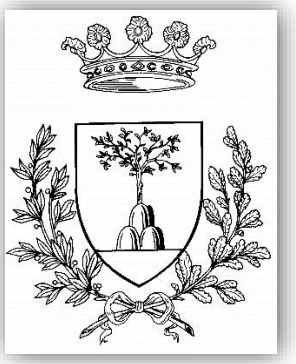


UNIVERSITÀ DEGLI STUDI DI FERRARA

Dipartimento di Ingegneria

Corso di Laurea in

Ingegneria Informatica e dell'Automazione

DOCUMENTAZIONE BASI DI DATI:

Customer Relationship Management

A cura di:

Docente:
Ing. Denis Ferraretti

INDICE

CAPITOLO 1: *Specifiche di Progetto*

1.1 *Introduzione documentazione*

Nelle seguenti pagine si riporta la documentazione relativa ad un esempio di progettazione e l'implementazione di un sistema di basi di dati e ad un prototipo di applicazione web, realizzate tramite PHP/HTML che si poggia su tale sistema.
La situazione che si è ipotizzata è quella di un'azienda, fornitrice di servizi di consulenza ad altre aziende, che voglia commissionare un applicativo web per la realizzazione di un CMR (Customer Relationship Management).

1.2 *Elenco specifiche*

Vengono qui elencate quelle che sono le caratteristiche che l'applicativo deve garantire:

1) Possibilità di visualizzare il singolo cliente (con i dati dell'azienda e del referente aziendale) con tutti i dettagli e le caratteristiche, l'elenco delle note cliente e l'elenco dei servizi di consulenza acquistati. Ogni cliente è assegnato ad un utente dell'applicativo.

2) Possibilità di visualizzare l'elenco clienti per utente a cui è assegnato.

3) Gestione delle note cliente: ogni volta che un cliente viene contattato deve essere possibile registrare/modificare/cancellare una o più note relative alla conversazione avvenuta e dell'utente che l'ha registrata.

4) Gestione delle opportunità: per ogni cliente deve essere possibile inserire una nuova opportunità, cioè una proposta commerciale.

5) Gestione degli appuntamenti: deve essere possibile inserire un appuntamento con una nota descrittiva, una data e un cliente a cui è riferito.

6) Visualizzazione dell'agenda degli appuntamenti per quell'utente.

7) Possibilità di inserire nuovi servizi di consulenza.

8) Possibilità di inserire nuovi clienti.

9) Possibilità di inserire nuovi utenti dell'applicativo web.

Il sistema prevede che le categorie di utenti sia così rappresentata:

- **Clienti**, possono visualizzare i propri appuntamenti, le proprie note, le proprie opportunità e i servizi di consulenza acquistati e acquistarne di nuovi.

- **Utenti**, protagonisti dell'applicativo, possono effettuare i punti dal 1 al 6.
- **Amministratori**, possono effettuare i punti 7, 9.

CAPITOLO 2: *Progettazione del Diagramma ER*

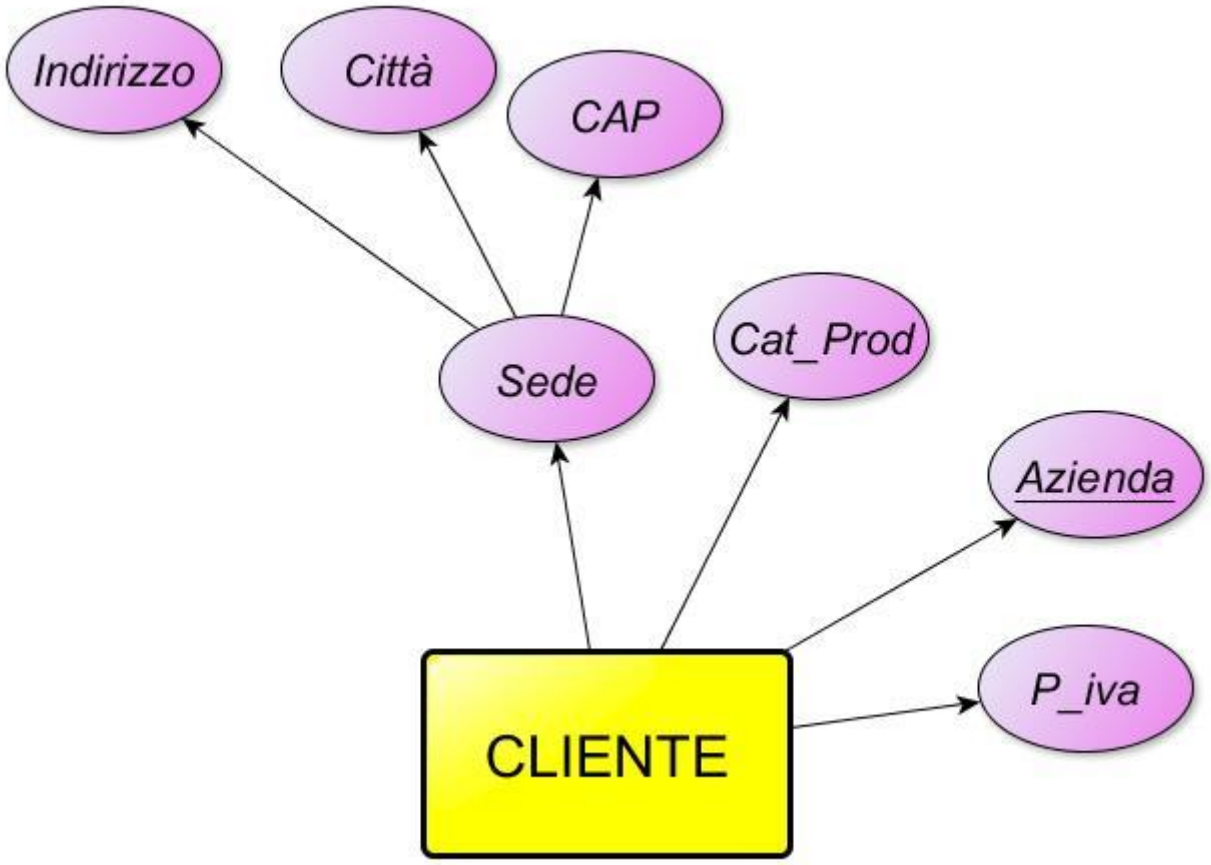
2.1 *Passaggio dal minimondo allo schema ER*

Nell'attività di progettazione e implementazione di una base di dati, una volta superata la prima fase che consiste nella raccolta e analisi dei requisiti la seconda fase consiste nel passaggio da specifiche in linguaggio naturale alla progettazione concettuale del database.
La progettazione concettuale permette di focalizzare l'attenzione sul significato che avranno i vari dati memorizzati nella base di dati, senza scendere in dettagli troppo tecnici che in una prima fase di progetto risultano superflui. Il risultato, quindi, della progettazione concettuale è un diagramma che mette in risalto le entità, proprietà e associazioni in gioco, comprensibile anche per i non addetti ai lavori.
Esistono vari tipi di modello dei dati concettuale che ci permettono di assolvere i compiti sopracitati, quello più diffuso ed anche quello utilizzato in questo contesto è il Diagramma ER.

2.2 *Dettagli sulle entità*

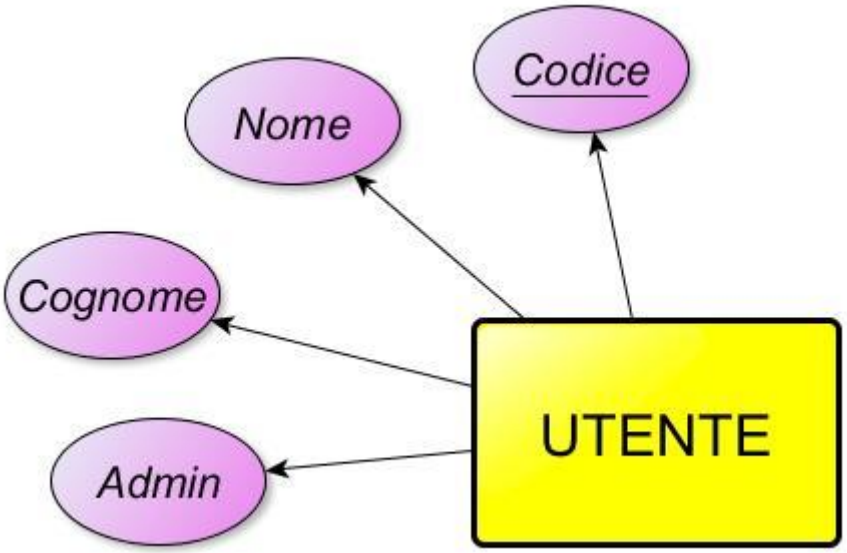
Si è scelto di creare le entità cliente, utente, servizio, referente, note_cliente, proposta e appuntamento in quanto esse rappresentato quelle "cose", "oggetti" o concetti del mondo reale che hanno esistenza indipendente.

