappongono con altri. Scegliendo l'azione *cut* (taglia) gli oggetti selezionati vengono eliminati dal canvas ma copiati in memoria; a questo

[8] Un oggetto risulta selezionato se il suo contorno è rosso; nel caso il contorno sia verde l'oggetto è solo stato evidenziato, poiché è posizionato in prossimità del cursore del mouse. Per selezionare un oggetto è necessario premere il tasto sinistro del mouse nel momento in cui l'oggetto risulta evidenziato oppure effettuare una selezione multipla premendo il tasto sinistro del mouse e muovendo contemporaneamente lo stesso nell'area d'interesse.

punto, premendo *paste* (incolla), ciò che è in memoria viene incollato sul canvas corrente. L'azione *copy* (copia) opera allo stesso modo di *cut*, ma senza eliminare gli oggetti selezionati. Infine, scegliendo *delete* (cancella), gli oggetti selezionati vengono eliminati senza essere copiati in memoria.
Sempre con la pressione del tasto destro si può accedere alle proprietà esclusive di entità, relazioni e generalizzazioni o subset. Le proprietà di un'entità sono relative a nome e stile dell'oggetto (figura 3.7) e agli eventuali attributi semplici/composti, sia dal punto di vista logico che grafico (figura 3.8 e figura 3.9); come indicato in figura, è possibile specificare se un attributo è anche un identificatore così come la posizione che tale attributo dovrà avere all'interno del canvas di disegno.

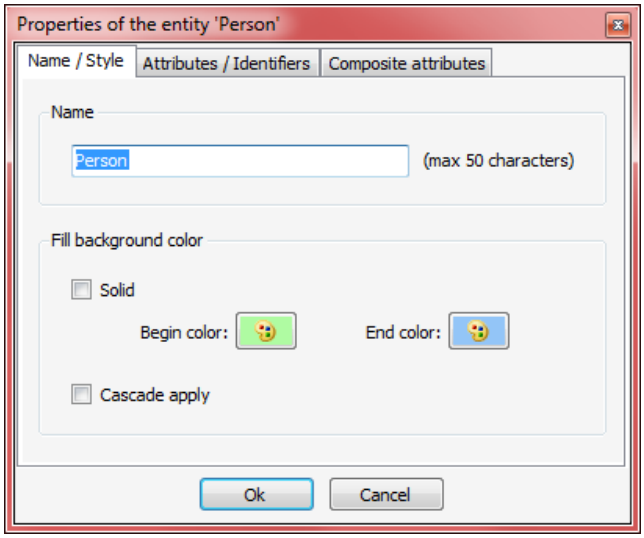


Figura 3.7: *modifica del nome e dello stile di un'entità.*

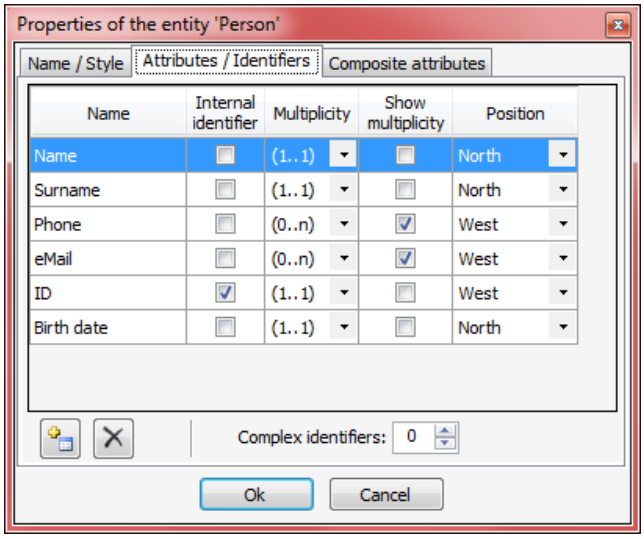


Figura 3.8: *modifica degli identificatori e degli attributi di un'entità.*

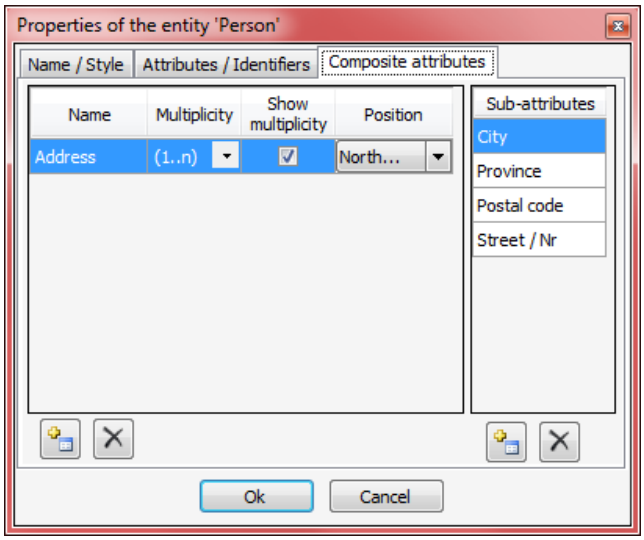


Figura 3.9: *modifica degli attributi composti di un'entità.*

Come si nota in figura 3.8 è presente un'opzione particolare, nominata *complex identifiers* (identificatori complessi). Tramite questa l'utente può definire il numero di identificatori complessi legati all'entità in oggetto; gli identificatori complessi sono una caratteristica peculiare di SER-DE con cui è possibile specificare qualunque tipo di identificatore a prescindere dalla forma geometrica e dal significato logico che questo può assumere. Grazie a questo oggetto si possono definire identificatori interni composti, esterni misti ed esterni puri. Come illustrato in figura 3.10, è invero un oggetto

molto versatile poiché, utilizzando l'opzione *segment type* (tipologia segmento), l'utente può scegliere tra quattro forme geometriche disponibili e con l'opzione *rotate clockwise (90°)* (ruota in senso orario di 90°) può ruotare l'oggetto da 0° a 360°. Infine, gli identificatori complessi possono essere ridimensionati o spostati in qualunque posizione del canvas; per fare questo è sufficiente passare con il mouse in prossimità dell'identificatore stesso e prestare attenzione al cursore del mouse. In corrispondenza del nodo centrale dell'identificatore, detto pivot, la grafica del cursore sarà composta da quattro frecce direzionali; a questo punto, tenendo premuto il tasto sinistro e spostando il mouse, verrà spostato anche l'identificatore. In tutti gli altri nodi dell'identificatore la grafica del cursore sarà composta da due sole frecce, il cui orientamento indica chiaramente in quali direzioni è possibile ridimensionare l'identificatore stesso.

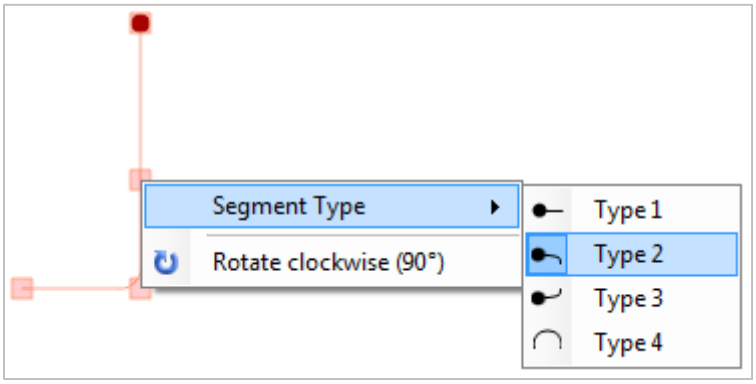


Figura 3.10: *esempio d'identificatore complesso.*

Passando all'oggetto relazione, le proprietà specifiche di questo sono sempre relative a nome, stile e attributi, in maniera similare a quella già vista per l'oggetto entità (figura 3.11 e figura 3.12).

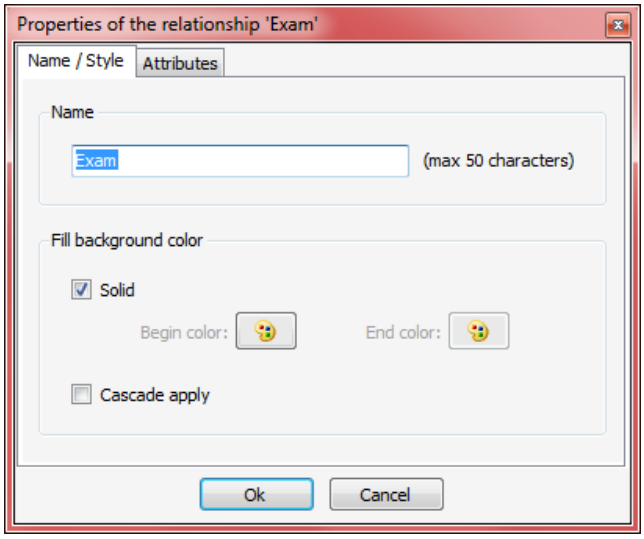


Figura 3.11: *modifica del nome e dello stile di una relazione.*

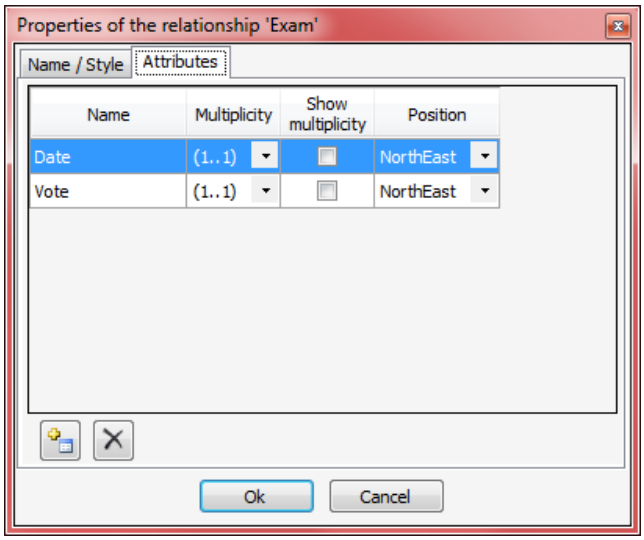


Figura 3.12: *modifica degli attributi di una relazione.*

Per quanto concerne le opzioni di subset e generalizzazioni, queste sono facilmente accessibili direttamente nel menu contestuale che appare sempre premendo il tasto destro (figura 3.13).

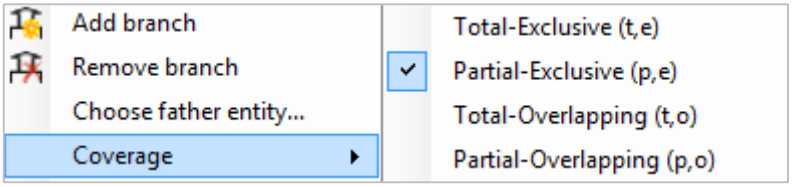


Figura 3.13: *menu contestuale di una generalizzazione.*

La funzionalità *add branch* (aggiungi ramo) consente di aggiungere un ramo alla generalizzazione o al subset selezionato. In maniera opposta la funzionalità *remove branch* (rimuovi ramo) elimina un ramo dalla generalizzazione selezionata. Tramite l'opzione *choose father entity* (scegli entità padre) è possibile definire qual è l'entità padre di una data generalizzazione o di un subset; verranno proposte solo entità che non sono già padre o figlio di un'altra generalizzazione o subset.
Infine, servendosi dell'opzione *coverage* (copertura), è possibile indicare la copertura della gerarchia o del subset selezionato.

Ultima funzionalità specifica, relativa agli oggetti disegnabili con SER-DE, è l'oggetto connessione che consente di collegare tra loro entità e relazioni oppure entità e generalizzazioni o subset. Seppur graficamente identiche, le tipologie di connessioni disponibili cambiano profondamente dal punto di vista logico; pertanto, premendo il tasto destro in corrispondenza di una connessione, l'utente può trovarsi di fronte a differenti menu contestuali.

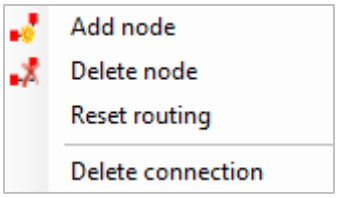


Figura 3.14: *menu contestuale di una connessione entità-generalizzazione.*

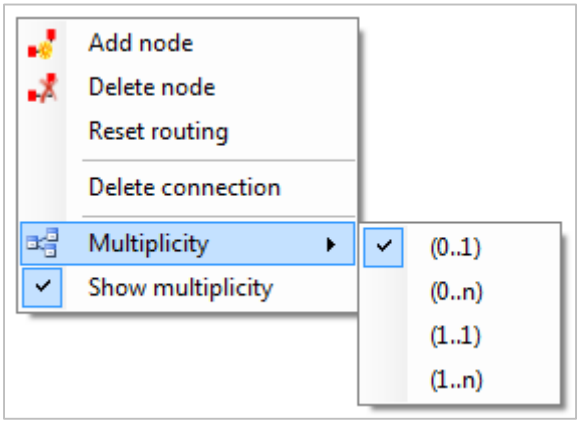


Figura 3.15: *menu contestuale di una connessione entità-relazione.*

Come si nota nelle figure 3.14 e 3.15, alcune opzioni fornite dal menu contestuale sono comuni a prescindere dal tipo di connessione; queste sono *add node/delete node* (aggiungi nodo/elimina nodo) che consentono di aggiungere o rimuovere un nodo nel caso in cui sia necessario disegnare una connessione con curve o deviazioni. In corrispondenza dei nodi, infatti, è possibile spostare i segmenti della connessione tenendo premuto il tasto sinistro del mouse e muovendolo dove desiderato. Inoltre, è possibile lasciar decidere a SER-DE quale instradamento assegnare alla connessione; questo avviene per mezzo dell'opzione *reset routing* (resetta instradamento) alla quale è associato un algoritmo in grado di calcolare il percorso più breve tra due oggetti, creando al massimo una sola curva con angolo di 90°. L'opzione *delete connection* (cancella connessione) procederà, senza alcuna conferma, all'eliminazione della connessione selezionata.
Come mostrato in figura 3.15, nel caso in cui la connessione sia tra un oggetto entità e uno relazione, il menu contestuale conterrà due ulteriori voci: *multiplicity* (molteplicità) e *show multiplicity* (mostra molteplicità).
Per mezzo di queste opzioni l'utente può indicare i valori di partecipazione e cardinalità scegliendo tra i quattro previsti e può scegliere se tali valori devono essere visibili in un'etichetta di testo disegnata sul diagramma o se devono rimanere come sola caratteristica logica, ossia assegnata ma non visibile.