CLIENTE



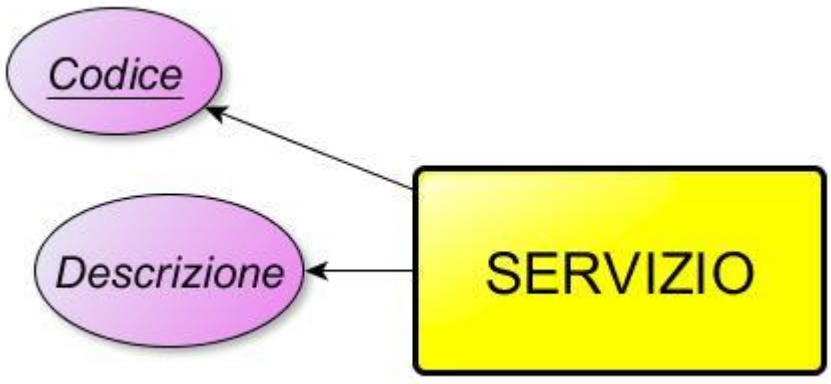
Il tipo di entità CLIENTE, la prima delle 3 entità "**cuore**", raffigura quelle che sono le aziende clienti. Essa è composta da 4 attributi, dove in particolare Cat_Prod identifica categoria merceologica dei prodotti dell'azienda, mentre Sede è un attributo composto, formato da 3 attributi semplici (Indirizzo, Citta, Cap). Gli attributi Azienda e P_Iva sono due chiavi distinte.

UTENTE



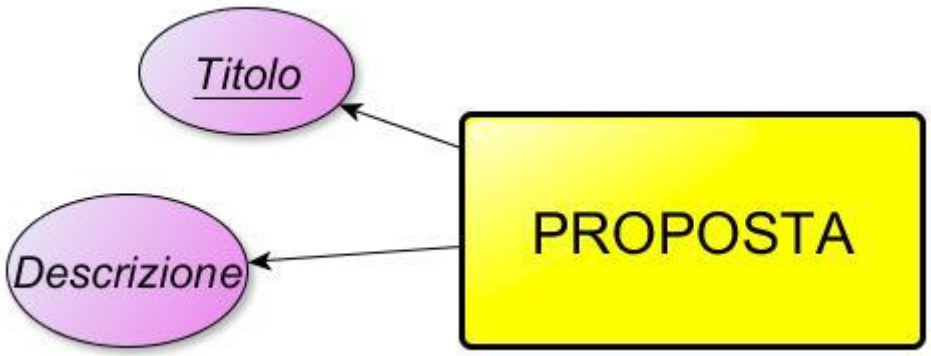
Il tipo di entità UTENTE rappresenta i fruitori dell'applicazione. Essa è contraddistinto da 4 attributi, Codice, Nome, Cognome ed Admin. Codice è la chiave che permette di distinguere i vari utenti utilizzatori dell'applicazione, mentre Admin è un campo che permette di distinguere gli utenti amministratori da quelli non.

SERVIZIO



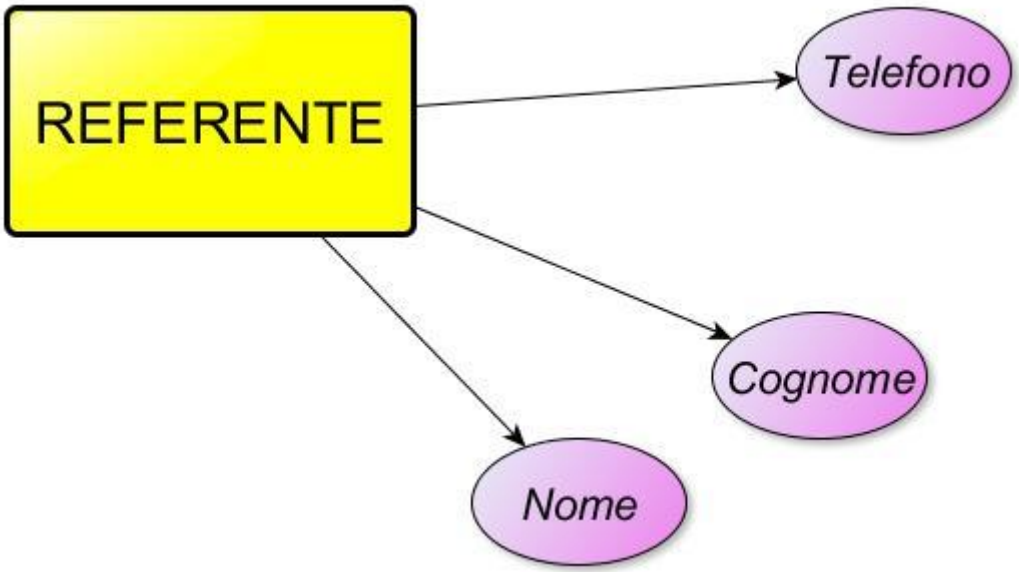
Il tipo di entità SERVIZIO, l'ultima delle 3 entità **"cuore"**, rappresenta il servizio di consulenza offerto dall'azienda. Essa è caratterizzata da soli due attributi: Codice è l'attributo chiave mentre Descrizione è un attributo che permette di descrivere la tipologia di servizio offerto.

PROPOSTA



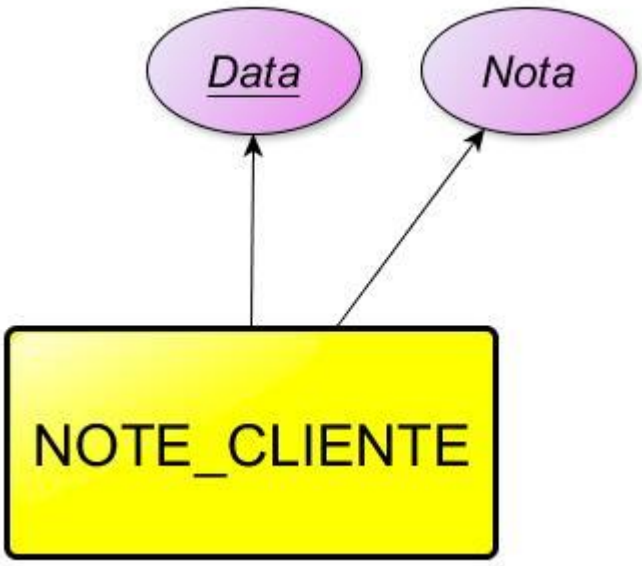
Il tipo di entità PROPOSTA modella una proposta commerciale fatta da un utente ad un cliente. Essa è formata da due attributi: Titolo è una chiave che differenzia le varie proposte, mentre Descrizione è un campo aggiuntivo che dà la possibilità di aggiungere ulteriori informazioni relative alla proposta. Si è scelto di usare Titolo come chiave e di non usare un ulteriore attributo (es. Codice) che fungesse da chiavi poiché si è supposto che ogni proposta abbia già di suo un nome differente, inoltre si è pensato di progettare PROPOSTA come entità forte, o meglio non debole, affinché sia possibile creare una nuova proposta senza doverla obbligatoriamente legare a qualche cliente.

REFERENTE



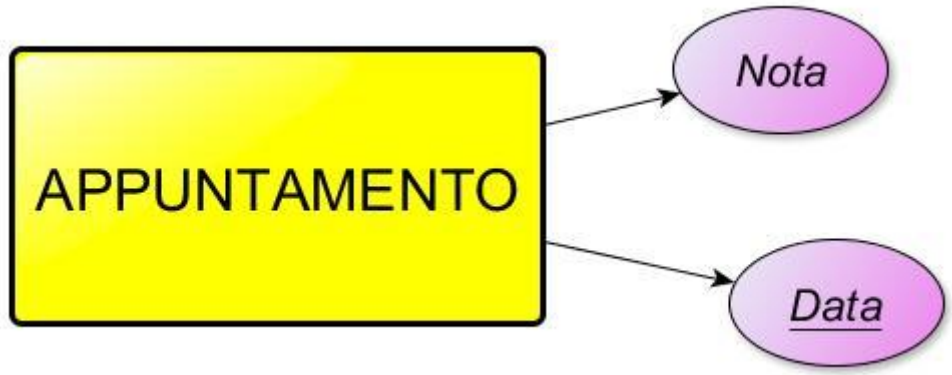
Il tipo di entità REFERENTE descrive il referente per ogni azienda. Essa è costituita da 3 attributi semplici Nome, Cognome e Telefono. Si è pensato di crearla come entità debole per due motivi principalmente: uno, un referente senza un'azienda non avrebbe senso, in quanto, in generale, il referente è un impiegato che lavora per quella data azienda, due, poiché ogni azienda (entità CLIENTE) ha un solo referente ed un referente a sua volta lavora per una sola azienda, era di conseguenza, inutile introdurre un ulteriore attributo chiave per il tipo di entità REFERENTE.

NOTE_CLIENTE



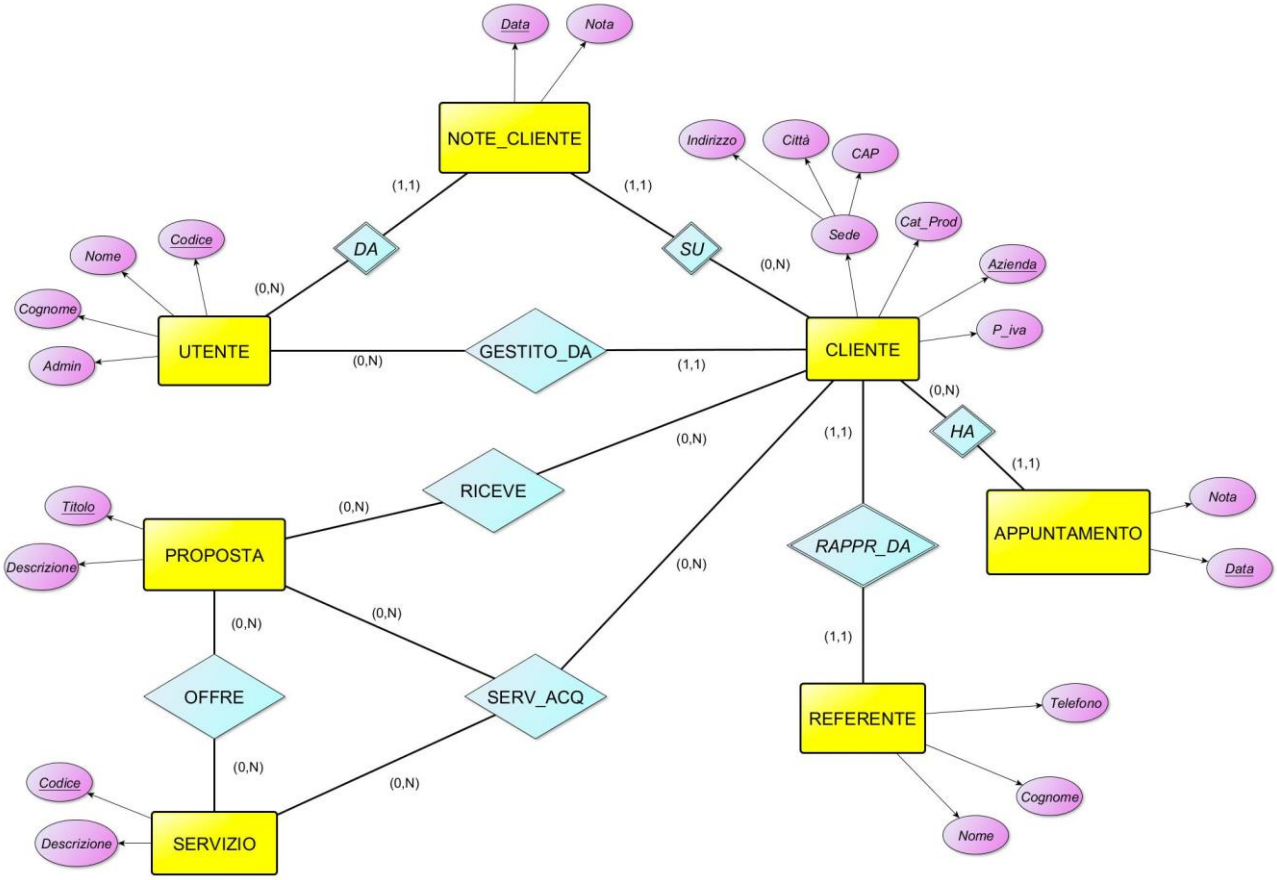
Il tipo di entità NOTE_CLIENTE simboleggia le note che un utente inserisce/modifica su un cliente ogni volta che viene contattato per memorizzare delle note relative alla conversazione avvenuta. Essa è formata da due attributi: Data è la chiave parziale (Partial Key),mentre Nota dà la possibilità di specificare la nota della conversazione. NOTE_CLIENTE è stata realizzata come entità debole, dipendente da CLIENTE e UTENTE, in quanto una nota senza un cliente ed un utente non avrebbe nessuna consistenza, in più è stata usata Data come chiave parziale per differenziare le varie note associate ad uno stesso cliente, supponendo che non ci possano essere due note diverse con la stessa data, poiché vorrebbero dire che un cliente ha avuto due conversazioni diverse nello stesso tempo, che è una cosa impossibile.

APPUNTAMENTO



Il tipo di entità APPUNTAMENTO modella gli appuntamenti ai quali un cliente è invitato a partecipare. Essa è composta di due attributi: Data, chiave parziale, e Nota che include ulteriori informazioni relative all'appuntamento. APPUNTAMENTO è stata modellata come entità debole perché essa è sempre associata ad un cliente e per distinguere i differenti appunti associati al medesimo cliente basta la sola chiave parziale. Un attributo chiave che rendesse l'entità in questione un entità non debole sarebbe superfluo.

2.3 *Diagramma ER completo*



2.4 Discussione Associazioni

Per quanto riguarda le associazioni il tipo di entità CLIENTE è quella più “attiva” poiché partecipa a 6 associazioni su 8.

ASSOCIAZIONE **SU**
L'associazione SU è un'associazione di 2° grado che coinvolge i tipi di entità CLIENTE e l'entità debole NOTE_CLIENTE. Essa mette in relazione un'azienda con le proprie note ricevute. Il cliente ha il vincolo di partecipazione parziale e una cardinalità pari ad N perché così come potrebbe avere 0 note potrebbe avere anche N, mentre una singola nota ha un vincolo di partecipazione totale in quanto è sempre associata ad uno ed uno solo di cliente, da questo segue anche cardinalità pari ad 1.

ASSOCIAZIONE **SERV_ACQ**
L'associazione SERV_ACQ è un'associazione di 3° grado che coinvolge i tipi di entità CLIENTE, PROPOSTA e SERVIZIO. Essa mette in relazione un'azienda con la propria lista di servizi di consulenza acquistati e in riferimento a quale proposta commerciale. Ogni tipo di entità in gioco ha un vincolo di partecipazione parziale ed una cardinalità pari ad N perché ogni entità può, come non può, appartenere a nessun acquisto, per esempio un cliente nuovo potrebbe non avere ancora effettuato nessun acquisto tra i servizi disponibili per una determinata proposta.

ASSOCIAZIONE **GESTITO_DA**
L'associazione GESTITO_DA è un'associazione di 2° grado che coinvolge i tipi di entità CLIENTE e UTENTE. Essa mette in relazione un'azienda con il proprio utente. Il cliente ha il vincolo di partecipazione totale in quanto deve per forza essere associato e cardinalità 1 poiché può avere al massimo un solo utente, mentre un utente può avere da 0 a N cliente per questo ha un vincolo di partecipazione parziale e una cardinalità pari ad N.

ASSOCIAZIONE **DA**
L'associazione DA è un'associazione di 2° grado che coinvolge i tipi di entità UTENTE e l'entità debole NOTE_CLIENTE. Essa mette in relazione un utente con le note che ha creato. L'utente ha il vincolo di partecipazione parziale e una cardinalità pari ad N perché così come potrebbe aver inserito 0 note potrebbe avere inserite anche N, mentre una singola nota ha un vincolo di partecipazione totale in quanto è sempre associata ad uno ed uno solo di utente, da questo segue anche cardinalità pari ad 1.

ASSOCIAZIONE **RICEVE**
L'associazione RICEVE è un'associazione di 2° grado che coinvolge i tipi di entità CLIENTE e PROPOSTA. Essa mette in relazione un'azienda con le proposte commerciali ricevute. Il cliente ha il vincolo di partecipazione parziale e una cardinalità pari ad N in quanto potrebbe avere 0 proposte, ma anche N, una proposta ha anche essa un vincolo di partecipazione parziale e una cardinalità pari

ad N perché si è supposto che essa possa sia non essere associata a nessun cliente sia essere associata a più clienti, ovvero più clienti possono avere la medesima proposta commerciale.

ASSOCIAZIONE **HA**

L'associazione HA è un'associazione di 2° grado che coinvolge i tipi di entità CLIENTE e l'entità debole APPUNTAMENTO. Essa mette in relazione un'azienda con i propri appuntamenti. Il tipo di entità CLIENTE ha un vincolo di partecipazione parziale e una cardinalità pari a N in quanto, come logico, un'azienda potrebbe avere in agenda da 0 ad N appuntamenti, al contrario un appuntamento deve obbligatoriamente essere associato ad un'azienda, poiché entità debole, e per questo ha un vincolo di partecipazione totale, inoltre ha anche una cardinalità pari ad 1 perché può essere associato al più ad un'azienda, infatti due aziende non possono avere lo stesso appuntamento.