Infine abbiamo la funzionalità di gestione della documentazione, relativa ai diagrammi E-R nel loro complesso, raggiungibile dal menu *Tools*, opzione *show notes* (mostra note); al momento, come illustrato in figura 3.16, è solamente possibile specificare delle note generiche, con lunghezza arbitraria, in una finestra dedicata il cui stile ricorda quello di un post-it, ciò proprio per richiamare l'idea di una nota che viene associata al diagramma.

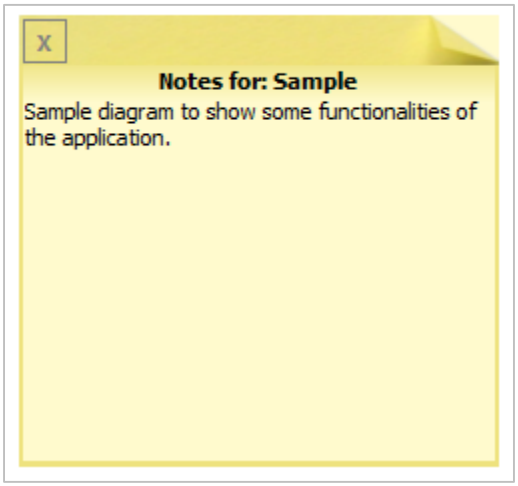


Figura 3.16: *mostra/modifica note diagramma.*

Non è possibile avere una struttura delle note su più livelli, così come non è possibile specificare una nota per un oggetto diverso dal generico *ERCanvas*.

Capitolo 4
IMPLEMENTAZIONE

Lo sviluppo di un applicativo di quest'entità ha richiesto centinaia di ore uomo di sviluppo. È opportuno, pertanto, suddividere accuratamente le varie fasi che hanno portato alla realizzazione di SER-DE, dalla progettazione, allo sviluppo del codice, per finire con un accurato test e le ottimizzazioni che ne sono seguite.

4.1 Strumenti utilizzati

- Analisi, progettazione, modellazione: la progettazione è stata imperniata su alcuni dei diagrammi messi a disposizione dal linguaggio UML; a tal proposito è stato scelto di utilizzare uno degli applicativi ad oggi più diffusi, ossia UModel [AUM], realizzato da *Altova*. Il maggior punto di forza di UModel è il supporto completo a tutte le versioni del .NET framework e al linguaggio C#; in tal modo è stato possibile creare una prima bozza di architettura che, successivamente, è stata facilmente sincronizzata con tutte le modifiche avvenute in fase di sviluppo.

- Scrittura del codice, debug, test: per queste fasi, molto impegnative, la scelta è ricaduta sull'ambiente di sviluppo integrato di *Microsoft*, Visual Studio 2010 [MVS]; è stato scelto il linguaggio C# e la versione del

.NET framework 2.0. Il progetto realizzato è compatibile anche con la più recente versione di Visual Studio al momento disponibile, ossia la 2012.

- Documentazione tecnica: utilizzando Sandcastle Help File Builder [SHF], un ottimo progetto open source, è stata realizzata la documentazione tecnica nel formato standard di Windows, ossia CHM[9]. Con la semplice modifica di un parametro è possibile ottenere la stessa documentazione in formato HTML.

- Setup: grazie ad Inno Setup [INN], altro ottimo strumento gratuito, è stato possibile realizzare un setup standard, compatibile con tutti i sistemi operativi Windows.

- Manuale utente: questo manuale è stato redatto utilizzando l'editor di testo di *Microsoft*, Word [MWD], ed esportando dallo stesso direttamente in formato PDF al fine di consentire una più ampia compatibilità con i sistemi degli utenti finali.

4.2 Analisi e modellazione

Per prima cosa è stato deciso in quali e quanti pacchetti suddividere il progetto, disegnando il diagramma dei pacchetti UML (figura 4.1).
Componente fondamentale, al quale tutti gli altri fanno riferimento, è il framework .NET nello specifico utilizzato attraverso il linguaggio C#. Nella figura 4.1 questo componente è indicato con la dicitura *C# Profile (from root)*; tale dicitura non è modificabile poiché creata direttamente da UModel.

[9] Microsoft Compiled HTML Help

Sempre in figura 4.1 troviamo il pacchetto *SharpERD*, che è il cuore dell'applicativo realizzato, contenente tutte le funzionalità e gli oggetti di business; parallelamente abbiamo *CustomControls*, una collezione di controlli grafici ampiamente utilizzati da SER-DE.
Infine abbiamo i sotto-pacchetti *Properties*, creati automaticamente da Visual Studio e contenuti nei già citati *SharpERD* e *CustomControls*; questi sotto-pacchetti sono utilizzati per gestire le impostazioni applicative, le risorse e le informazioni del pacchetto stesso (i.e. versione, produttore, copyright).

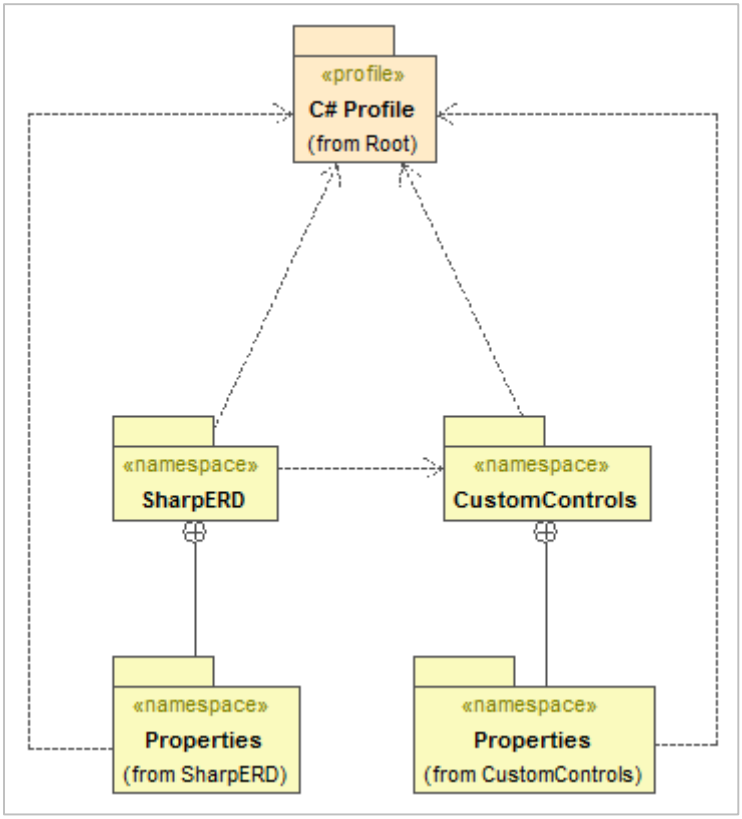


Figura 4.1: *diagramma dei pacchetti di SER-DE.*

Una volta definiti i blocchi costituenti l'architettura del sistema si è proceduto con un'analisi più approfondita, realizzata per mezzo del diagramma UML delle classi (figura 4.2).

Questo diagramma, in versione completa, è molto complesso, pertanto, in figura 4.2, ne viene illustrata solamente una porzione, quella di maggior interesse, poiché relativa alla modellazione degli oggetti cardine necessari alla creazione di un diagramma E-R. Non troviamo, ad esempio, la modellazione delle form contenute all'interno del pacchetto *SharpERD*, così come del pacchetto *CustomControls* e dei sotto-pacchetti *Properties*. Tale omissione è giustificata dal fatto che questi oggetti sono puramente applicativi e non possiedono alcuna peculiarità correlata alla nozione di diagramma E-R.

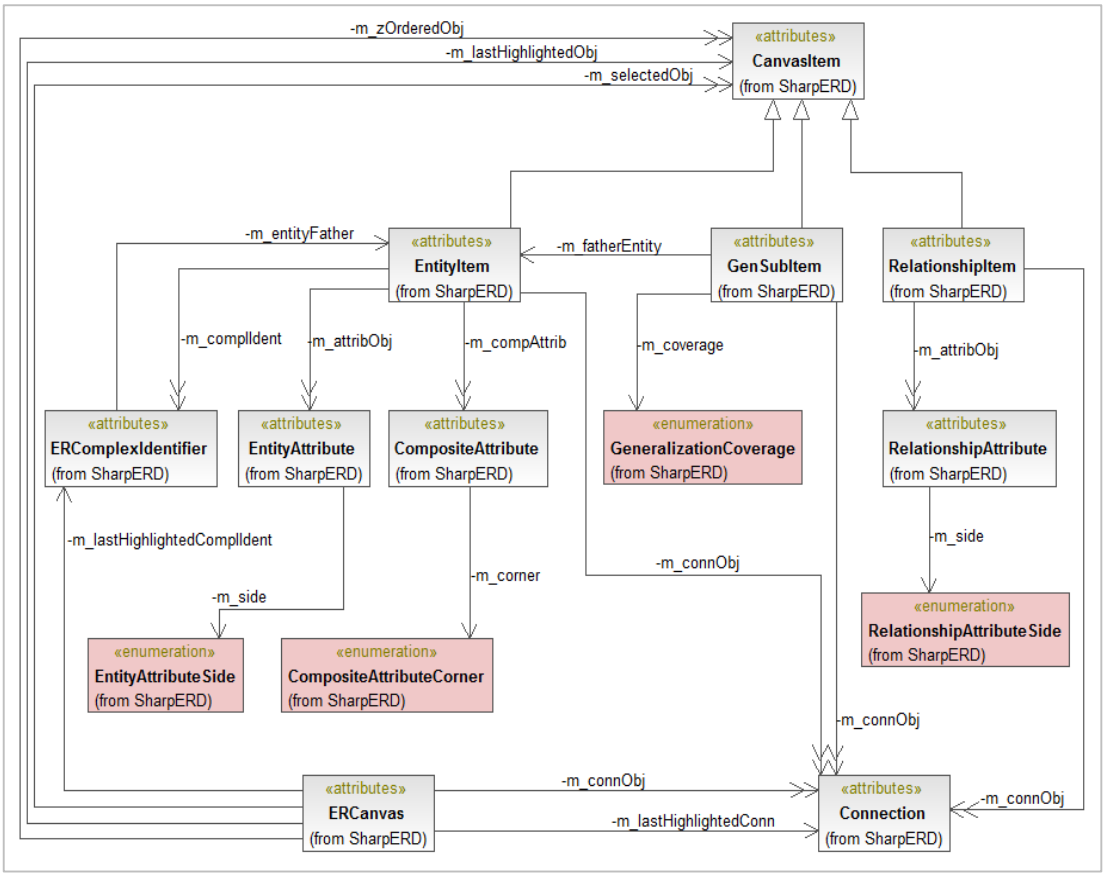


Figura 4.2: *diagramma delle classi di SER-DE.*

Elemento fondamentale del diagramma delle classi è *CanvasItem*: questo è un oggetto astratto, ossia una classe base (superclasse) contenente le firme

dei metodi relative alle funzionalità e alle caratteristiche comuni degli oggetti disegnabili sul canvas di disegno (figura 4.3).

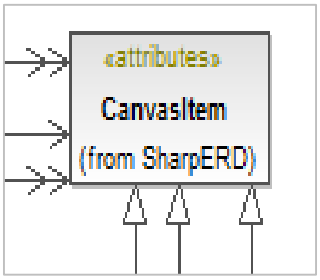


Figura 4.3: *particolare UML, classe base CanvasItem.*

Da questa classe, seguendo i principi dell'ereditarietà, derivano tre ulteriori oggetti fondamentali che rappresentano la specializzazione di un generico oggetto *CanvasItem* nei reali oggetti che verranno posizionati sul canvas; questi sono *EntityItem*, *GenSubItem* e *RelationshipItem* (figura 4.4).