ASSOCIAZIONE **OFFRE**

L'associazione OFFRE è un'associazione di 2° grado che coinvolge i tipi di entità PROPOSTA e SERVIZIO. Essa mette in relazione una proposta con i possibili servizi ad essa associati. Entrambi i tipi di entità hanno un vincolo di partecipazione parziale ed una cardinalità massima pari ad N, in particolare il tipo di entità PROPOSTA ha un vincolo di partecipazione parziale in accordo con le specifiche dove una proposta “può” essere collegata a vari servizi offerti.

ASSOCIAZIONE **RAPPR_DA**

L'associazione RAPPR_DA è un'associazione di 2° grado che coinvolge i tipi di entità CLIENTE e l'entità debole REFERENTE. Essa mette in relazione un'azienda (CLIENTE) con il proprio referente. Entrambe hanno un vincolo di partecipazione totale ed una cardinalità massima pari ad 1 poiché il tipo di entità CLIENTE deve avere, e può avere al massimo, un referente, viceversa un referente, in quanto facente parte di un tipo di entità debole, deve avere, ed ha al massimo in quanto come già detto si suppone un referente non lavori per più aziende, un cliente.

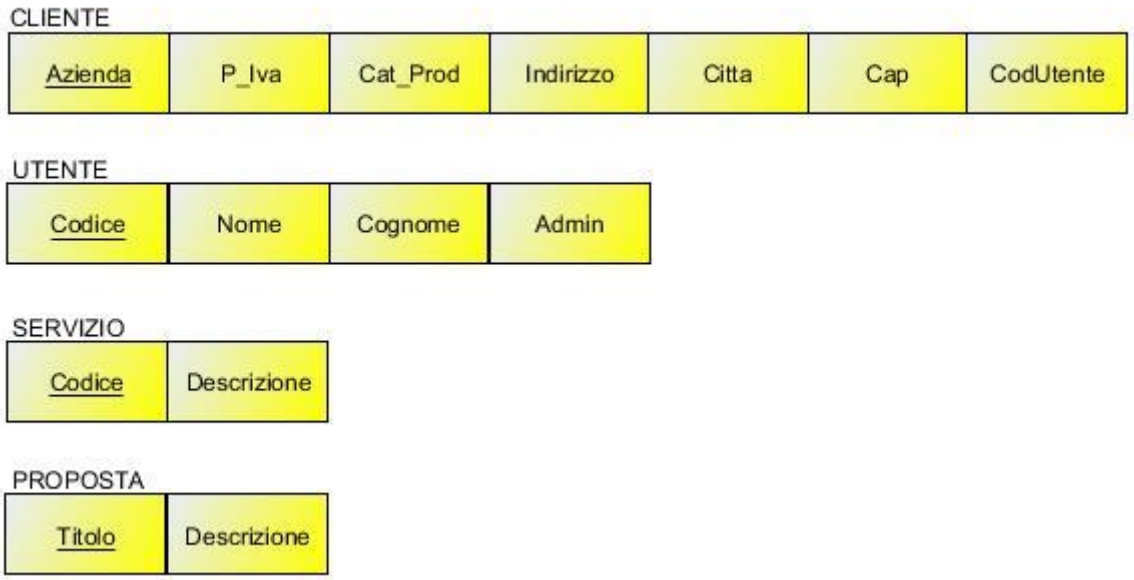
CAPITOLO 3: *Dal Diagramma ER al Modello Relazionale*

3.1 Passaggio da diagramma ER a Modello Relazionale

La terza fase della progettazione di un sistema basi di dati consiste nel passare dalla progettazione concettuale alla progettazione logica.
La progettazione logica permettere di mettere in evidenza il modo in cui i dati sono organizzati, a livello logico, all'interno della base di dati, per questo il modello dei dati logico utilizzato dipende dal tipo di database utilizzato.
Nella realizzazione del modello relazionale, si è seguito un semplice l'algoritmo che permette il passaggio dallo schema concettuale a quello relazionale si fonda sui seguenti passi:

1. traduzione di tipi di entità
2. traduzione di tipi di entità deboli
3. traduzione di tipi di associazioni binarie 1:1
4. traduzione di tipi di associazioni binarie 1:N
5. traduzione di tipi di associazioni binarie N:M
6. traduzione di attributi multivalore
7. traduzioni di tipi di associazione N-arie

PASSO 1: Traduzione di tipi di entità
In questo primo passaggio per ogni entità forte definita nel modello ER si costruisce una nuova relazione R che contiene tutti gli attributi semplici, tralasciando per il momento quelli multivalore, e di quelli composti lo loro scomposizione in attributi semplici. Come chiave primaria per la relazione R viene scelto uno degli attributi chiave del tipo di entità.
Nel diagramma ER in analisi sono presenti 4 tipi di entità forti: CLIENTE, UTENTE, SERVIZIO e PROPOSTA.



PASSO 2: Traduzione di tipi di entità deboli

La traduzione di tipi di entità deboli consiste nella realizzazione di una nuova relazione che ha come attributi tutti gli attributi propri dell'entità debole (con quelli composti opportunamente scomposti) e gli attributi della chiave primaria delle relazioni corrispondenti ai tipi di entità proprietari che fungono da attributi di chiave esterna. La chiave primaria della nuova relazione è data dalla combinazione delle chiavi primarie delle proprietarie e dalla chiave parziale dei tipo di entità debole. In questo modo vengono risolte anche le associazioni che legano un'entità debole all'entità proprietarie.
Nel diagramma ER sono presenti 3 entità deboli: REFERENTE, NOTE_CLIENTE e APPUNTAMENTO.

In questo passaggio è da notare come per ogni nuova relazione si sia inserita come chiave esterna la chiave primaria delle relazione di appartenenza, in più per i tipi

di entità NOTE_CLIENTE e APPUNTAMENTO, data la presenza di un attributo chiave parziale, nel passaggio allo schema di relazione le chiave primarie sono diventate rispettivamente (Azienda, CodUtente, Data) ed (Azienda, Data). Per REFERENTE la chiave esterna Azienda funge anche da chiave primaria.

PASSO 3: Traduzione di tipi di associazioni binarie 1:1
In questo passo ogni tipo di associazione binaria 1:1 nello schema ER viene tradotto utilizzando uno dei seguenti approcci:
- approccio basato su chiave esterna: in una delle due relazioni coinvolte, generalmente quella con vincolo di partecipazione totale per non avere chiavi NULL, si inserisce una chiave esterna alla chiave primaria dell'altra relazione
- approccio basato su fusione delle relazioni: può essere utilizzato solo se entrambe le relazioni hanno un vincolo di partecipazione totale, ovvero stesso numero di righe
- approccio basato su relazione associazione: viene creata una nuova relazione con chiave primaria le chiavi esterne alle relazioni coinvolte

In ognuno dei tre approcci tutti gli attributi dell'associazione in questione vengono inseriti nella relazione di destinazione che contiene la chiave esterna verso l'altra relazione o all'interno della relazione associazione, nel caso di scelta di traduzione basata sul terzo approccio.
Nel caso in studio non sono presenti tipi di associazioni binarie 1:1.

PASSO 4: Traduzione di tipi di associazioni binarie 1:N
Per tradurre un associazione binaria 1:N sono disponibili due tecniche che corrispondono agli approcci 1 e 3 del PASSO 3, con la restrizione che, nel caso in cui si opti per una soluzione basata su chiave esterna (approccio 1), la relazione che possiede la chiave esterna è quella che nell'associazione è posta sul lato-N.
Nel diagramma ER è presente un solo tipo di associazione binaria 1:N: GESTITO_DA.

Si noti che poiché CLIENTE rappresenta la relazione lato-N vi è stato aggiunto un attributo di chiave esterna, CodUtente, alla relazione UTENTE.

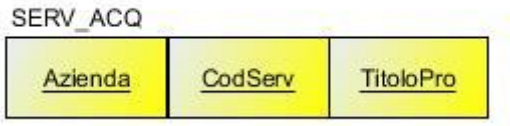
PASSO 5: Traduzione di tipi di associazioni binarie N:M
Per ogni tipo di associazione binaria N:M l'unica soluzione per tradurla in uno schema relazionale è quella di utilizzare una relazione associazione, che come suddetto ha come chiave primaria le chiavi esterne delle relazioni in questione e come restati attributi gli attributi appartenenti all'associazione.
Nel diagramma ER in studio sono presenti due tipi di associazioni binari N:M: RICEVE ed OFFRE.

Come si può osservare sono state create due nuove relazioni associazioni che hanno come chiave primarie le chiavi esterne alle relazioni partecipanti all'associazione. Rispettivamente per RICEVE e OFFRE si hanno come chiavi primarie (Azienda, TitoloPro) e (TitoloPro, CodServ).