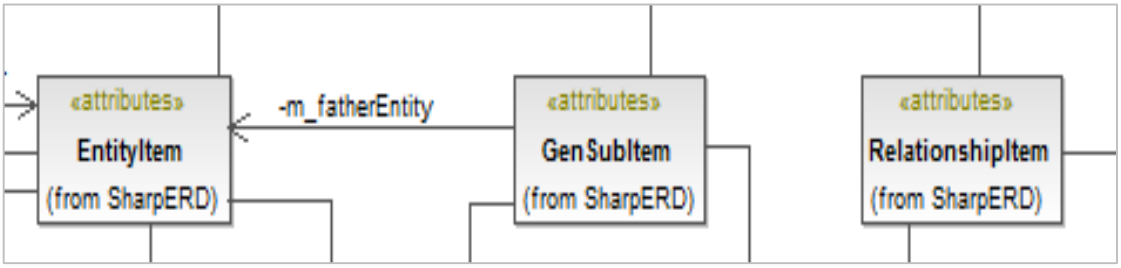


Figura 4.4: *particolare UML, specializzazioni di CanvasItem.*

Tali oggetti, tramite la tecnica di override, contengono le implementazioni specifiche dei metodi dichiarati nell'oggetto *CanvasItem*. Un metodo notevole che viene ridefinito in questi sotto-oggetti è *Draw* (disegna); qui viene esplicitato il codice necessario alla creazione grafica dell'oggetto. Ad esempio per disegnare un oggetto entità si disegna prima un rettangolo, quindi si disegnano delle linee con un cerchio per ogni attributo presente e, per finire, tutte le etichette di testo relative al nome dell'entità, degli attributi e delle eventuali molteplicità degli stessi.

Proseguendo verso la parte bassa del diagramma di figura 4.2 notiamo che all'oggetto *EntityItem* sono collegate tre collezioni di oggetti, che rappresentano le proprietà fondamentali di un'entità; questi sono: *EntityAttribute*, *CompositeAttribute*, *ERComplexIdentifier* (figura 4.5).

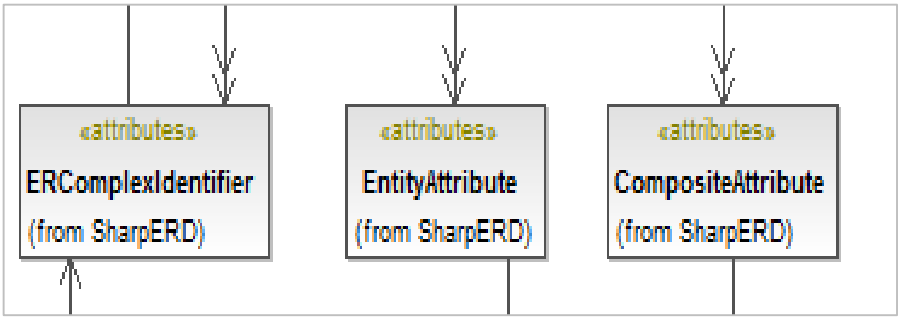


Figura 4.5: *particolare UML, oggetti collegati ad EntityItem.*

L'oggetto *EntityAttribute*, così come *CompositeAttribute*, è una struttura molto semplice contenente solo le proprietà *get* e *set* necessarie alla valorizzazione di alcuni parametri specifici (i.e. nome, molteplicità, posizione); tali parametri verranno poi utilizzati dal metodo *Draw* dell'entità padre per poter disegnare correttamente tutti gli attributi ad esso collegati. Pertanto *EntityAttribute* e *CompositeAttribute* non possiedono un proprio metodo *Draw*, questo perché non possono essere spostati e la loro presenza è sempre legata all'entità padre e alla sua posizione.

L'oggetto *ERComplexIdentifier*, invece, possiede un proprio metodo *Draw* poiché è un oggetto molto complesso da disegnare e, nonostante sia comunque collegato obbligatoriamente ad un'entità padre, può essere spostato in qualunque posizione del canvas a prescindere dall'oggetto entità cui appartiene.

A loro volta *EntityAttribute* e *CompositeAttribute*, contengono le referenze alle rispettive enumerazioni *EntityAttributeSide* e *CompositeAttributeCorner*. Grazie a queste enumerazioni viene indicata la posizione degli attributi in relazione all'entità cui appartengono (figura 4.6).

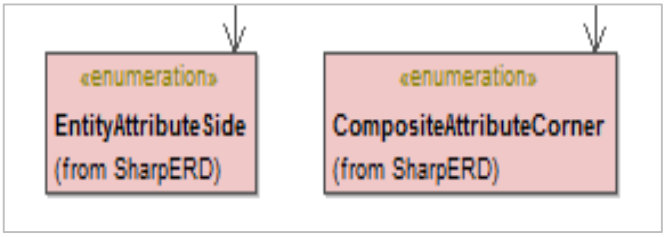


Figura 4.6: *particolare UML, enumerazioni di EntityItem.*

L'oggetto *GenSubItem* non contiene invece alcun riferimento ad altri oggetti di rilievo, fatta eccezione per l'enumerazione *GeneralizationCoverage*, tramite la quale è possibile specificare la copertura (opzionale) di una generalizzazione (figura 4.7).

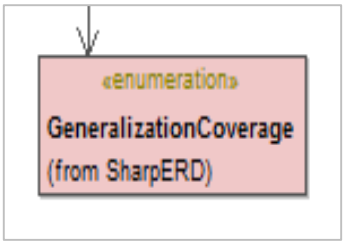


Figura 4.7: *particolare UML, enumerazione di GenSubItem.*

Dei tre oggetti derivati da *CanvasItem* abbiamo, infine, *RelationshipItem*; questo possiede una collezione di *RelationshipAttribute* che a loro volta contengono dei riferimenti all'enumerazione *RelationshipAttributeSide* tramite la quale viene specificata la posizione dell'attributo rispetto alla relazione in cui è contenuto (figura 4.8).
Come già spiegato per *EntityAttribute* e *CompositeAttribute* anche *RelationshipAttribute* è un semplice contenitore di proprietà (*get* e *set*) atte a valorizzare alcuni parametri specifici; parametri che verranno successivamente utilizzati dal metodo *Draw* dell'oggetto *RelationshipItem*, cui gli attributi fanno riferimento, per disegnare su canvas gli attributi stessi.

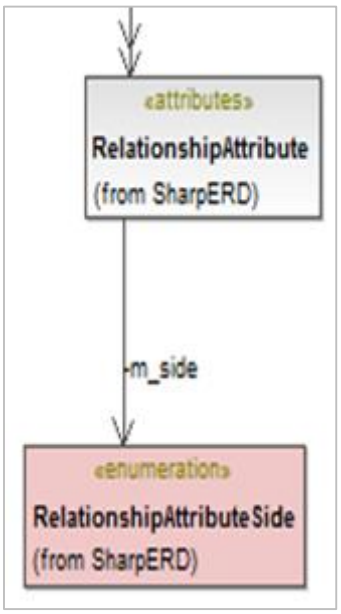


Figura 4.8: *particolare UML, attributi ed enumerazione di RelationshipItem.*

Ultimi due oggetti fondamentali illustrati nel diagramma delle classi di figura 4.2 sono *ERCanvas* e *Connection* (figura 4.9).

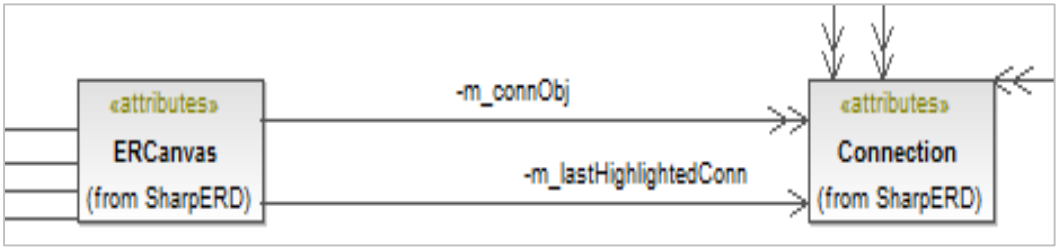


Figura 4.9: *particolare UML, oggetti ERCanvas e Connection.*

La classe *ERCanvas* è un oggetto piuttosto complesso, in grado di svolgere una pletora di compiti differenti.

Fondamentalmente rappresenta l'area di disegno sulla quale è possibile disegnare tutti gli altri oggetti; tuttavia non si limita a fungere da contenitore, infatti, ad esempio, mette a disposizione i metodi che consentono di spostare un *CanvasItem* all'interno del canvas, oppure i metodi per selezionare i *CanvasItem* contenuti all'interno di una specifica area.

Sempre all'interno di *ERCanvas* troviamo il codice in grado disegnare tutti gli oggetti sul canvas nell'ordine corretto (rispettando lo z-order) così come il codice che è in grado di stabilire se ciò che l'utente sta cercando di fare è ammissibile oppure no ed eventualmente bloccarlo (i.e. l'utente vuole collegare tra loro due relazioni utilizzando un'associazione).
Pertanto *ERCanvas* può essere considerato un vero e proprio strato software e non un semplice oggetto, nello specifico si tratta del cosiddetto *biz-layer*.
Infine troviamo la classe *Connection*, un oggetto relativamente semplice il quale, a partire da alcune proprietà in esso contenute, è in grado di disegnare una linea di connessione, sia nel caso questa sia un'associazione tra entità e relazione, sia nel caso in cui si tratti di una connessione tra generalizzazioni ed entità derivate.
Come si può notare in figura 4.2 tutte le specializzazioni di *CanvasItem* contengono una collezione di oggetti *Connection* poiché ad ognuno di questi *CanvasItem* può essere collegata una collezione di *Connection*. Anche la classe *ERCanvas*, tuttavia, contiene una collezione di *Connection*; questa verrà utilizzata per disegnare fisicamente le connessioni sul canvas, richiamando, per ogni oggetto *Connection*, il relativo metodo *Draw*.
Cosa fondamentale da notare è che la classe *ERCanvas* non contiene alcun riferimento a *EntityItem*, *GenSubItem* e *RelationshipItem*. Sfruttando l'ereditarietà, infatti, l'unico oggetto conosciuto da *ERCanvas* è quello generico, *CanvasItem*. In questo modo è stato possibile creare un codice unico ma in grado di lavorare su oggetti differenti, ottenendo così uno sviluppo più snello ed evitando, soprattutto, delle ripetizioni di porzioni di codice simili se non addirittura identiche.

4.3 Sviluppo

Rispettando quanto progettato, sono stati creati due progetti C# con Visual Studio 2010, *Custom Controls.csprj* e *SharpER Diagram Editor.*csproj, descritti in dettaglio nelle sezioni seguenti.

4.3.1 Progetto C# Custom Controls

Custom Controls contiene tutti i controlli grafici personalizzati la cui creazione si è resa necessaria per realizzare SER-DE:

- *DoubleBuffPanel*;
- *GradientPanel*;
- *MoveablePanel*;
- *ScrollingText*.

DoubleBuffPanel è un controllo fondamentale, poiché consente di ottenere un rendering più fluido degli oggetti sul canvas di disegno; l'oggetto *Panel*, fornito di base dal .NET framework e utilizzato inizialmente da SER-DE come superficie di disegno, non implementa la tecnica del *double buffering*[10]. È stato pertanto sostituito con l'oggetto *DoubleBuffPanel* che deriva sempre dall'oggetto *Panel*, ma ne ridefinisce alcune proprietà che consentono l'abilitazione della tecnica suddetta e altre ottimizzazioni minori.
L'oggetto *GradientPanel* è utilizzato come sfondo a gradiente del canvas di disegno; anche questo deriva dall'oggetto *Panel* e, ridefinendone completamente il metodo *OnPaint*, è in grado di creare un effetto a gradiente verticale con i due colori specificati dallo sviluppatore.

[10] Per evitare lo sfarfallio delle immagini, l'area corrispondente alla superficie di disegno viene prima copiata in un buffer su RAM, quindi tutte le trasformazioni grafiche vengono eseguite solo sul buffer e, al termine, viene inviato al monitor il contenuto del buffer con tutte le modifiche alla superficie di disegno associata; questa viene pertanto visualizzata in una sola operazione, evitando sfarfallii relativi ad operazioni parziali [DBG].

MoveablePanel è un controllo che deriva ancora una volta da *Panel* ma che, a differenza di quest'ultimo, consente il proprio spostamento utilizzando il mouse; questo controllo è utilizzato dal pannello di disegno mobile, precedentemente illustrato in figura 3.4, e dalla funzionalità *show notes* (mostra note) illustrata in figura 4.10 e accessibile tramite il menu *Tools*.

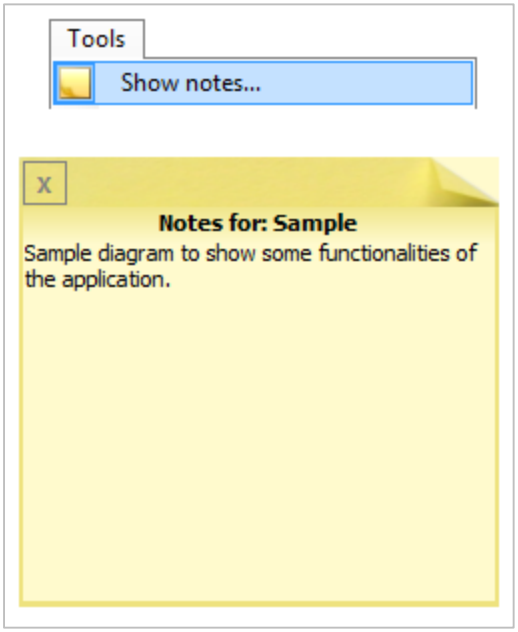


Figura 4.10: *porzione del menu Tools e funzionalità mostra note.*

L'ultimo controllo contenuto in *Custom Controls* è *ScrollingText*, un semplice oggetto che deriva direttamente dalla classe base *Control*; questo oggetto, sfruttando un timer e ridefinendo il metodo *OnPaint*, è in grado di renderizzare a video un'etichetta di testo che, come suggerisce il nome stesso del controllo, si muove da una parte all'altra del suo contenitore. Viene utilizzata dalla funzionalità *about* (a proposito di) dove il contenitore è una form il cui compito è mostrare alcune informazioni su SER-DE (i.e. autore, versione).

4.3.2 Progetto C# SharpER Diagram Editor

Il contenuto del progetto *SharpER Diagram Editor*, vista la notevole complessità, è stato suddiviso in varie cartelle.

Nella cartella *CanvasObjects* abbiamo tutti gli oggetti fondamentali di SER-DE, ossia quelli abbondantemente illustrati nel paragrafo 4.2; ad ogni oggetto è associata una classe e, per rispettare la buona prassi della OOP[11], ogni classe è contenuta in un proprio file con estensione .cs:

- *CanvasItem;*
- *CompositeAttribute;*
- *Connection;*
- *EntityAttribute;*
- *EntityItem;*
- *ERCanvas;*
- *ERComplexIdentifier;*
- *GenSubItem;*
- *RelationshipAttribute;*
- *RelationshipItem;*
- *SharpEnumConst.*

Aldilà della struttura e delle relazioni tra oggetti, già illustrate in figura 4.2, analizzando il codice notiamo che questo è strettamente correlato al rendering grafico degli oggetti stessi. La tecnologia usata per tale rendering è GDI+ [GDI]; questa è stata scelta poiché più semplice da utilizzare rispetto ad OpenGL [OGL] o DirectX [DTX], il cui utilizzo sarebbe stato ingiustificato data la natura esclusivamente bidimensionale dei diagrammi realizzati.

Risulta particolarmente interessante l'analisi di uno dei metodi fondamentali, comune a molti degli oggetti appena elencati: il metodo *Draw*.

[11] Object-Oriented Programming

Di seguito viene mostrata una porzione di codice di tale metodo, estratta dalla classe *Connection.*

```
/// <summary>
/// Draws the current instance of this connection.
/// </summary>
/// <param name=" drawSurface ">The drawing surface.</param>
/// <returns>The drawn bitmap.</returns>
public Bitmap Draw(Bitmap drawSurface)
{
   // Creates a Graphics object that references the bitmap and
   // automatically clear it when finished
   using (Graphics objGraphics = Graphics.FromImage(drawSurface))
   {
        […]
        objGraphics.DrawLines(new Pen(color, 1), nodes.ToArray());
        […]

        return drawSurface;
   }
}
```

Da questo codice possiamo notare alcune peculiarità, come l'utilizzo dell'istruzione *using* il cui compito è quello d'indicare al compilatore che l'oggetto *Graphics* istanziato dovrà essere distrutto quando non più necessario. L'utilizzo di questa istruzione è fondamentale poiché ogni oggetto *Graphics* istanziato occupa qualche megabyte di memoria e, se non venissero distrutti subito dopo il loro utilizzo, la memoria di sistema si saturerebbe molto velocemente.

Notiamo inoltre come il rendering a video non viene eseguito direttamente dal metodo *Draw* che si limita a disegnare ciò che è di sua competenza sulla bitmap passata al metodo stesso (nel caso dell'oggetto *Connection* è la linea relativa alla connessione rappresentata).

L'obiettivo del metodo, pertanto, è quello di ritornare al chiamante (*ERCanvas*) l'oggetto *Bitmap* aggiornato con tutte le modifiche grafiche; sarà *ERCanvas* ad occuparsi di memorizzare tutte le modifiche, per tutti gli

oggetti, mentre spetterà a *MainForm* il compito di coordinarne il rendering a video nel miglior modo possibile. Solo l'oggetto *MainForm* può occuparsi di tale compito poiché solo questo conosce il *DoubleBuffPanel* sul quale effettuare il rendering finale.

Nella cartella *Forms* troviamo invece tutte le form utilizzate dall'applicativo per interagire con l'utilizzatore dello stesso; queste sono:

- *AboutForm;*
- *AttribValueForm;*
- *FatherEntityForm;*
- *MainForm;*
- *NewDiagramForm;*
- *PropertiesForm;*
- *Splash.*

AboutForm è quella contenente tutte le informazioni sul programma, in particolare viene mostrata la versione, l'autore e tutte le informazioni relative alla licenza utilizzata e ai contatti. È una form il cui stile è stato personalizzato, pertanto non presenta i classici pulsanti e la barra del titolo tipici del sistema operativo (figura 4.11).

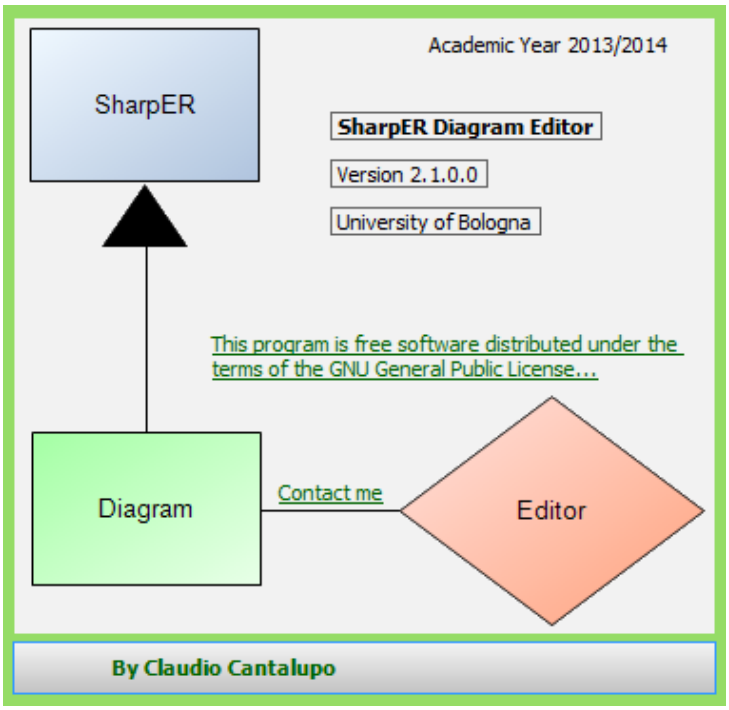


Figura 4.11: *AboutForm.*

Tramite *AttribValueForm* è possibile indicare una cardinalità personalizzata di un attributo, minima e massima (i.e. 3..n), sia che questo appartenga ad un'entità o ad una relazione. Anche lo stile di questa form è stato personalizzato, quindi non appaiono i tasti per minimizzare/massimizzare e chiudere la finestra, né la barra del titolo specifica del sistema operativo in uso (figura 4.12).

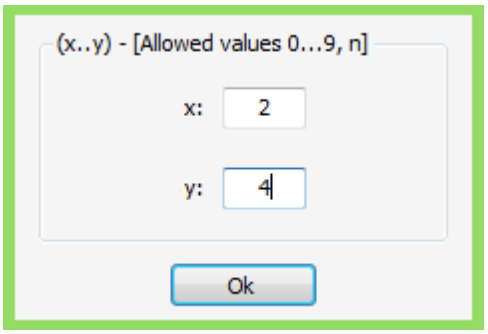


Figura 4.12: *AttribValueForm.*

FatherEntityForm è l'interfaccia tramite la quale indicare l'entità padre per una data generalizzazione o subset (figura 4.13); a livello di codice questo si traduce nell'impostazione, tramite *set*, di una proprietà sull'oggetto *EntityItem* e di un'altra proprietà sull'oggetto *GenSubItem*. È possibile accedere a tale funzionalità per mezzo della voce di menu illustrata precedentemente in figura (3.13).

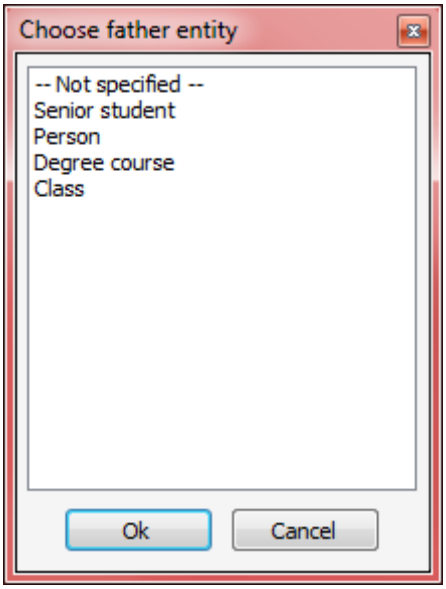


Figura 4.13: *FatherEntityForm.*

MainForm è la finestra principale dell'applicativo il cui compito primario è quello di contenere tutti gli altri oggetti e tutte le altre form (figura 4.14). Il codice di questa form è molto complesso e supera le 3000 righe, poiché all'oggetto *MainForm* sono demandati una pletora di compiti fondamentali come il salvataggio, l'apertura e la chiusura di un diagramma, lo zoom del diagramma, la stampa, il copia e incolla di oggetti tra diagrammi differenti e tantissimo altro ancora.
Solo un'accurata analisi del codice può consentire di comprendere appieno tutte le funzionalità implementate.

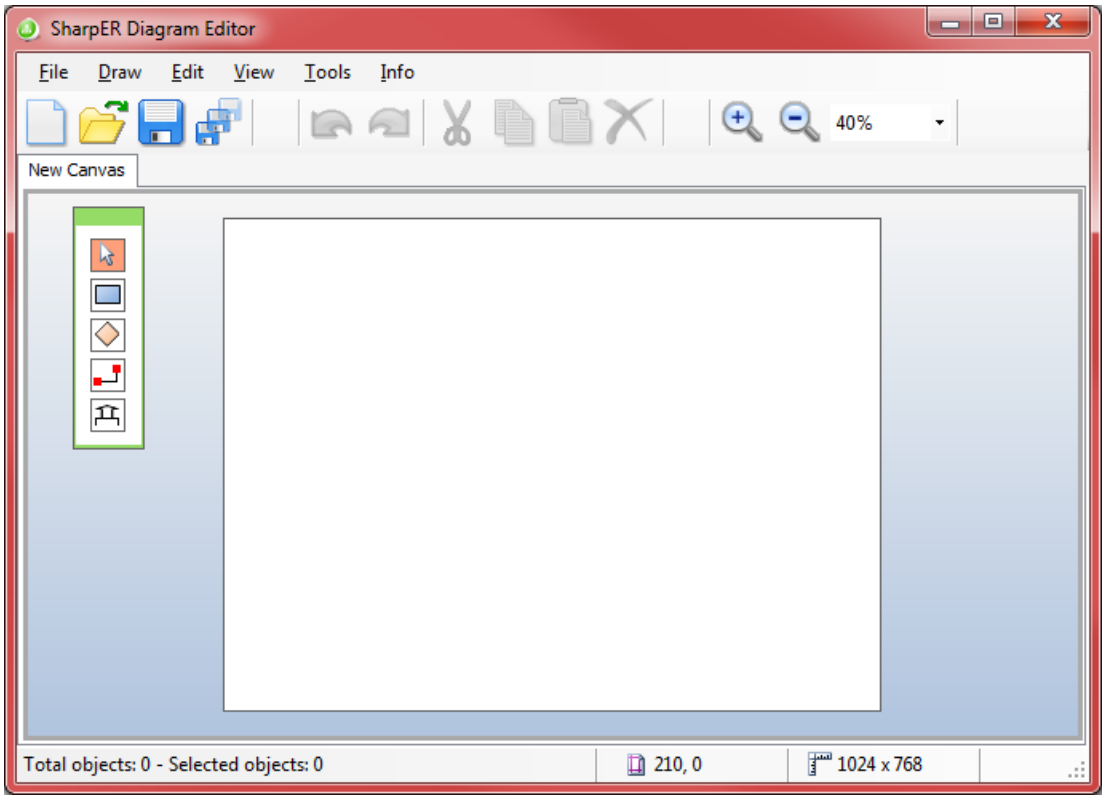


Figura 4.14: *MainForm.*

NewDiagramForm è la finestra che appare nel momento in cui si decide di creare un nuovo diagramma; è necessario specificare la dimensione, in pixel, del canvas di disegno (figura 4.15). Tramite *set* della proprietà *Size* viene impostata la dimensione del *DoubleBuffPanel* relativo al canvas corrente.

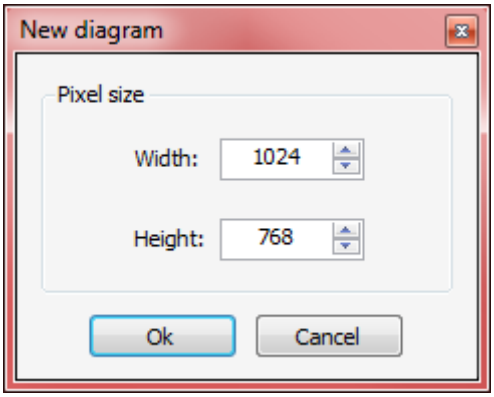


Figura 4.15: *NewDiagramForm.*

Attraverso *PropertiesForm* si accede all'impostazione delle proprietà di entità e relazioni (figura 4.16); per maggiori dettagli fare riferimento al paragrafo 3.2.2.