PASSO 6: Traduzione di attributi multivalore
In questa fase di traduzione si passa all'analisi e conversione degli attributi multivalore di ogni tipo di entità che nei primi passi vengono volontariamente sorvolati. Questo passaggio porta alla creazione di una nuova relazione per ogni attributo multivalore di ogni tipo di entità. Ogni relazione costituita è caratterizzata da una chiave esterna che fa riferimento alla chiave primaria della relazione corrispondente al tipo di entità di appartenenza e da un attributo semplice (o dagli attributi semplici in caso di attributo multivalore composto) che per ogni valore creerà una nuova tupla. La chiave primaria sarà, quindi, il valore stesso dell'attributo e la chiave esterna.
Nel diagramma ER considerato non ci sono attributi mutivalore, perciò tale passaggio non viene eseguito.

PASSO 7: Traduzioni di tipi di associazione N-arie
L'ultimo passaggio dell'algoritmo di risoluzione di uno schema ER in modello relazionale consiste nella traduzione dei tipi di associazione N-arie.
L'operazione di traduzione consiste, anche in questo caso, nella produzione di un nuovo schema di relazione avente tanti attributi di chiave esterna quante ne sono le relazioni dei corrispettivi tipi di entità e tanti attributi quanti sono gli attributi del tipo di associazione.

Nel diagramma ER in oggetto si trova un solo tipo di associazione N-aria: SERV_ACQ.



Come si può notare poiché SERV_ACQ è un tipo di associazione di 3° grado sono state inserite tre chiavi esterne, dove ognuna fa riferimento ad una relazione diversa riproduce il tipo di entità corrispondente. La chiave primaria è data dalla combinazione di tutte e tre le chiavi parziali.

3.2 Normalizzazione dello schema di base di dati relazionale

Al termine del passaggio dal modello dei dati concettuale (digramma ER) al modello dei dati logico (modello relazionale in questo caso) l'ultima operazione da eseguire è la normalizzazione, che consente di calcolare in modo oggettivo la qualità della progettazione.
La normalizzazione è un processo che consiste nel modificare un oggetto per renderlo più conforme a qualche criterio di regolarità.
Quest'ultima viene eseguita sottoponendo i vari schemi di relazione a 3 livelli di normalizzazione.
I "test" utilizzati sono stati:

- 1NF (1 Forma Normale)
- 2NF (2 Forma Normale)
- 3NF (3 Forma Normale)

L'attività di normalizzazione è guidata da quelle che sono le chiavi di uno schema di relazione e le dipendenze funzionali definite su ogni schema.
La dipendenza funzionale (DF), indicata con $X \rightarrow Y$, è un vincolo sulle tuple che possono formare uno stato di relazione di uno schema di relazione. Nello specifico dato uno schema di relazione, siano X e Y due insiemi di attributi sottoinsiemi del dato schema, il vincolo è che, per ogni coppia di tuple t_1 e t_2 nello stato di relazione dello schema per le quali $t_1[x]=t_2[x]$ allora anche $t_1[Y]=t_2[Y]$, ovvero i valori della componente X di una tupla determinano univocamente (o funzionalmente) i valori della componente Y.

PRIMA FORMA NORMALE (1NF)
La prima forma normale richiede che il dominio di un attributo di uno schema di relazione sia composto di soli valori atomici e che il valore di qualsiasi attributo di una tupla sia un valore singolo del dominio corrispettivo.
Nel nostro caso è già soddisfatta.

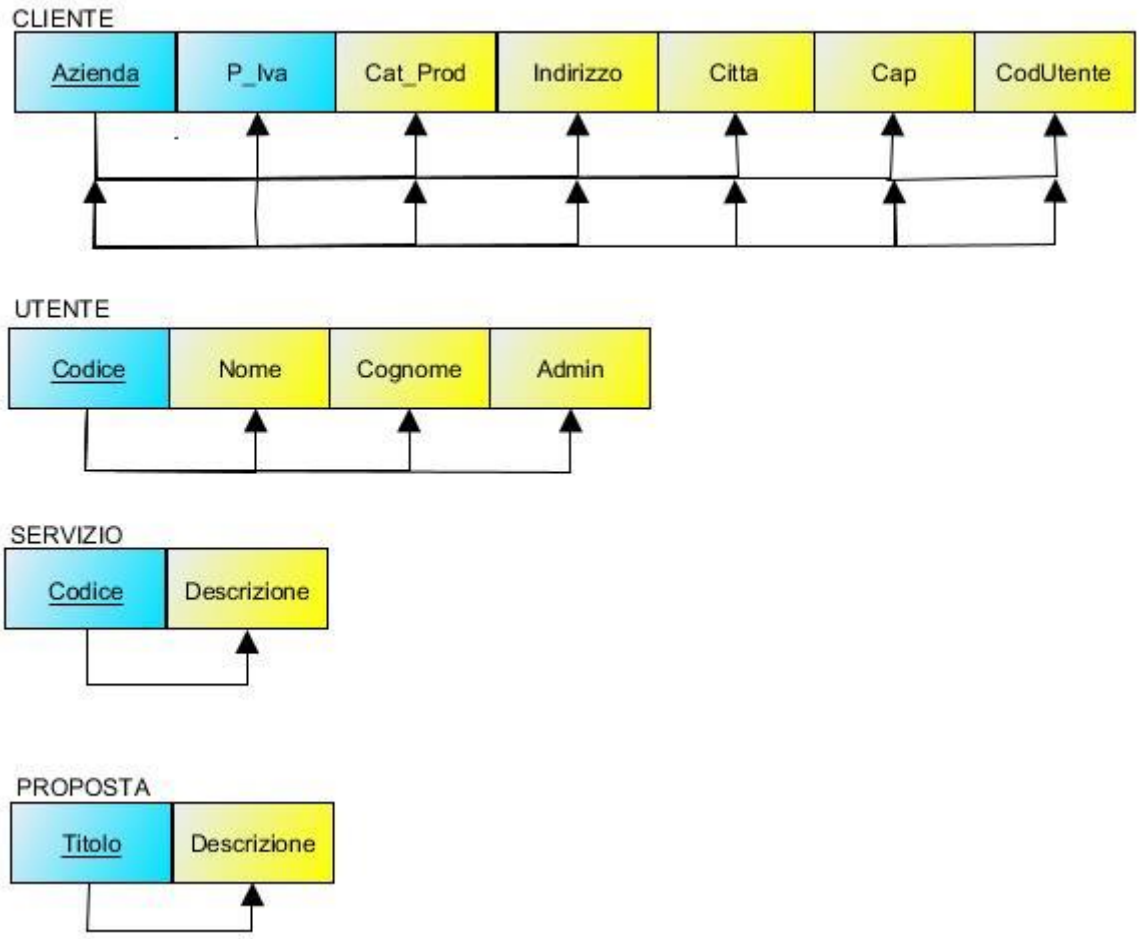
SECONDA FORMA NORMALE (2NF)
La seconda forma normale si basa sul concetto di dipendenza funzionale completa.
Una DF $X \rightarrow Y$ di uno schema di relazione è una dipendenza funzionale completa (DFC) se la rimozione di qualsiasi attributo A da X comporta che la DF non sussiste più, nel caso in cui ciò non sia verificato si parla di dipendenza funzionale parziale (DFP).
Uno schema di relazione si dice in 2NF se ogni attributo non-primo (un attributo non facente parte di nessuna chiave) dello schema dipende in modo funzionale e completo da ogni chiave dello schema.
Lo studio per il passaggio alla 2NF dello schema di base di dati relazionale progettato verrà analizzato in seguito.