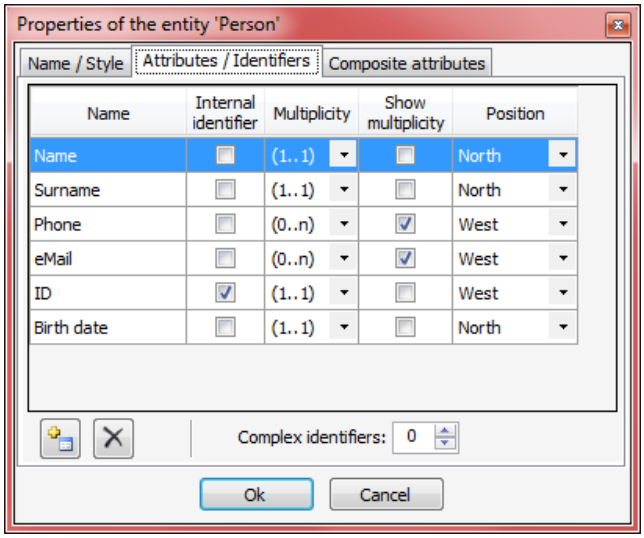


Figura 4.16: *PropertiesForm.*

Infine abbiamo la form *Splash*, una finestra con stile personalizzato e alla quale è associata un'immagine con il logo, autore e versione di SER-DE (figura 4.17) che appare durante il caricamento all'avvio dell'applicativo.

Figura 4.17: *Splash.*

Nella cartella *Miscellaneous* abbiamo due classi di utilità:

- *FileAssociation;*
- *ObjectCopier.*

L'oggetto *FileAssociation* è una classe tramite la quale viene gestita l'associazione dell'estensione applicativa (.serx). In questo modo i file contenenti diagrammi E-R realizzati con SER-DE verranno riconosciuti e associati a livello di sistema operativo e verrà ad essi assegnata un'icona (figura 4.18); aprendo il file .serx direttamente dal sistema operativo verrà automaticamente lanciato SER-DE con il file scelto già caricato.

Figura 4.18: *icona di sistema per l'estensione di file .serx.*

La classe *FileAssociation* implementa tutto il codice necessario alla scrittura dei dati sul registro di sistema, al fine d'indicare al sistema operativo come gestire l'associazione. Tale associazione viene applicata in automatico solo al primo avvio di SER-DE ma è modificabile manualmente accedendo al menu *Tools* (strumenti) e alle relative voci *Register .SERX extension* (registra estensione .SERX) e *Unregister .SERX extension* (deregistra estensione .SERX) come illustrato in figura 4.19.

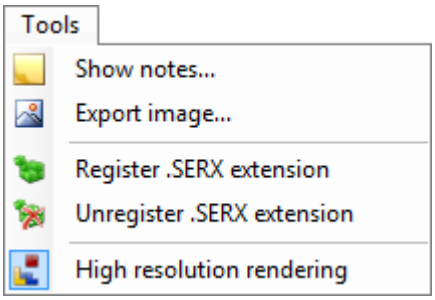


Figura 4.19: *menu Tools.*

ObjectCopier è una classe statica contenente un solo metodo, statico e generico, in grado di clonare qualunque oggetto passato purché questo non sia nullo oppure non serializzabile. La particolarità di *ObjectCopier* è quella di riuscire a clonare completamente un'istanza di un oggetto, a prescindere dalla struttura e da eventuali sotto-oggetti in esso contenuti e senza che sia necessario specificare alcun dettaglio sulla composizione di ciò che si sta clonando. Questa classe viene utilizzata per una moltitudine di compiti tra cui il copia e incolla, la gestione delle funzionalità *undo/redo* (disfare/rifare), accessibile dal menu *Edit* indicato in figura 4.20, e altro ancora.

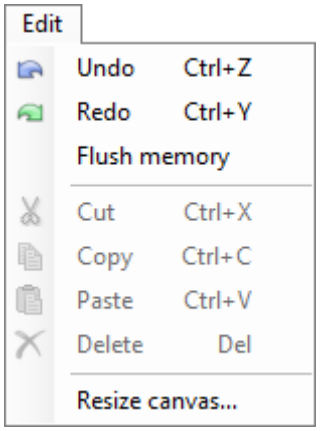


Figura 4.20: *menu Edit.*

Nella cartella radice del progetto troviamo tutti i file di configurazione creati automaticamente da Visual Studio, il file di configurazione del log applicativo, implementato tramite la libreria open source *log4net* [L4N] e la classe statica *Main* contenuta nel file *Program.cs*.

Questa classe, obbligatoria per poter creare un programma .NET con interfaccia grafica, contiene tutte le inizializzazioni di SER-DE oltre ad alcuni controlli aggiuntivi come la verifica di istanza applicativa già presente o il controllo sui permessi di SER-DE per evitare limitazioni dello stesso durante il normale utilizzo.

Tra le funzionalità principali implementate rimangono da analizzare, in particolare, le seguenti:

- salvataggio e apertura;
- stampa;
- esporta immagine;
- ridimensiona canvas;
- zoom in/zoom out;
- mostra griglia/allineamento alla griglia.

La funzionalità di salvataggio, accessibile tramite il menu *File* (figura 4.21), è stata implementata nel modo più conveniente e rapido possibile, ossia utilizzando un oggetto predefinito del framework .NET, il *BinaryFormatter*; questo *formatter* è in grado di trasformare un oggetto, nel nostro caso *ERCanvas* con tutto il suo contenuto, in un blob di bit che vengono successivamente serializzati sul file con estensione .serx. L'apertura di un file .serx, in maniera inversa, deserializza il blob di bit in un oggetto *ERCanvas*, servendosi sempre di un *BinaryFormatter.*

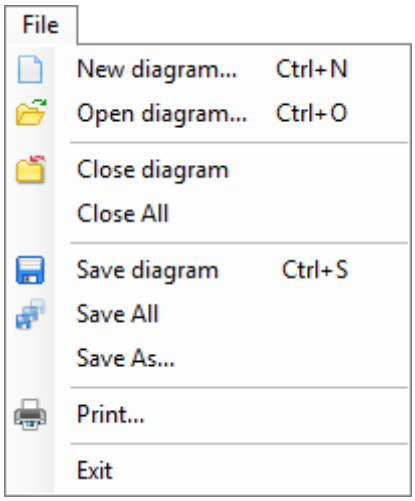


Figura 4.21: *menu File.*

La funzionalità di stampa, accessibile dal menu di figura 4.21, è stata implementata servendosi del sottosistema WIA[12] [WIA] messo a disposizione dal sistema operativo (Windows ME e successivi). Grazie a questa architettura di stampa e acquisizione immagini è possibile ottenere, in maniera molto semplice, delle stampe di elevata qualità e con una pletora di opzioni (figura 4.22), il cui sviluppo partendo da zero richiederebbe centinaia di ore uomo.

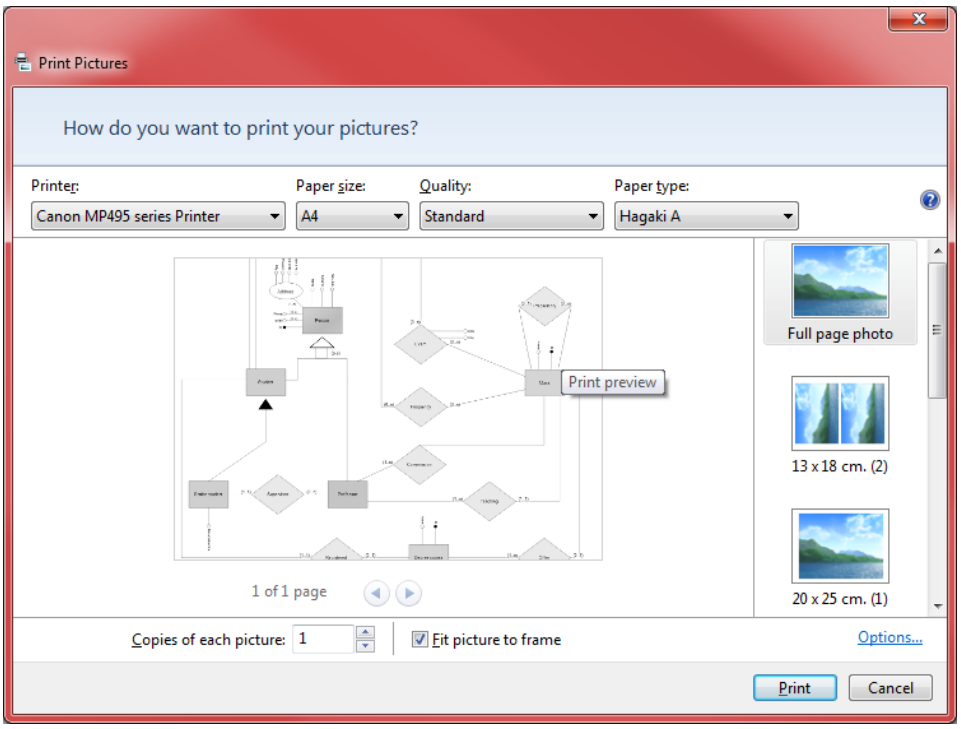


Figura 4.22: *sottosistema di stampa WIA.*

Per poter richiamare l'interfaccia di stampa del sistema operativo è sufficiente referenziare l'assembly *Interop.WIA.dll*, creare un'immagine bitmap temporanea del canvas di disegno e passarla con poche semplici righe di codice all'oggetto *WIA* istanziato.

[12] Windows Image Acquisition

L'utente può inoltre esportare il contenuto del canvas in diversi formati immagine e salvare il relativo file in qualunque posizione desiderata su file system; i formati a disposizione sono JPEG[13], PNG[14], GIF[15] e bitmap. Questa funzionalità è accessibile dal menu *Tools*, voce *Export image*, illustrato in figura 4.19.

Troviamo poi la funzionalità ridimensiona canvas, accessibile dal menu *Edit*, voce *resize canvas*, illustrato in figura 4.20. Questa funzionalità associa all'oggetto *DoubleBuffPanel* relativo al canvas corrente le nuove dimensioni specificate dall'utente, modificando la proprietà *Size*.

Caratteristica base di ogni editor grafico è la possibilità di ingrandire o rimpicciolire la visuale; questo si può ottenere per mezzo delle funzionalità *zoom in/zoom out* (ingrandisci/rimpicciolisci) accessibili dal menu *View* (figura 4.23).

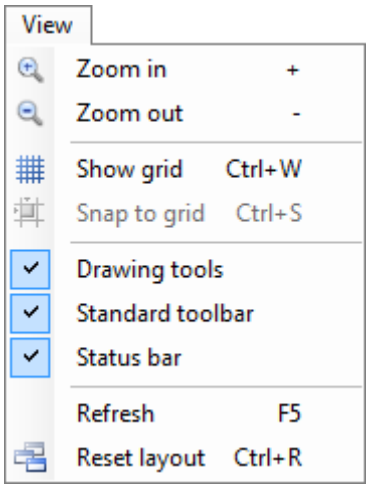


Figura 4.23: *menu View.*

[13] Joint Photographic Experts Group
[14] Portable Network Graphics
[15] Graphics Interchange Format

L'ingrandimento e il rimpicciolimento della visuale sono ottenuti per mezzo del metodo *Scale,* che è predefinito per ogni oggetto derivato da *Control*, pertanto esposto anche dal nostro *DoubleBuffPanel*.

Infine sono presenti le opzioni *show grid* (mostra griglia) e *snap to grid* (allineamento alla griglia); queste sono accessibili sempre dal menu *View*, illustrato in figura 4.23, e risultano essere molto comode in fase di disegno poiché consentono un facile allineamento degli oggetti sul canvas. Questa funzionalità è implementata aggiungendo una serie di linee alla bitmap associata all'oggetto *ERCanvas* corrente; inoltre sono stati introdotti dei controlli in fase di spostamento degli oggetti (evento *MouseMove*) per far si che in automatico questi vengano sempre posizionati in esatta corrispondenza di due linee perpendicolari della griglia (solo se l'allineamento alla griglia è abilitato).

4.4 Test ed ottimizzazioni

La fase di debug e test è stata molto importante è ha richiesto un notevole sforzo per accertarsi che SER-DE fosse compatibile con tutti i sistemi operativi Windows più importanti. In particolare il test è stato eseguito su Windows XP, Vista (32 e 64 bit), 7 (32 e 64 bit) e 8 (32 e 64 bit).
Sono state testate singolarmente tutte le funzionalità dell'applicativo (unit test) e, al termine, è stato testato nuovamente l'applicativo nel suo complesso. Tuttavia il test non è stato automatizzato utilizzando metodologie TDD[16], poiché si è preferito eseguire delle verifiche manuali delegando, ove possibile, a persone differenti il controllo di diverse funzionalità di SER-DE.

[16] Test Driven Development (Sviluppo Guidato dal Test).

Tutti i feedback ottenuti su bug presenti, comportamenti anomali o performance scadenti in particolari situazioni, hanno avviato la cosiddetta fase di *tuning*, nella quale è stato eseguito il refactoring di alcune porzioni di codice e sono state messe in atto diverse ottimizzazioni; di seguito vengono elencate le principali:

- qualità del rendering grafico;
- controllo numero elementi disposti su un singolo diagramma;
- ottimizzazione automatica della memoria e dei livelli di undo/redo;
- controllo istanza applicativa già presente;

Il sistema hardware di riferimento utilizzato per quantificare le differenze prestazionali è un processore quad-core *AMD* A8-3870 con velocità di clock pari a 3.0 GHz e 8 GByte di memoria RAM DDR3 a 1600 Hz. Disco e scheda video non sono rilevanti poiché, aldilà del caricamento iniziale dei file/assembly necessari, non vengono effettuate operazioni su disco ricorrenti e la scheda video viene utilizzata in minima parte dalla tecnologia GDI+ che sfrutta invece la potenza della CPU.