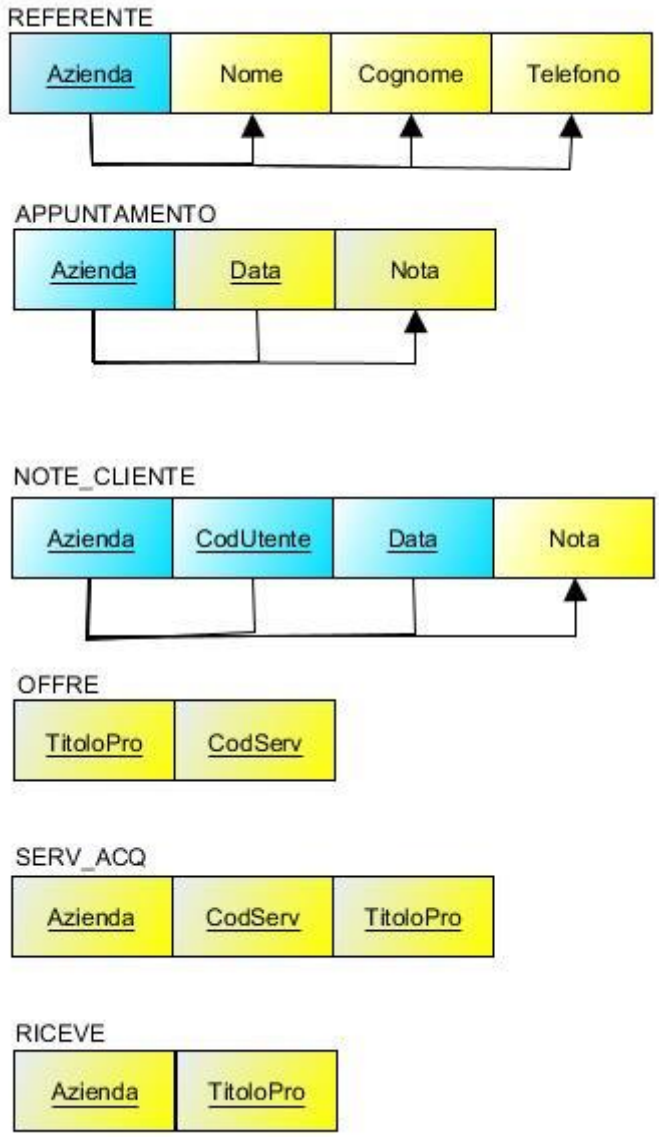
TERZA FORMA NORMALE (3NF)

La terza forma normale, a differenza della 2NF, si basa anche sulla dipendenza funzionale transitiva, andando ad eliminare ulteriori ridondanze.
Una DF X → Y di uno schema di relazione è una dipendenza funzionale transitiva (DFT) se esiste Z, sottoinsieme di attributi dello schema, che non è né chiave candidata ne sottoinsieme di una chiave dello schema, tale per cui valgono contemporaneamente X → Z ed Z → Y.

PROCESSO DI NORMALIZZAZIONE

Nel seguito si analizzeranno le varie chiavi e dipendenze funzionali ai fini di poter ridurre lo schema relazionale della base di dati in 2NF o 3NF.

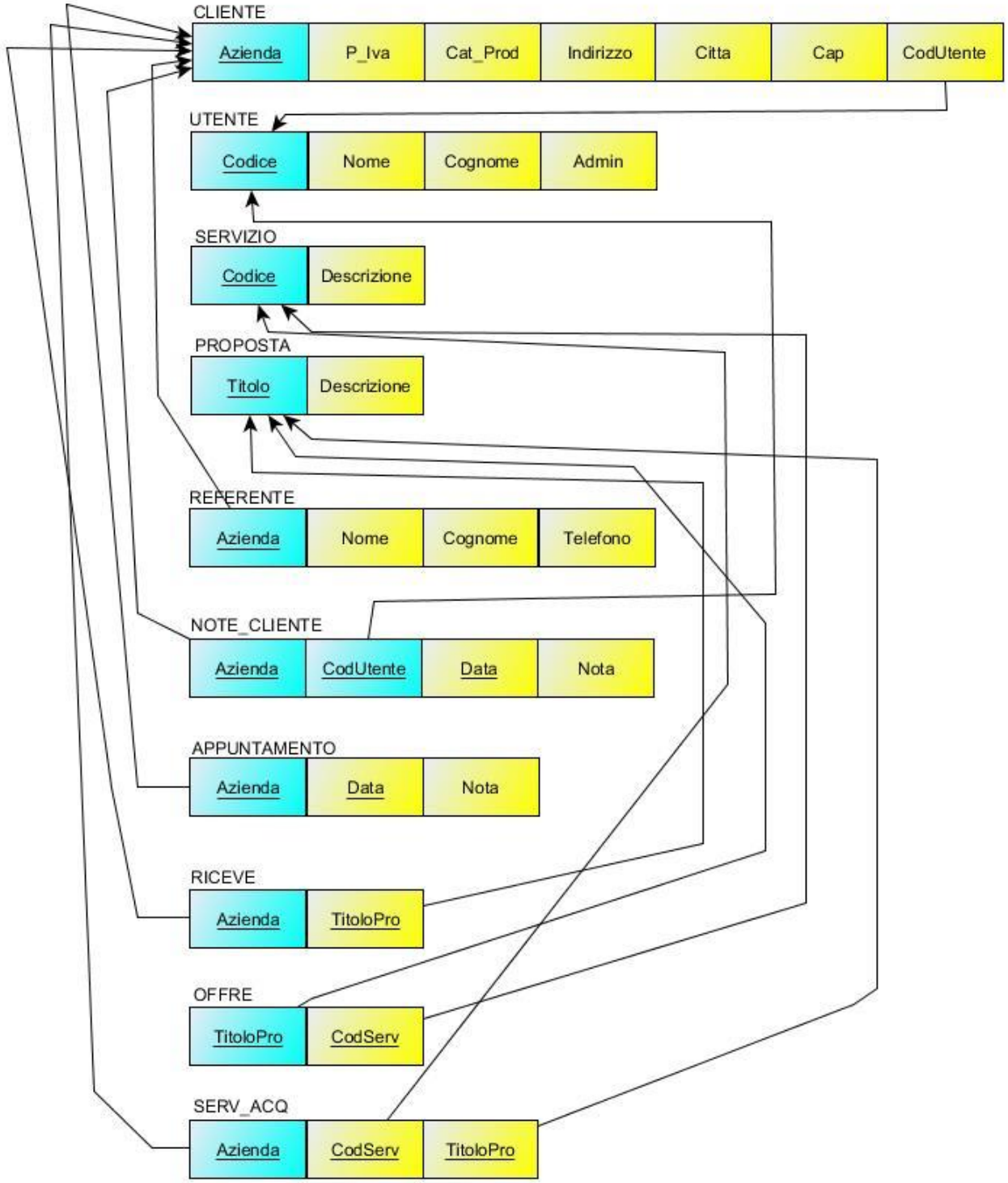




Partendo dalle relazioni più semplici, RICEVE, OFFRE e SERV_ACQ essere risultano già essere in 3NF e quindi anche in 2NF poiché presenta solo attributi chiave.
Analizzando le relazioni REFERENTE, PROPOSTA, SERVIZIO e UTENTE esse sono caratterizzate da un'unica chiave semplice e da un'unica dipendenza funzionale, poiché ogni attributo non-primo delle rispettive relazioni dipende obbligatoriamente in modo completo dalla corrispettiva chiave e non essendovi dipendenze funzionali transitive tutte le relazioni sono in 3NF.
La relazione APPUNTAMENTO presenta un unico attributo non-primo (Nota), il quale dipende in modo funzionale e completo dalla chiave (Azienda, Data), infatti togliendo, a titolo di esempio, l'attributo Azienda, non si è più in grado di individuare un appuntamento, ne segue che la relazione è in 2NF. Non essendovi successivamente altri attributi primi per i quali possa sussistere una DFT ne consegue che è anche in 3NF.
Analogamente alla relazione APPUNTAMENTO anche NOTE_CLIENTE è in 3NF, perché non vi sono DFT e presenta un unico attributo primo (Nota) che ha una

dipendenza funzionale completa dalla chiave composta, infatti eliminando l'attributo Data non si possono più individuare due note rivolte alla stessa azienda e memorizzate da uno stesso utente, ma in date diverse.
Infine la relazione CLIENTE seppure avente due chiavi semplici e due dipendenze funzionali, risulta che ogni attributo non-primo ha una dipendenza funzionale completa da ogni chiave, non essendovi per giunta delle DFT si deduce che la relazione CLIENTE è in 3NF.