L'utilizzatore può scegliere se il rendering grafico del canvas deve essere eseguito in alta qualità oppure in qualità standard; tale opzione è disponibile nel menu *Tools* (figura 4.19), voce *High resolution rendering* (rendering ad alta risoluzione). A livello di memoria di sistema utilizzata la differenza è minimale, con un risparmio medio di circa il 10% in caso si utilizzi il rendering standard. Più importante è invece la differenza di utilizzo della CPU, con punte di utilizzo del 30% inferiori in caso di operazioni complesse (i.e. spostamento contemporaneo di molti oggetti sul canvas).
Nell'immagine di figura 4.24 si può notare l'impatto della modifica del tipo di rendering dal punto di vista qualitativo.

Figura 4.24: *differenza tra rendering standard e ad alta risoluzione.*

In caso l'applicativo rilevi un utilizzo della memoria, da parte di sé stesso, maggiore di 1/3 di quella complessiva, verrà automaticamente e obbligatoriamente abbassata la qualità grafica a livello standard. Stesso comportamento verrà attuato nel caso in cui il numero di oggetti totali disposti su un singolo diagramma sia superiore a 100.

Sempre nel caso in cui l'utilizzo della memoria sia superiore ad 1/3 di quella complessiva, il numero massimo di livelli undo/redo memorizzati viene portato a 5. In condizioni normali di utilizzo della memoria, invece, il numero di undo/redo disponibili può raggiungere un massimo di 50. Inoltre l'utente, a sua discrezione, può decidere di liberare in qualunque momento la memoria occupata dai livelli undo/redo utilizzando la funzionalità *flush memory* (svuota memoria), disponibile nel menu *Edit* (figura 4.20).

L'utilizzatore viene sempre avvisato, circa gli automatismi di ottimizzazione, per mezzo di una notifica che appare nella barra di stato e che rimane presente fino a quando la condizione di allerta persiste.

Infine, l'ultima importante ottimizzazione che è stata eseguita, è quella di evitare più di una istanza contemporanea di SER-DE; questo comporta un notevole risparmio di memoria poiché un'istanza appena inizializzata di tutto l'editor richiede circa 50 MByte di memoria. D'altra parte l'interfaccia a schede di SER-DE consente di aprire e lavorare contemporaneamente su di un numero teoricamente infinito di diagrammi, limitato solo dalle caratteristiche fisiche del sistema hardware in uso, pertanto l'apertura di più istanze non solo non è utile ma è addirittura controproducente.

Capitolo 5
RISULTATI OTTENUTI

In questo capitolo verranno mostrati alcuni esempi notevoli di diagrammi che è possibile produrre utilizzando SER-DE. Saranno prese in considerazione, in particolare, porzioni di diagramma, al fine d'illustrare meglio le potenzialità specifiche dell'applicativo.

In figura 5.1 è mostrata la più semplice tipologia di diagramma E-R realizzabile attraverso SER-DE: un'associazione, detta binaria o di grado 2, che rappresenta due istanze di entità coinvolte in un'istanza dell'associazione.
In figura 5.2 è mostrato un esempio di associazione ternaria o di grado 3, nella quale tre istanze di entità sono coinvolte in un'istanza dell'associazione.
In figura 5.3 troviamo un esempio di associazione multipla, dove sono rappresentate più associazioni, di diverso significato, tra diverse entità.
L'ultima tipologia di associazione rappresentabile con SER-DE è l'associazione ad anello dove le istanze di una stessa entità vengono messe in relazione tra loro; questa è rappresentata in figura 5.4.

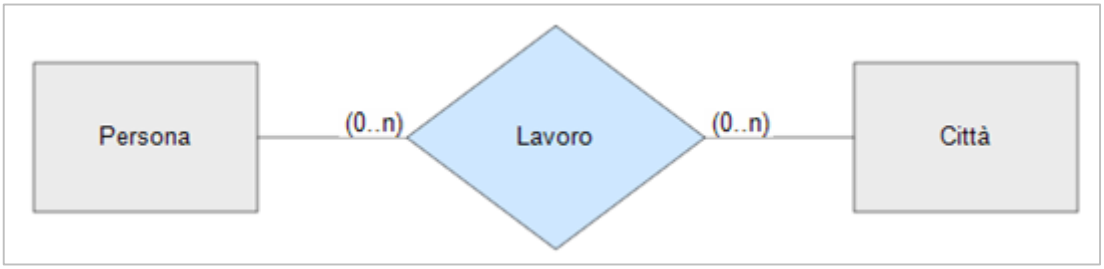


Figura 5.1: *associazione binaria.*

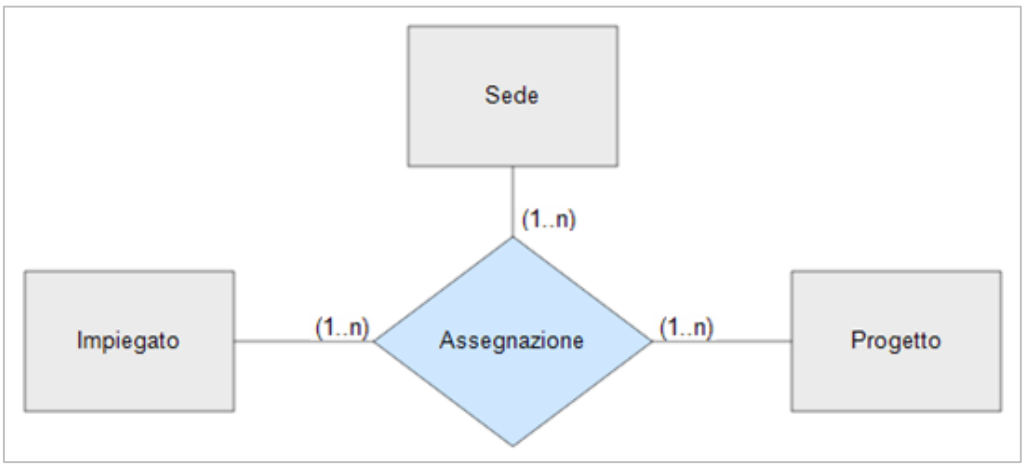


Figura 5.2: *associazione ternaria.*

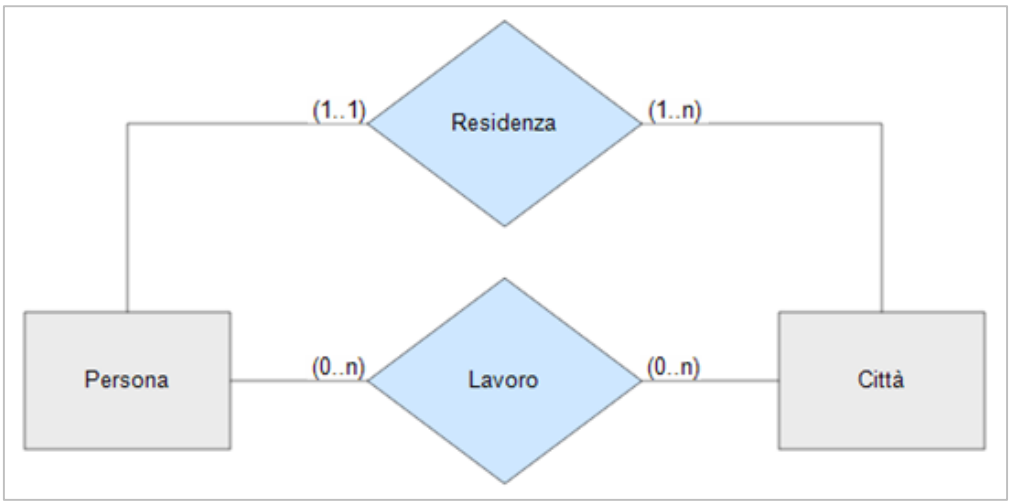


Figura 5.3: *associazione multipla tra stesse entità.*

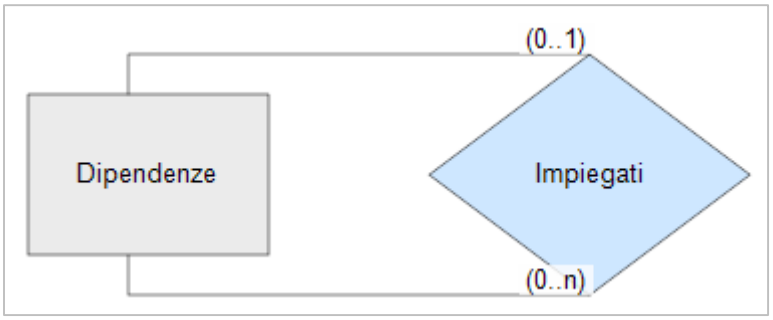


Figura 5.4: *associazione ad anello.*

Per quanto riguarda gli attributi, utilizzando SER-DE è possibile definire gli attributi semplici delle entità (figura 5.5) e delle associazioni (figura 5.6).
Inoltre è possibile specificare gli attributi composti così come i relativi sotto-attributi (figura 5.7).

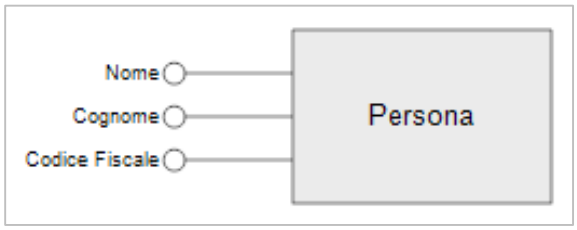


Figura 5.5: *attributi di un'entità.*

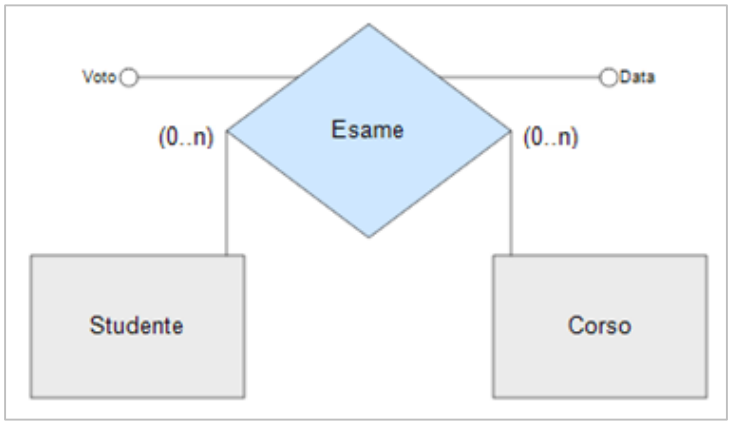


Figura 5.6: *attributi di un'associazione.*

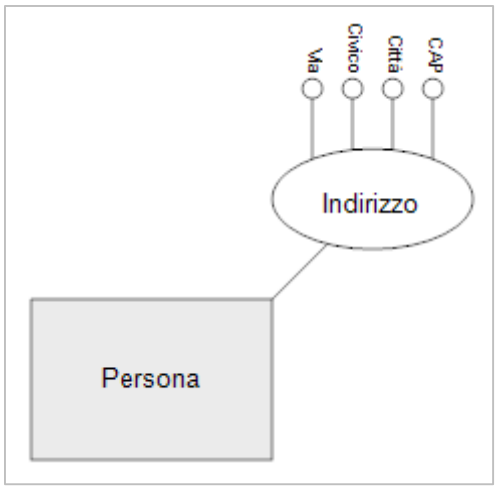


Figura 5.7: *attributo composto.*

Per ogni associazione è possibile specificare i vincoli di cardinalità (figura 5.8) così come per ogni singolo attributo (figura 5.9); tali vincoli, a discrezione dell'utente, potranno o meno essere visualizzati nel diagramma.

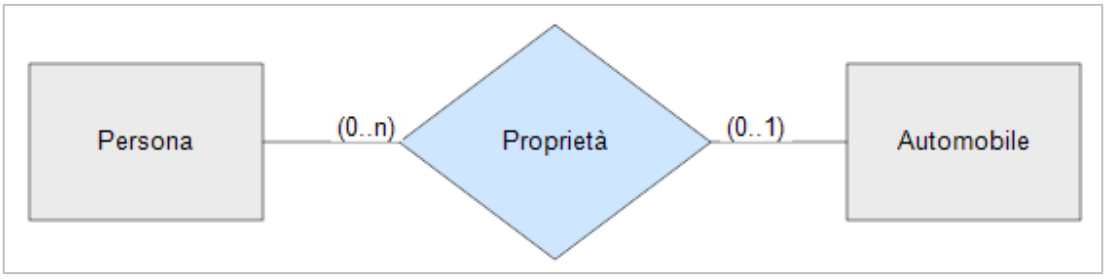


Figura 5.8: *vincoli di cardinalità delle associazioni*

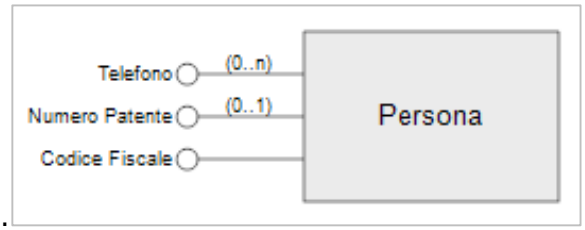


Figura 5.9: *vincoli di cardinalità degli attributi.*

Altra caratteristica fondamentale rappresentabile con SER-DE sono gli identificatori che possono essere interni semplici (figura 5.10), interni e composti (figura 5.11), esterni e composti misti (figura 5.12) ed esterni e composti puri (figura 5.13).

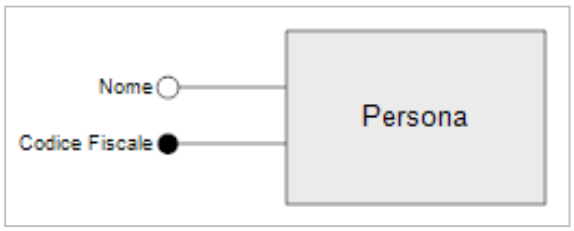


Figura 5.10: *identificatore interno semplice.*

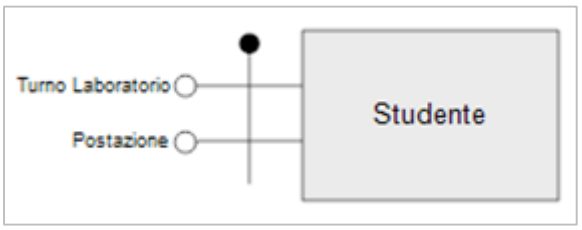


Figura 5.11: *identificatore interno e composto.*

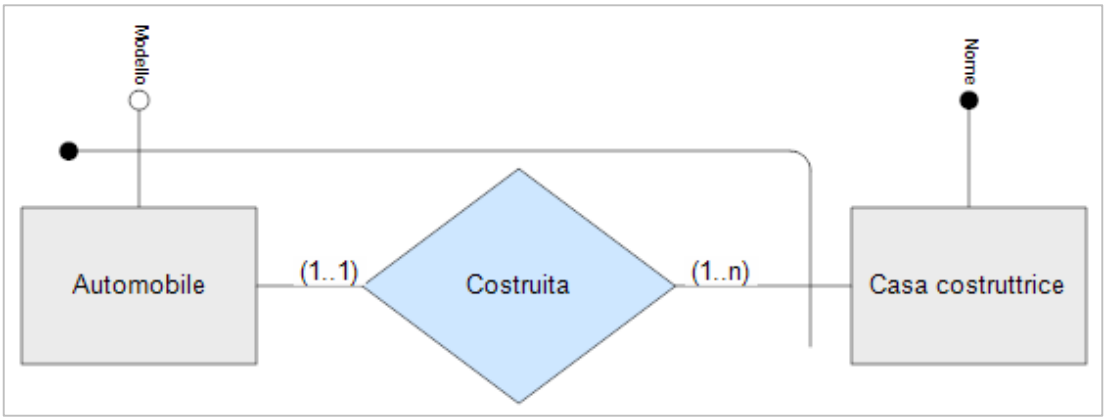


Figura 5.12: *identificatore esterno e composto (misto).*

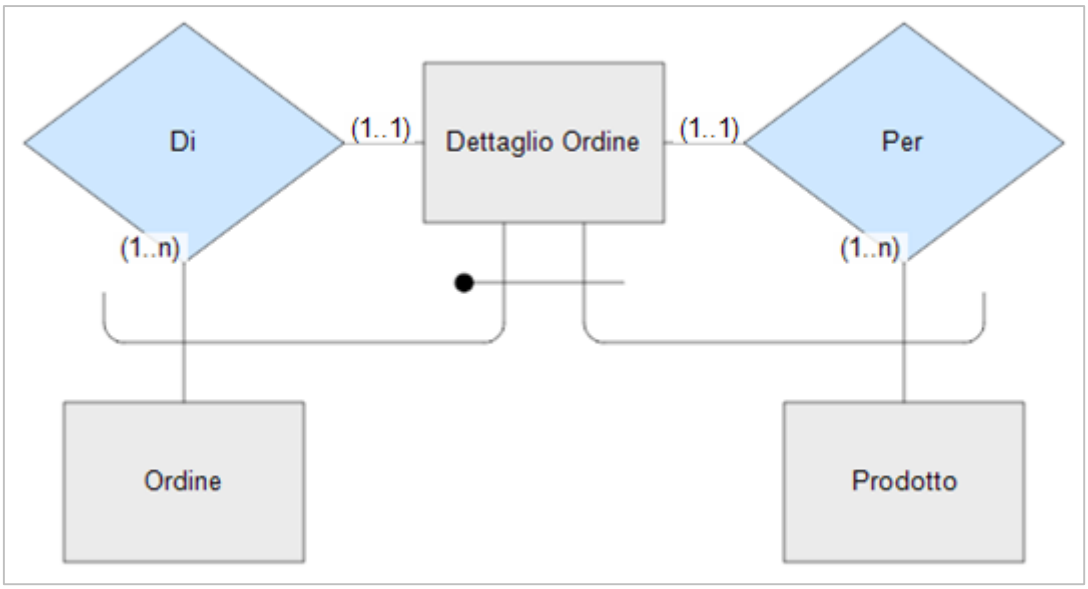


Figura 5.13: *identificatore esterno e composto (puro).*

SER-DE offre anche la possibilità di rappresentare qualunque tipo di generalizzazione o di subset.

In figura 5.14 troviamo un esempio di generalizzazione con copertura totale ed esclusiva (t,e = *total-exclusive*).

In figura 5.15 abbiamo invece una generalizzazione la cui copertura è parziale ed esclusiva (p,e = *partial-exclusive*).

In figura 5.16 abbiamo un esempio di generalizzazione con copertura totale e sovrapposta (t, o = *total-overlapped*).

In figura 5.17 è rappresentata una generalizzazione con copertura parziale e sovrapposta (p, o = *partial-overlapped*).

L'etichetta di testo relativa alla copertura, a prescindere da quale l'utente abbia scelto, può essere o meno visualizzata sul diagramma a discrezione dell'utente.

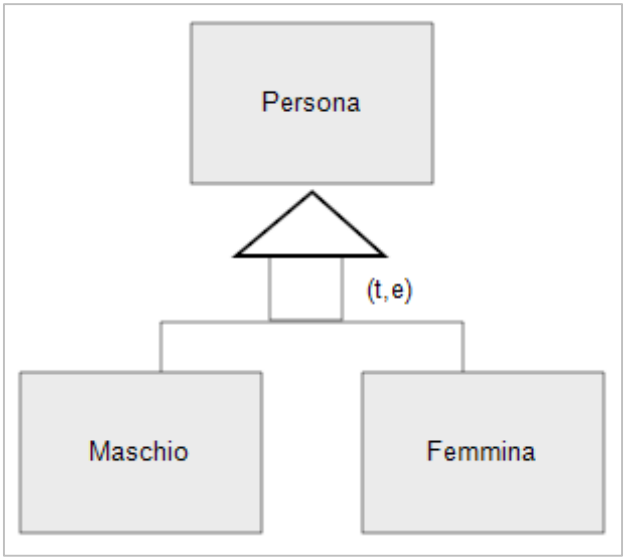


Figura 5.14: *generalizzazione con copertura totale ed esclusiva.*

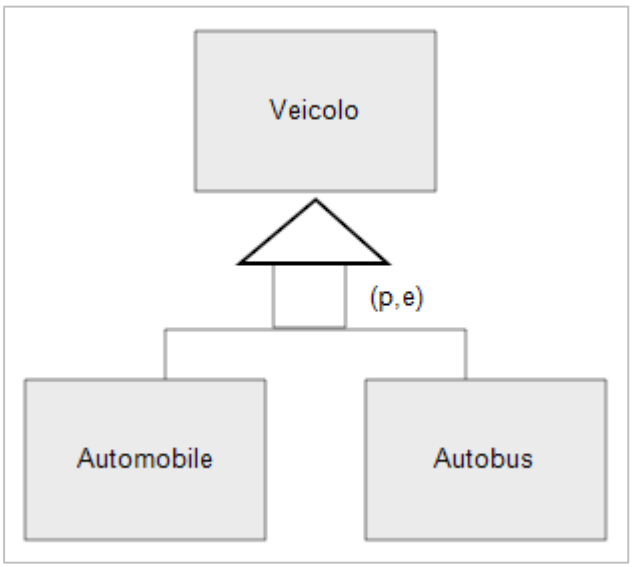


Figura 5.15: *generalizzazione con copertura parziale ed esclusiva.*

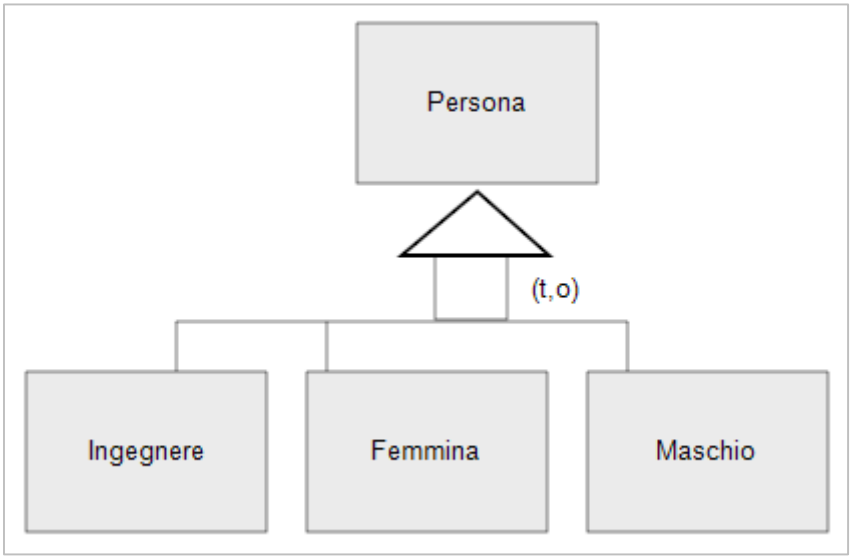


Figura 5.16: *generalizzazione con copertura totale e sovrapposta.*

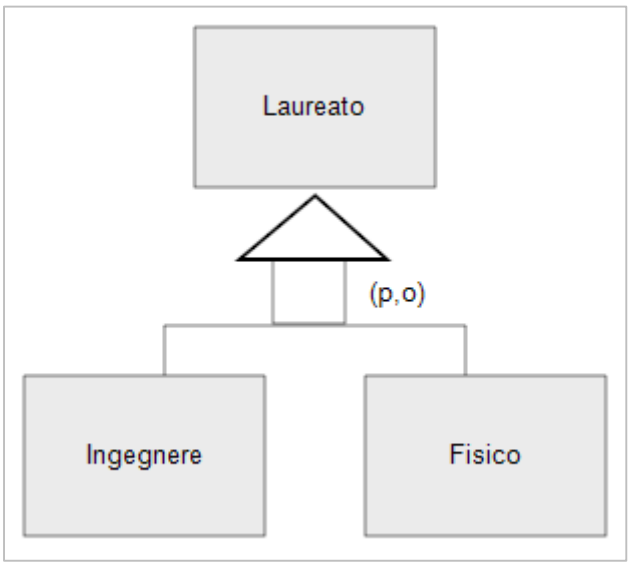


Figura 5.17: *generalizzazione con copertura parziale e sovrapposta.*

Tra gli esempi notevoli di porzioni di diagramma che è possibile rappresentare con SER-DE troviamo infine il subset, rappresentato in figura 5.18.

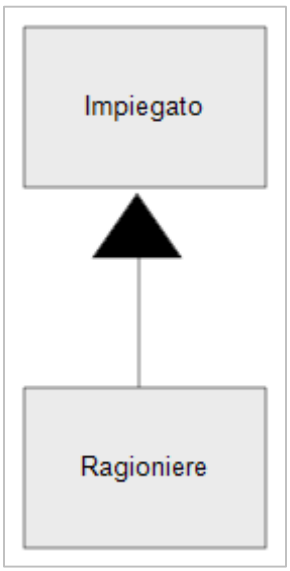


Figura 5.18: *subset.*

Capitolo 6
CONCLUSIONI E SVILUPPI FUTURI

L'applicativo sviluppato è ricco di funzionalità ed è stato pensato con l'obiettivo primario di consentire la creazione agile di diagrammi E-R in notazione estesa. È possibile realizzare diagrammi completi, in grado di descrivere pressoché qualunque scenario, come si può notare nell'esempio di figura 6.1.