3.3 Schema di base di dati relazionale completo in 3NF



CAPITOLO 4: *Codice SQL*

4.1 Codice SQL per la realizzazione di tabelle

```
-- ------------------------------------
-- Struttura tabella UTENTE
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `UTENTE` (
  Codice INTEGER          AUTO_INCREMENT,
  Nome VARCHAR(20)        NOT NULL,
  Cognome VARCHAR(20)     NOT NULL,
  Admin TINYINT(1)        NOT NULL DEFAULT 0,
  PRIMARY KEY(Codice)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;




-- ------------------------------------
-- Struttura tabella CLIENTE
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `CLIENTE` (
  Azienda VARCHAR(30),
  P_Iva CHAR(11)          NOT NULL,
  Cat_Prod VARCHAR(10)    NOT NULL,
  Indirizzo VARCHAR(80)   NOT NULL,
  Citta VARCHAR(20)       NOT NULL,
  Cap CHAR(5)             NOT NULL,
  CodUtente INTEGER,
  PRIMARY KEY(Azienda),
  FOREIGN KEY(CodUtente) REFERENCES UTENTE(Codice),
  UNIQUE(P_Iva),
  INDEX(P_Iva),
  INDEX(Cat_Prod),
  INDEX(CodUtente)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
-- ------------------------------------
```

```
 -- ------------------------------------
-- Struttura tabella PROPOSTA
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `PROPOSTA` (
  Titolo VARCHAR(40),
  Descrizione TEXT        NOT NULL,
  PRIMARY KEY(Titolo)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;


-- Struttura tabella SERVIZIO
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `SERVIZIO` (
  Codice INTEGER          AUTO_INCREMENT,
  Descrizione TEXT        NOT NULL,
  PRIMARY KEY(Codice)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;



-- ------------------------------------
-- Struttura tabella REFERENTE
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `REFERENTE` (
  Azienda VARCHAR(30),
  Nome VARCHAR(20)        NOT NULL,
  Cognome VARCHAR(20)     NOT NULL,
  Telefono INTEGER UNSIGNED NOT NULL,
  FOREIGN KEY(Azienda) REFERENCES CLIENTE(Azienda),
  PRIMARY KEY(Azienda)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;


-- ------------------------------------
-- Struttura tabella NOTE_CLIENTE
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `NOTE_CLIENTE` (
  Azienda VARCHAR(30),
  CodUtente INTEGER,
  Data DATETIME,
  Nota TEXT               NOT NULL,
  FOREIGN KEY(Azienda) REFERENCES CLIENTE(Azienda),
  FOREIGN KEY(CodUtente) REFERENCES UTENTE(Codice),
  PRIMARY KEY(Azienda, CodUtente, Data)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-- ------------------------------------
-- Struttura tabella APPUNTAMENTO
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `APPUNTAMENTO` (
  Azienda VARCHAR(30),
  Data DATETIME,
  Nota TEXT                NOT NULL,
  FOREIGN KEY(Azienda) REFERENCES CLIENTE(Azienda),
  PRIMARY KEY(Azienda, Data)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;


-- ------------------------------------
-- Struttura tabella OFFRE
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `OFFRE` (
  TitoloPro VARCHAR(40),
  CodServ INTEGER,
  FOREIGN KEY(TitoloPro) REFERENCES PROPOSTA(Titolo),
  FOREIGN KEY(CodServ) REFERENCES SERVIZIO(Codice),
  PRIMARY KEY(TitoloPro, CodServ)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;


-- ------------------------------------
-- Struttura tabella RICEVE
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `RICEVE` (
  Azienda VARCHAR(30),
  TitoloPro VARCHAR(40),
  FOREIGN KEY(Azienda) REFERENCES CLIENTE(Azienda),
  FOREIGN KEY(TitoloPro) REFERENCES PROPOSTA(Titolo),
  PRIMARY KEY(Azienda, TitoloPro)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;


-- ------------------------------------
-- Struttura tabella SERV_ACQ
-- ------------------------------------
CREATE TABLE IF NOT EXISTS `SERV_ACQ` (
  Azienda VARCHAR(30),
  CodServ INTEGER,
  TitoloPro VARCHAR(40),
  FOREIGN KEY(Azienda) REFERENCES CLIENTE(Azienda),
  FOREIGN KEY(CodServ) REFERENCES SERVIZIO(Codice),
  FOREIGN KEY(TitoloPro) REFERENCES PROPOSTA(Titolo),
```

```
  PRIMARY KEY(Azienda, CodServ, TitoloPro)
)ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Elemento particolare è l'utilizzo della keyword AUTO_INCREMENT nelle tabelle CLIENTE ed SERVIZIO che garantiscono un incremento automatico dell'attributo considerato ogni qual volta che viene effettuato un nuovo inserimento.
Per la tabella CLIENTE sono stati, inoltre, creati 3 indici secondari basati rispettivamente sui campi di indicizzazione P_Iva, Cat_Prod ed CodUtente al fine di ottimizzare una possibile ricerca basata su uno degli attributi citati.

4.2 Asserzioni e Trigger

Nelle righe precedenti sono stati riportati i comandi che permettono di creare all'intero di MySQL delle relazioni.
L'istruzione CREATE TABLE, tuttavia, non permette di definire quelli che sono i vincoli basati sull'applicazione (vincolo di stato e vincolo di transizione), ovvero quelli collegati con il funzionamento del programma. In generale, per definire questi vincoli si ricorre a linguaggi di specifica dei vincoli oppure a meccanismi tipo asserzioni e vincoli.
L'asserzione è un meccanismo che permette di definire delle condizioni su una tabella che devono essere sempre verificate, in ogni condizione.
Il trigger permette, invece, di esprimere le operazioni da svolgere al verificarsi di determinate situazioni e sotto determinate condizioni.
Da sottolineare che, seppur parlando di asserzioni, purtroppo MySQL non le implementa quindi per poterle imitare si è ricorso all'uso dei trigger, dove quando una determinata condizione non era verificata si poneva un attributo ad un valore che non gli era ammesso al fine di scatenare un errore e bloccare l'inserimento di un record.

```
-- ----------------------------------------
-- Definizione di alcune ASSERZIONI/TRIGGER
-- ----------------------------------------
DELIMITER //

CREATE TRIGGER CHECK_NOME_COGNOME_UTENTE BEFORE INSERT ON `UTENTE` FOR EACH ROW
BEGIN
  IF NEW.Nome REGEXP '[0-9]' THEN SET NEW.Nome = NULL;
  END IF;
  IF NEW.Cognome REGEXP '[0-9]' THEN SET NEW.Cognome = NULL;
  END IF;
END;//

CREATE TRIGGER CHECK_NOME_COGNOME_REFERENTE BEFORE INSERT ON `REFERENTE` FOR
EACH ROW
BEGIN
  IF NEW.Nome REGEXP '[0-9]' THEN SET NEW.Nome = NULL;
  END IF;
  IF NEW.Cognome REGEXP '[0-9]' THEN SET NEW.Cognome = NULL;
  END IF;
END;//
```

```
-- ------------------------------------
-- Definizione di alcuni TRIGGER
-- ------------------------------------
CREATE TRIGGER CHECK_CAP BEFORE INSERT ON `CLIENTE` FOR EACH ROW
BEGIN
  IF CHAR_LENGTH(NEW.Cap)<5 THEN SET NEW.Cap = LPAD(NEW.Cap, 5, "0");
  END IF;

-- Asserzione/Trigger
  IF CHAR_LENGTH(NEW.P_Iva)<11 THEN SET NEW.P_Iva = NULL;
  END IF;
END;//


CREATE TRIGGER CHECK_DATA_NOTE BEFORE INSERT ON `NOTE_CLIENTE` FOR EACH ROW
BEGIN
  IF NEW.Data > NOW() THEN SET NEW.Data = NOW();
  END IF;
END;//

CREATE TRIGGER CHECK_DATA_APP BEFORE INSERT ON `APPUNTAMENTO` FOR EACH ROW
BEGIN
  IF NEW.Data < NOW() THEN SET NEW.Data = NOW();
  END IF;
END;//

DELIMITER ;
```

Nel dettaglio le asserzioni/trigger CHECK_NOME_COGNOME_UTENTE ed CHECK_NOME_COGNOME_REFERENTE controllano se, prima di essere inseriti, i campi Nome e Cognome, delle rispettive tabelle contengano dei numeri, in tal caso imposta i loro valori a NULL e poiché essi non possono contenere tale valore viene generato un errore, proprio come se fossero delle asserzioni vere e proprie.
Il trigger CHECK_CAP controlla, prima che venga inserito, se il campo cap indicato è di 5 cifre, se non lo è vengono aggiunti tanti 0 quanti ne mancano per formare una stringa di 5 caratteri.
I trigger CHECK_DATA_NOTE ed CHECK_DATA_APP controllano, prima che venga inserito nelle rispettive tabelle, il campo DATA. Per il primo trigger se la data inserita è futura a quella attuale allora viene impostata all'ora attuale, perché non è logico avere una nota che fa riferimento ad una conversazione avvenuta nel passato ad un valore futuro. Per il secondo trigger la situazione è analoga, dato che un appuntamento riguarda un evento futuro la data inserita non può essere antecedente alla data attuale, per questo, nel caso di violazione, viene impostata all'ora attuale.

4.3 Le Viste

Nell'implementazione della base di dati per il sistema CMR sono state realizzate anche 2 view.
Una view è una singola tabella realizzata partendo da altre tabelle, che siano esse tabelle base o altre view. Una view è anche detta tabella virtuale in quanto non è fisicamente memorizzata sul database, ciò che viene memorizzato è l'istruzione che la crea, infatti, le view sono tenute sempre aggiornate proprio per questo motivo, poiché la tabella che esse rappresentano viene creata ogni volta al momento dell'invocazione.
Nel caso in esame le 2 view realizzate hanno lo scopo di facilitare alcune ricerche. Nello specifico la view `CLIENTE_UTENTE` permette una più pronta ricerca, basata sugli utenti, dei clienti, mentre la view `CLIENTE_REFERENTE` collega direttamente un'azienda con il suo referente e quindi migliora una possibile ricerca basata su nome e cognome del referente.

```
-- ------------------------------------
-- Definizione di alcune VISTE
-- ------------------------------------
CREATE VIEW CLIENTE_UTENTE AS SELECT C.*, U.Nome, U.Cognome
FROM CLIENTE AS C
INNER JOIN UTENTE AS U ON U.Codice = C.CodUtente;

CREATE VIEW CLIENTE_REFERENTE AS SELECT C.*, R.Nome, R.Cognome, R.Telefono
FROM CLIENTE AS C
INNER JOIN REFERENTE AS R ON R.Azienda = C.Azienda;
```

4.4 Interrogazioni, Inserimenti e Cancellazioni importanti

Nella realizzazione dell'applicazione web, basata sulla base di dati progettata e implementata sono state utilizzate le seguenti importati operazioni di interrogazione, inserimento e cancellazione.

QUERY D'INTERROGAZIONE

```
SELECT Descrizione, TitoloPro FROM SERV_ACQ INNER JOIN SERVIZIO ON
CodServ=Codice WHERE Azienda='$Azienda'
```

La sovrastante query unisce ogni tupla nella tabella SERV_ACQ, relative ad una determinata azienda, con il corrispettivo servizio e ne estrapola la descrizione e il titolo. Vengono elencate quelli che sono i servizi acquistati da un'azienda.
$Azienda è una variabile PHP.

```
SELECT Titolo, Descrizione FROM RICEVE INNER JOIN PROPOSTA ON TitoloPro=Titolo
WHERE Azienda='$Azienda'
```

Tale query collega ogni tupla nella tabella RICEVE, relative ad una determinata azienda, con la corrispettiva proposta e ne preleva la descrizione e il titolo. Vengono prelevate, quindi, tutte le proposte fatte ad una determinata azienda.

```
SELECT Codice, Descrizione FROM OFFRE INNER JOIN SERVIZIO ON Codice=CodServ
WHERE TitoloPro='$row[0]'
```

Tale operazione mette in relazione ogni tupla della tabella OFFRE, relative ad una determinata proposta, con i corrispettivi servizi offerti. Vengono, quindi, mostrati tutti quelli che sono i servizi offerti per una data proposta.
$row[0] è una variabile PHP che memorizza il titolo di una proposta.

QUERY DI CANCELLAZIONE

```
DELETE FROM NOTE_CLIENTE WHERE Azienda='$Azienda' AND Data='$Data' AND
CodUtente=$CodUtente
```

La presente query di cancellazione consente di eliminare dalla tabella NOTE_CLIENTE una nota associata ad un cliente, aggiunta in precedenza, indicandone il nome dell'azienda, la data e il codice dell'utente che l'aveva creata, in quanto la chiave primaria dalla tabella è data dagli attributi appena elencati.

QUERY D'INSERIMENTO

```
INSERT INTO CLIENTE VALUES('$Azienda', '$P_Iva', '$Cat_Prod', '$Indirizzo',
'$Citta', '$Cap', '$CodUtente')
```

Tale query permette ad un amministratore di inserire un nuovo cliente, nell'apposita tabella CLIENTE, tramite l'applicazione web.
$Azienda, $P_Iva, $Cat_Prod, $Indirizzo, $Citta, $Cap, $CodUtente sono variabili PHP che memorizzano rispettivamente nome dell'azienda del cliente, la partita IVA, la

categoria dei prodotti venduti dall'azienda, l'indirizzo della sede, con città e cap, e il codice dell'utente dell'applicativo al quale l'azienda è affidata.

```
INSERT INTO RICEVE VALUES('$Azienda', '$Titolo')
```

La sovrastante query consente di inserire all'interno della tabella RICEVE quelle che sono le proposte commerciali fatte ad un'azienda. I valori memorizzati sono riferimenti alle chiavi primarie delle tabelle CLIENTE e PROPOSTA.
$Azienda, $Titolo sono variabili PHP che memorizzano nome dell'azienda e titolo della proposta.

```
INSERT INTO SERV_ACQ VALUES('$Azienda', '$Servizio', '$Titolo')
```

Tale operazione permette, infine, di salvare nella tabella SERV_ACQ quelli che sono i servizi acquistati da un cliente memorizzando nome dell'azienda, servizio acquistato e in relazione a quale proposta commerciale, ovvero salvando le corrispettive chiavi primarie delle tuple nelle tabelle CLIENTE, SERVIZIO e PROPOSTA.
$Azienda, $Servizio, $Titolo sono variabili PHP.

QUERY DI AGGIORNAMENTO

```
UPDATE NOTE_CLIENTE SET Azienda='$Azienda', CodUtente=$CodUtente, Data='$Data', Nota='$Nota' WHERE Azienda='$Azienda' AND CodUtente=$CodUtente AND Data='$Data'
```

Quest'ultima interrogazione permette di modificare una nota nella tabella NOTE_CLIENTE, precedentemente creata e relativa ad un'azienda, selezionandola grazie alla clausola WHERE e modificandola con l'istruzione SET.

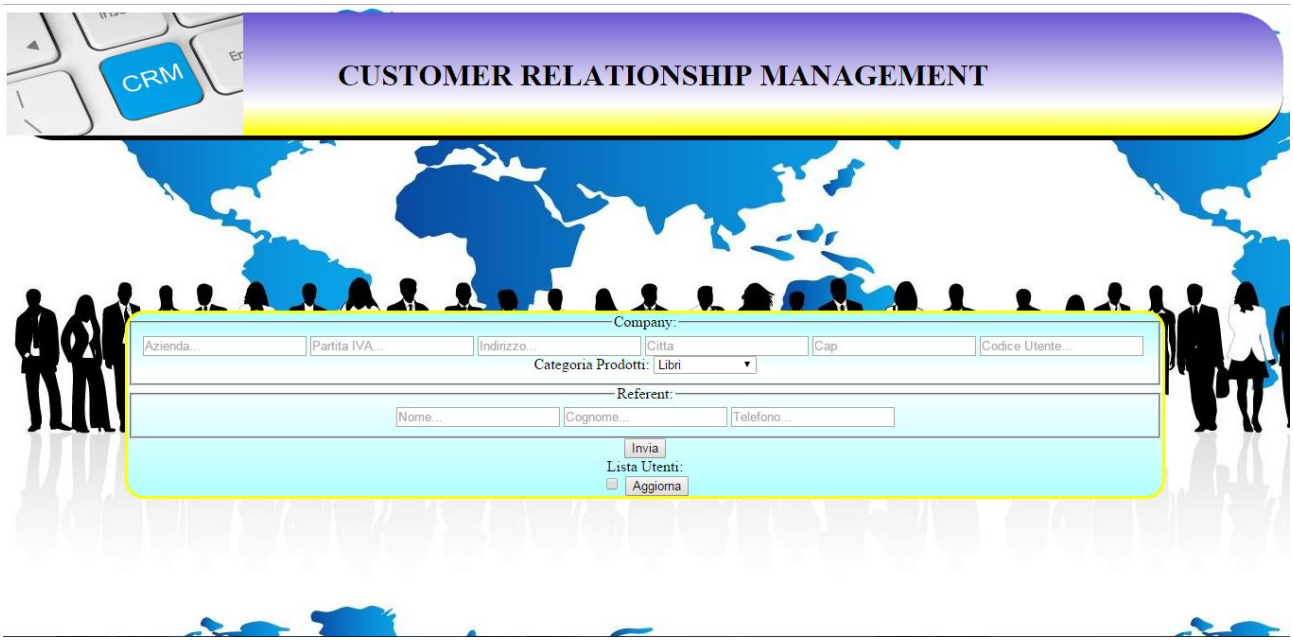
CAPITOLO 5: *Utenti ed Interfaccia*

5.1 Struttura applicazione e screenshot

Nelle successive pagine si riportano alcuni screenshot dell'applicazione realizzata. Essa è stata sviluppata combinando HTML, CSS e PHP e pensata per 3 categorie di utilizzatori, i clienti (alias le aziende), gli utenti reali utilizzatori dell'applicazione e gli amministratori.
Per ogni tipo di fruitore sono state realizzate delle homepage personali attraverso le quali si può accedere, inserendo il nome dell' azienda o il codice personale nel caso di utenti ed amministratori, alle rispettive operazioni.
L'applicazione seppur molto semplice presenta numerosi spunti per l'implementazione di un sistema reale e complesso di Customer Relationship Management.

***Homepage Principale**: suddivisione dei servizi secondo la categoria di utilizzatori*

***Homepage Admin**: vista delle operazioni concesse all'amministratore*



***Nota**: Esempio inserimento di un nuovo cliente da parte dall'amministratore*

***Homepage Utente**: vista delle operazioni ammesse all'utente*

***Nota**: Esempio di vista dei clienti da parte di un utente*

***Homepage Cliente**: vista delle operazioni possibili per un cliente*