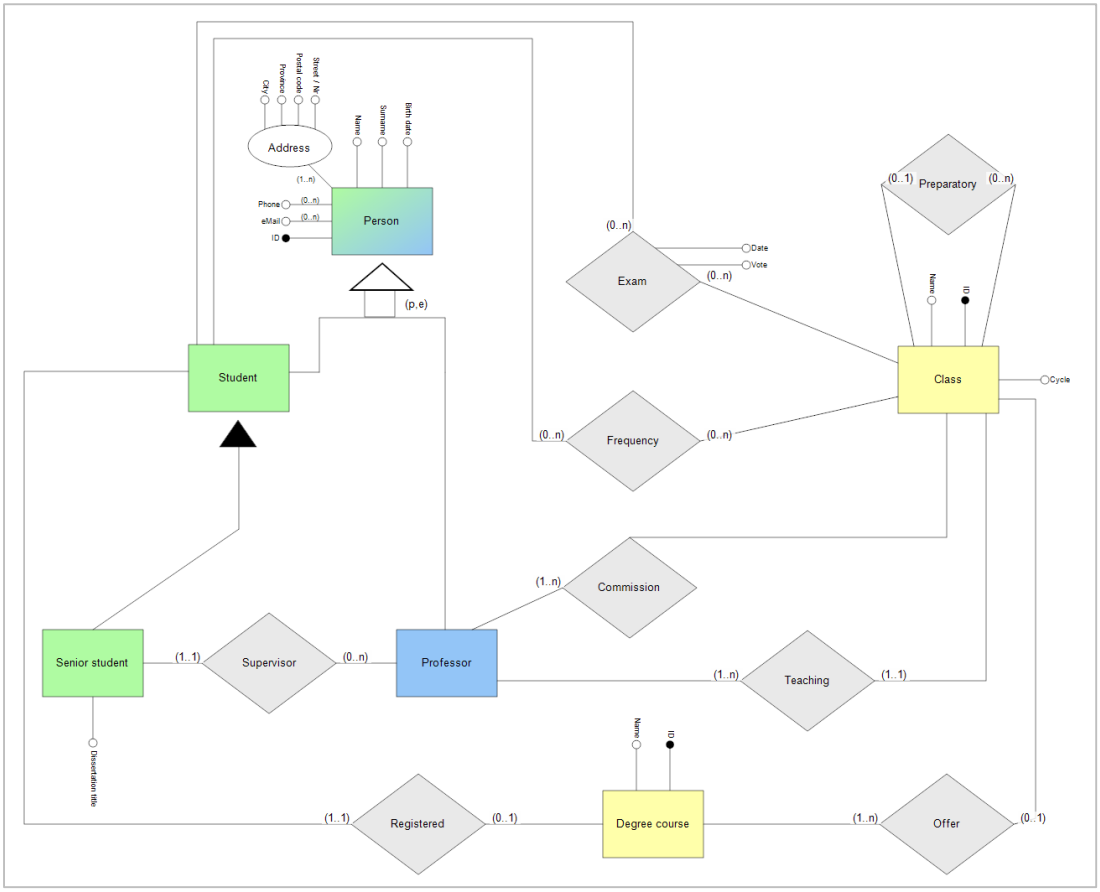


Figura 6.1: *diagramma E-R completo.*

L'obiettivo è stato senza dubbio raggiunto ma esistono alcuni punti che possono essere migliorati e altri che possono essere estesi; inoltre è possibile aggiungere ancora tante funzionalità avanzate, le quali renderebbero SER-DE davvero utile, non solo in fase di progettazione concettuale pura ma anche in tutte le fasi successive, compresa la creazione fisica della base dati.
Di seguito sono elencate le azioni più importanti da mettere in campo e le funzionalità da implementare per migliorare e completare SER-DE.

- Supporto multi-piattaforma: al momento l'applicativo è compatibile con tutti i sistemi operativi *Microsoft* Windows sui quali è possibile installare il .NET framework 2.0 (Windows 2000 e successivi); in realtà, per utilizzare tutte le funzionalità, in particolare quella di stampa avanzata, è necessario utilizzare almeno Windows XP.
  Il supporto è, pertanto, già decisamente ampio ed è in grado di coprire oltre il 90% di tutti i PC presenti al mondo. Ciò taglia fuori, purtroppo, una fetta di utenti in netto aumento negli ultimi anni, ossia chi utilizza sistemi operativi basati su kernel Linux e sistemi MacOS.
  Il porting dell'applicativo a questi sistemi è possibile e dovrebbe essere la prima cosa da fare prima di proseguire con l'implementazione di nuove funzionalità; a tal fine si può utilizzare un progetto, sempre open source, assolutamente maturo e stabile: *Mono Project* [MPJ]. Tale framework, ed il relativo runtime, sono compatibili con il framework .NET originale e consentirebbero, con qualche accorgimento, di utilizzare SER-DE praticamente in tutti gli ambienti operativi per PC esistenti. Gli accorgimenti sono relativi al fatto che, al momento, SER-DE utilizza funzionalità specifiche dei sistemi operativi Windows: la stampa WIA, la libreria grafica GDI+, il registro di sistema

e le query WMI [17] (utilizzate per ottenere informazioni fisiche sul sistema in uso). Isolando queste porzioni di codice e creando uno strato software in grado di astrarre da queste specificità si potrebbe, senza troppe complicazioni, eseguire il suddetto porting. Anche la sostituzione della libreria GDI+, nonostante di primo acchito possa apparire complessa, in realtà non lo è poiché con il progetto *Mono* è compresa anche la libreria *libgdiplus* [LGP], compatibile a livello di API[18] con quella originale sviluppata da *Microsoft*.

- Refactoring ed ottimizzazioni: durante la fase di porting, o subito dopo averla completata, sarebbe particolarmente utile eseguire un ulteriore ciclo di refactoring per individuare e migliorare i punti di codice più complessi e poco leggibili ed ottimizzare tutte quelle porzioni particolarmente lente. Ad esempio, al momento, per decidere se un oggetto posizionato su canvas deve essere evidenziato o selezionato, si procede valutando, per ogni singolo oggetto, se la posizione del cursore del mouse interseca l'area occupata dall'oggetto stesso. In realtà questa metodologia ha già subito dei cicli di refactoring che ne hanno migliorato tantissimo le prestazioni: per aumentare la probabilità di trovare l'oggetto da evidenziare si procede in funzione del posizionamento degli oggetti sull'asse delle Z e, nel momento in cui il primo viene trovato, s'interrompe la ricerca.
  Per ottimizzare drasticamente questa ricerca si potrebbe pensare di introdurre l'utilizzo di una struttura dati ad accesso veloce, come una tabella di hash, le cui chiavi potrebbero essere calcolate utilizzando le coordinate e l'ingombro dei singoli oggetti posizionati su canvas.

[17] Windows Management Instrumentation [WMI]
[18] Application Programming Interface

- Estensioni controlli di validità: al momento SER-DE implementa già molti controlli sulle operazioni effettuate dall'utente, al fine di prevenire gli errori concettuali più gravi. Ad esempio non è possibile collegare, tramite un oggetto connessione, direttamente due entità o due relazioni. Inoltre non è possibile specificare un'entità come padre di due differenti insiemi di generalizzazioni, oppure non si può generalizzare una relazione. Estendendo questi controlli, ad esempio, anche agli identificatori, interni o esterni, si potrebbero evitare tutte le anomalie che derivano dalla definizione di chiavi errate o inconsistenti; con tali potenzialità l'applicativo acquisterebbe probabilmente un interesse di rilievo, anche al di fuori del mondo accademico.
  È importante notare che l'architettura attuale del sistema, per quanto senza dubbio migliorabile, grazie alla modularità con la quale è stata realizzata consente già l'aggiunta di tutti questi controlli con modifiche mirate, facilmente integrabili nel codice esistente.

- Domini degli attributi: la possibilità di specificare i domini degli attributi, ossia il tipo e la dimensione dei dati (i.e. il campo *nome utente* è composto da una stringa di massimo 50 caratteri), è un passo chiave per gettare le fondamenta di tutte le funzionalità avanzate di forward engineering che abbiamo visto essere presenti in tutti gli applicativi, commerciali e non, di riferimento.

- Collegamento dinamico tra file: i file .serx rappresentano dei diagrammi a sé stanti, senza alcun collegamento con altri diagrammi o porzioni di essi.
  Una funzionalità aggiuntiva molto interessante potrebbe essere quella di introdurre il concetto di pacchettizzazione, ossia l'inserimento di un intero diagramma, o di una parte di esso, all'interno di un pacchetto.

Tale pacchetto dovrebbe essere quindi riutilizzabile da un altro diagramma (contenuto in un altro file .serx); per mezzo di questa funzionalità si potrebbero affrontare problemi di dimensioni enormi, suddividendoli in più file, pur mantenendo la coerenza logica data dal collegamento tra gli stessi.

- Forward/Reverse engineering: la possibilità di creare direttamente da SER-DE il file DDL contenente la struttura del database, darebbe agli utenti la possibilità di gestire al meglio sia la fase di progettazione che quella di creazione fisica del database.
  Ugualmente utile sarebbe la funzionalità inversa, ossia quella in grado di creare il diagramma E-R a partire da un database già esistente.
  È inutile dire che tali funzionalità sono senza dubbio le più difficili da implementare, anche perché esistono innumerevoli specificità dei DBMS, spesso in continua evoluzione, da tenere in considerazione.

- Documentazione strutturata: sarebbe molto utile migliorare la sezione di SER-DE dedicata alla documentazione, per poterla strutturare meglio e per poter aggiungere note con una granularità maggiore, anziché limitarsi a note generiche inserite a livello di diagramma. Con tale struttura verrebbe favorita anche la condivisione e la leggibilità della documentazione stessa che potrebbe, ad esempio, essere esportata direttamente in formato HTML.

- File .serx con struttura XML: ultima importante modifica è quella relativa al formato dei file creati e gestiti dall'applicativo. Al momento, per ragioni di semplicità, si fa uso di dati binari serializzati direttamente su un file con estensione .serx.

Sarebbe decisamente meglio, al fine di creare un formato standard e condivisibile con altri applicativi, utilizzare i potenti oggetti messi a disposizione dal framework .NET per serializzare/deserializzare su/da file dei flussi XML contenenti i diagrammi E-R.
Per questa modifica non è richiesta alcuna ristrutturazione o ridefinizione degli oggetti poiché è sufficiente decorare, per mezzo di attributi specifici, le classi relative agli oggetti da serializzare e le proprietà degli stessi.

È fondamentale ricordare che SER-DE è rilasciato sotto licenza di tipo open source, in particolare la scelta è ricaduta sulla GPL[19] versione 3. In questo modo, chiunque volesse partecipare al miglioramento dell'applicativo, sia a livello accademico che non, potrà farlo, seppur con l'obbligo di rilasciare, sempre sotto la stessa licenza, il codice prodotto.

[19] Gnu General Public License [GPL]

BIBLIOGRAFIA

[Cod70] Edgar F. Codd, “A Relational Model of Data for Large Shared Data Banks”, Giugno 1970.

[Che76] Peter Chen, “The Entity-Relationship Model - Toward a Unified View of Data”, Marzo 1976.

[Che02] Peter Chen, “Entity-Relationship Modeling - Historical Events, Future Trends, and Lessons Learned”, Software Pioneers (9783540430810), 2002, pp. 296-310.

[Tyf86] Toby J. Teorey, Dongqing Yang, James P. Fry, “A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model”, Computing Surveys, Giugno 1986.

[Bro93] Robert G. Brown, “Integration Definition for Information Modeling (IDEF1X)”, Federal Information

Processing Standards Publication, Dicembre 1993.

[Bac69] Charles W. Bachman, “Data Structure Diagrams”, ACM SIGMIS Database, Estate 1969, pp. 4-10.

[Hal01] Terry Halpin, “Information Modeling and Relational Databases” (9780080508665), 2001, pp. 327-331.

[Bar90] Richard Barker, “Case*Method: Entity Relationship Modelling” (9780201416961), 1990.

[Eve87] Gordon C. Everest, “Users do Logical Database Design”, Aprile 1987.

[RobCor09] Peter Rob, Carlos Coronel, “Database Systems - Design, Implementation & Management” (9781408030813), 2009.

Sitografia

[ERS] ER/Studio, (http://www.embarcadero.com/products/er-studio).

[ERW] Erwin, (http://erwin.com).

[TDM] Toad Data Modeler, (http://www.quest.com/toad-data-modeler).

[SPD] Sybase Power Designer, (http://www.sybase.it/products/modelingdevelopment).

[DBM] DB-Main, (http://www.db-main.eu).

[MSW] MySQL Workbench, (http://dev.mysql.com/downloads/tools).

[ODM] Oracle Data Modeler, (http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html).

[AUM] Altova UModel, (http://www.altova.com/umodel.html).

[MVS] Microsoft Visual Studio, (http://www.microsoft.com/visualstudio).

[SHF] Sandcastle Help File Builder, (http://shfb.codeplex.com/).

[INN] Inno Setup, (http://www.jrsoftware.org/isinfo.php).

[MWD] Microsoft Word, (http://office.microsoft.com).

[DBG] Double Buffered Graphics - MSDN, (http://msdn.microsoft.com/en-us/library/b367a457.aspx).

[GDI] GDI+ - Graphics Device Interface, (http://msdn.microsoft.com/en-us/library/windows/desktop/ms533798(v=vs.85).aspx).

[OGL] OpenGL - Open Graphics Library, (http://www.opengl.org).

[DTX] DirectX, (http://msdn.microsoft.com/en-us/library/windows/desktop/ee663274(v=vs.85).aspx).

[L4N] log4net, (http://logging.apache.org/log4net/).

[WIA] WIA - Windows Image Acquisition, (http://msdn.microsoft.com/en-us/library/windows/desktop/ms630368(v=vs.85).aspx).

[MPJ] Mono Project, (http://www.mono-project.com/Main_Page).

[WMI] WMI - Windows Management Instrumentation, (http://msdn.microsoft.com/en-us/library/aa394582(v=vs.85).aspx).

[LGP] Libgdiplus, (http://www.mono-project.com/Libgdiplus).

[GPL] GPL - Gnu General Public License, (http://www.gnu.org/copyleft/gpl.html).